ECOLE NORMALE SUPÉRIEURE DE CACHAN
ECOLE DOCTORALE DE SCIENCES PRATIQUES

THÈSE

présentée à
L'ECOLE NORMALE SUPÉRIEURE DE CACHAN
pour obtenir le grade de
DOCTEUR ÈS SCIENCES

Spécialité :
INFORMATIQUE

par

Guy Alain NARBONI

Un cas remarquable de systèmes linéaires : les systèmes monotones

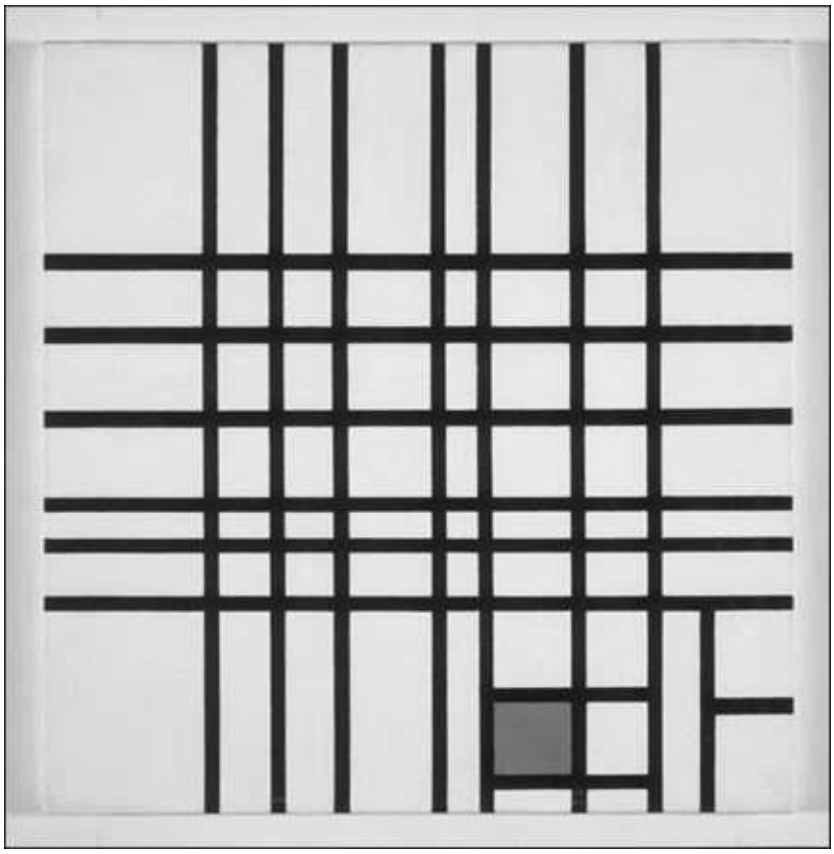
Résolution et application à la vérification formelle de programmes

Soutenue le 20 décembre 2001 devant le jury composé de :

MM.	Frédéric BENHAMOU	Rapporteur
	Alexander BOCKMAYR	Examinateur
	Alain COLMERAUER	Président du jury
	Alain FINKEL	Examinateur
	Laurent FRIBOURG	Directeur de thèse
	Michel RUEHER	Rapporteur

Laboratoire Spécification et Vérification - CNRS UMR 8643
ENS de Cachan — 61, Avenue du Président Wilson — 94 235 Cachan Cedex

© implexe
2001



Mondrian : Composition
The Transatlantic paintings
Cat. 5, No. 12, 1936-1942.

A toi
qui connaissais
la couleur du temps

Remerciements

A Alain Finkel, tout d'abord, pour la confiance qu'il m'a témoignée en m'acceptant dans cette prestigieuse Ecole doctorale. Je retiendrai son rôle de *coach* bénévole, pour canaliser (et c'était nécessaire !) mes efforts et mon ambition originale.

A Laurent Fribourg, mon cher directeur de thèse, pour m'avoir permis de renouer une collaboration scientifique, après une longue mise en sommeil ... qu'on aurait pu croire définitive si elle n'avait été entretenue par la carrière et favorisée par l'existence. La vérification de propriétés sur les réseaux de Petri (papier présenté au colloque VCL 2000) sera l'un de ces déclics. Aussi me vois-je à l'arrivée dépositaire d'un double record : celui de la thèse la plus longue ... et paradoxalement la plus courte.

A mon ancien «patron» Alain Colmerauer pour m'avoir fait l'honneur et l'amitié de présider ce jury. Ses remarques m'auront aidé à dépasser plus d'un blocage.

A mes rapporteurs Michel Rueher et Frédéric Benhamou, «client» et «collègue» de PrologIA, pour avoir bien voulu s'investir dans la lecture de ce manuscrit qui sollicitait leur expertise sur les techniques de filtrage par intervalles.

A Alexander Bockmayr pour l'intérêt qu'il a porté spontanément à ce travail, à cheval sur les domaines de la programmation linéaire et de la programmation par contraintes. Les résonances avec son récent opuscule paru dans le très savant *Handbook of Automated Reasoning* démontrent une indéniable communauté de vues.

A Mehmet Dincbas et à la société Cosytec qu'il dirige, pour m'avoir donné l'occasion de présenter mes recherches, en 1999 et 2000, lors des journées du Club des utilisateurs du langage CHIP. L'exposé sur les plus courts chemins (modèles duaux et relaxations parfaites) sera repris sur invitation à la conférence de Londres, APMOD 2000, en recherche opérationnelle.

Au conservateur de la bibliothèque de l'Ecole Nationale des Ponts et Chaussées pour m'avoir permis, en 1990 (avant le déménagement du fonds à Marne la Vallée), de feuilleter l'exemplaire — dédicacé au Baron de Prony — du traité posthume de Fourier publié en 1830 par Navier (mémoire n° 267 BN V7313). Le septième et dernier livre, inachevé, devait exposer le principe de l'analyse des inégalités.

Au professeur H. Paul Williams de l'Université de Southampton pour m'avoir aimablement communiqué ses papiers sur l'extension de l'élimination de Fourier aux variables entières.

Aux professeurs Jean-François Maurras et Pierre Siegel, à Marseille, pour leurs éclairages respectifs sur les problèmes de flots et la résolution unitaire.
Aux professeurs Gaubert, Hochbaum, Rockafellar et Veinott, sur Internet, qui, sans me connaître, ont bien voulu m'assister dans mes recherches sur l'origine et l'usage du terme «monotone».

A Pascal Bouvier pour son assistance technique sur Prolog III et Prolog IV.

A Mme Alecca Sengakis du service de documentation de l'ENS.

A mon cabinet, Implexe, qui m'a permis de financer en partie ces travaux.

A mes soutiens de toujours — plus que jamais.

Avertissement

Ce mémoire est un travail d'ingénieur. Il se veut thèse autant que synthèse. S'il y rentre quelques mathématiques, c'est pour prouver des propriétés simples mais importantes, à nos yeux, en pratique.

A quoi peut-on se raccrocher en effet lorsqu'on a posé un problème et qu'il s'avère de nature combinatoire ? Les techniques modernes de programmation par contraintes nous permettent de formuler très aisément de telles questions. Elles ne garantissent pas pour autant une meilleure efficacité de leur résolution.

Comme en calcul des structures, on se doit de recourir à une estimation de «complexité» pour dimensionner une application. Il se trouve que pour les systèmes linéaires qualifiés ici de «monotones» cette complexité reste raisonnable. Nous avançons de plus des résultats de complétude qui laissent ouvert le choix des solveurs (entendons ici, entre les solveurs linéaires et les solveurs d'intervalles des langages d'aujourd'hui).

Bien entendu, pour des problèmes «purs», les algorithmes spécialisés seront toujours plus efficaces que des solveurs généralistes. Mais, dans une philosophie de modélisation par contraintes, on souhaitera toujours pouvoir combiner des exigences pour s'approcher du problème réel, quitte à s'écarter d'un modèle mathématiquement pur. Ce dernier devient un sous-problème du problème à résoudre. Sa résolution donne une «relaxation». C'est à son aune que l'on pourra apprécier, en définitive, la pertinence de telle ou telle formulation.

«Non, non, non... Je fabrique ma petite terminologie, suivant les besoins ; — mais je la garde en général pour mon usage personnel et privé... Ce sont mes outils intimes. Je me fais mes ustensiles, et les fais pour moi seul : aussi individuels et adaptés que possible à ma manière de concevoir, et de combiner.»

P. VALERY

Abstract

Deciding whether or not a linear system of inequalities has integer solutions is an important sub-problem occurring in many case studies. It is well known that such a question of diophantine nature can be a formidable source of combinatorial difficulties.

In this thesis, we focus on a class of integer linear problems that can be solved without enumeration (either directly or iteratively, if the problem is bounded) — their relaxation giving here an exact solution. This 'polynomial' class is characterized by a remarkable confluence of syntactic and geometric properties (matrices having at most one positive entry per row, concording with solutions sets forming a semi-lattice).

We show that local consistency techniques based on interval narrowing are complete for that class (when the domains are bounded). This stems from the fact that one particular solution is a global maximum (or minimum).

The systems under study encompass questions ranging from the satisfiability problem for Horn propositional clauses to the computation of shortest paths on a graph. By comparatively studying their solution algorithms, we are able to establish connections between the classical procedures of variable *elimination* and discrete *optimization*, and the more recent interval (or finite domains) *approximation* solving techniques.

A more elaborate application, requiring the power of contraint logic programming, deals with reachability analysis on discrete state machines. Our implementations, using either the linear solver of Prolog III or the interval solver of Prolog IV, enable us to prove several program properties automatically, on examples taken from the field of formal verification.

Keywords :

CONSTRAINT LOGIC PROGRAMMING, INTERVAL ANALYSIS,
FORMAL METHODS, FOURIER ELIMINATION,
LEONTIEF MATRICES, LINEAR & INTEGER PROGRAMMING.

Résumé

Savoir décider si oui ou non un système d'inégalités linéaires admet une solution entière est un sous-problème important dans beaucoup d'études de cas. Il est bien connu que cette question, de nature diophantienne, peut être source de difficultés combinatoires.

Nous exhumons, dans ce travail, une classe particulière de problèmes linéaires à variables entières dont la résolution exacte peut s'obtenir sans énumération, à partir d'une relaxation continue (soit directement, soit itérativement sur des problèmes bornés). Cette classe «polynômiale» se caractérise par une concordance remarquable entre des propriétés syntaxiques (matrices ayant au plus un coefficient strictement positif par ligne) et des propriétés géométriques (ensemble des solutions formant un demi-treillis).

Nous montrons que les techniques de filtrage opérant localement par réduction d'intervalles sont complètes pour cette classe (lorsque les domaines sont bornés), du fait de l'existence d'une solution particulière qui est un extrêmum global.

Les systèmes étudiés recouvrent des problèmes aussi variés que la satisfaction de clauses de Horn en logique propositionnelle ou la recherche de plus courts chemins sur un graphe. En abordant leur résolution de manière comparée, nous établissons des ponts entre les procédures classiques d'*élimination* de variables et d'*optimisation* discrète et les techniques plus récentes d'*approximation* par intervalles.

Une application plus élaborée concerne l'analyse, en programmation logique avec contraintes, de l'accessibilité dans des machines à états discrets. Les implantations réalisées avec le solveur linéaire de Prolog III ou le solveur d'intervalles de Prolog IV nous permettent de prouver automatiquement plusieurs propriétés, sur des exemples tirés du domaine de la vérification formelle de programmes.

Mots clés :

ELIMINATION DE FOURIER, INTERVALLES, MATRICES DE LEONTIEF, MÉTHODES FORMELLES, PROGRAMMATION LINÉAIRE EN NOMBRES ENTIERS, PROGRAMMATION LOGIQUE AVEC CONTRAINTES.

CHAPITRE I

Introduction

Cadre général, conventions et notations

> «Les philosophes d'Alexandrie ont connu quelques éléments d'un art qui a pour objet la grandeur mesurable et qui consiste à suppléer aux opérations de l'esprit par la combinaison d'un petit nombre de signes. Cet art a pris un grand essor chez les nations modernes ; il est devenu l'analyse mathématique, science sublime qui, en nous découvrant les lois générales du mouvement et celles de la chaleur, explique tous les grands phénomènes de l'univers, et qui éclaire la société civile dans ses usages les plus importants.»
>
> J. FOURIER

La recherche opérationnelle a développé des méthodes générales et puissantes pour calculer des solutions particulières d'un système linéaire. Sous la forme d'un «programme linéaire», elle exprime classiquement un problème d'optimisation :

$$\min \{\boldsymbol{cx} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$$

qui définit le coût minimal $\boldsymbol{cx}$ d'une solution.

Plus largement, elle permet de répondre par oui ou par non à des questions que l'on peut se poser au cours d'un raisonnement logique, dans un contexte de preuve notamment, dès lors que des relations arithmétiques sont en jeu :

- $\exists \boldsymbol{x}$ *tq.* $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ (élimination de quantificateurs existentiels) :
  L'ensemble $\{\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$ est-il non vide ?
  Les contraintes numériques sont-elles satisfaisables ? L'objectif est-il réalisable ?
- $\forall \boldsymbol{x}$ $(\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b} \Rightarrow \mathbf{A'}\boldsymbol{x} \geq \boldsymbol{b'})$ (élimination de quantificateurs universels) :
  L'ensemble $\{\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$ est-il inclus dans $\{\boldsymbol{x} \mid \mathbf{A'}\boldsymbol{x} \geq \boldsymbol{b'}\}$?
  Y a-t-il redondance ou implication logique de contraintes ?

Si l'on change légèrement la question pour s'interroger sur l'existence de solutions à coordonnées entières à l'intérieur des limites spécifiées par les inégalités, il est bien connu que l'on passe d'un problème relativement «traitable» à un problème qui peut être de nature «NP-difficile» (avec un risque inhérent d'explosion combinatoire).

La formulation linéaire continue ne donne plus alors qu'une approximation de l'ensemble des solutions, au même titre que d'autres approximations conservatives mises en avant pour encadrer efficacement, par le calcul, les régions solutions de problèmes discrets, voire non-linéaires.

A partir du moment où plusieurs formulations entrent en concurrence, la difficulté devient méthodologique.

Les languages de «programmation en logique», qui nous intéressent ici pour les études de cas formelles et l'automatisation de la preuve, offrent ainsi, sous leur volet «contraintes», une alternative entre deux grandes techniques de résolution :

- l'une, plutôt algébrique, fondée sur l'élimination de Gauss et l'optimisation par l'algorithme du Simplexe,
- l'autre, plutôt analytique, fondée sur un mécanisme de réduction d'intervalles opérant par point fixe.

Comment choisir ?
Au delà des résultats techniques énoncés, l'ambition de ce travail est de contribuer, dans le long terme, à guider une démarche de conception.
Ce point nous semble particulièrement important dans le domaine des «ouvrages d'art combinatoires», où il n'existe pas de solution informatique préfabriquée.

Nous apportons un éclairage théorique sur une classe assez restreinte de systèmes linéaires mais non dénuée d'intérêt en pratique. Cette classe est aisée à définir syntaxiquement. Elle comprend les systèmes dont la matrice **A** vérifie la propriété (*m*) : *avoir au plus un coefficient > 0 par ligne*.

Cette définition caractérise des ensembles qui sont des demi-treillis. Ces derniers présentent, lorsqu'ils sont bornés, une propriété géométrique élémentaire qui a trait à l'existence d'un plus petit (ou plus grand) élément.

De cette remarque, nous tirons des théorèmes d'approximation et de complétude prédisant un bon comportement des algorithmes envisagés, tant pour la résolution de problèmes linéaires continus que, sous certaines conditions de divisibilité, pour la résolution effective de problèmes diophantiens.

En particularisant notre modèle, nous retrouvons bien entendu des résultats connus, mais dont les rapprochements sont peut-être plus inattendus.

Nous appliquons enfin ces connaissances à la vérification, en programmation logique avec contraintes, de propriétés de sûreté d'automates à compteurs. Dans cette dernière application, la résolution de contraintes intervient au sein d'un parcours de graphe spécifié par un programme logique. Toutes les subtilités algorithmiques des manipulations d'intersection et d'inclusion d'ensembles sont alors exigées pour remplacer l'exploration d'un graphe infini d'états par celle d'un graphe possiblement fini d'ensembles d'états.

1 Conventions

Cette thèse parle de *polyèdres*, ou plus précisément de polyèdres convexes fermés :

- polyèdres parce qu'ils s'agit de figures définies dans un espace (multi-dimensionnel), par un ensemble fini d'inégalités linéaires qui en délimitent les faces
- convexes, si l'on ne considère que des conjonctions d'inégalités (nous traiterons de façon explicite les cas de disjonctions)
- fermés, car toutes les inégalités sont prises au sens large.

Nous définirons donc un polyèdre comme l'intersection d'un nombre fini de demi-espaces (décrits chacuns par une contrainte, ou relation d'inégalité).

La théorie logique à laquelle nous nous réfèrerons aura :

- pour constantes : les réels (ou les entiers),
- pour fonction : +
- pour relation : ≤.

Nous nous intéresserons à résoudre des systèmes de contraintes sur les réels (ou les entiers) que nous noterons : $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$, écriture matricielle d'une formule logique exprimant des conjonctions d'inégalités sur, disons, n variables si l'on se situe dans un espace de dimension n.

Une solution de ce système est une affectation des variables qui vérifie chacune des inégalités. C'est donc un point du polyèdre $\mathrm{P} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$, ou, si l'on raisonne en nombres entiers, un point de l'ensemble $\mathrm{P} \cap \mathbb{Z}^n$.

Par résoudre, nous entendrons vérifier que le système admet des solutions, réelles ou entières, autrement dit, vérifier que $\mathrm{P} \neq \emptyset$ ou que $\mathrm{P} \cap \mathbb{Z}^n \neq \emptyset$.

Nous dirons que deux systèmes sont équivalents s'ils ont le même ensemble de solutions.

Pour des raisons de représentation des nombres en machine, nous restreindrons — sans grande conséquence — notre étude aux polyèdres rationnels, pour lesquels les coefficients de $\mathbf{A}$ et $\boldsymbol{b}$ sont des nombres rationnels. Un tel polyèdre peut être défini de façon équivalente par un système où $\mathbf{A}$ et $\boldsymbol{b}$ sont entiers.

Résoudre un système $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ sur les réels est équivalent à le résoudre sur les rationnels, mais n'est pas équivalent à le résoudre sur les entiers.

Si le système admet des solutions, nous dirons qu'il est soluble, satisfaisable ou cohérent. Le problème de la *satisfaisabilité* (mot barbare, employé par référence au problème SAT) est un problème logique de décision : la réponse attendue est oui ou non. Bien sûr, si le problème est soluble, nous voudrons en exhiber une solution (nous nous intéresserons donc à des algorithmes constructifs).

Cette solution pourra être choisie en butée, de sorte à minimiser ou maximiser un critère spécifique. On parlera alors de problème d'*optimisation* (OPT).

Dans les cas où l'énoncé admet des solutions multiples, on pourra vouloir connaître les plages numériques dans lesquelles ces solutions se situent. On aura alors affaire à un problème de *simplification*.

Ce problème de la simplification s'assimile au calcul des projections sur les axes. Nous verrons, avec Fourier, que l'opération de projection parallèlement à un axe est tout à fait générale et correspond à l'élimination de variables en algèbre, ou encore à l'élimination de quantificateurs en logique.

On pourra exiger d'être seulement informé du fait qu'une inconnue, x, n'a plus aucun de degré de liberté : sa projection se réduit alors à un point (sa valeur est fixée dans toute affectation). Cette capacité d'inférence devient importante dans le cas de raisonnements avec des inégalités strictes (ou des contraintes de différence $\neq$), mais ce point sort du cadre de cette dissertation.

Enfin, nous nous intéresserons à la *complexité* de toutes ces opérations, qui dans la théorie considérée, restent décidables. La mesure de la complexité donne une indication de la difficulté de la «montée en charge», car en passant d'un problème miniature à un problème réel, on change souvent d'échelle.

Autant que faire se peut, nous tenterons d'échapper à la complexité triplement exponentielle de la résolution des inégalités linéaires sur les entiers (dans ce que l'on appelle la théorie arithmétique de Presburger) pour nous rabattre sur le problème beaucoup plus raisonnable et prévisible de leur résolution sur les réels. Un résultat justement célèbre de Khachiyan, publié en 1979, donne en effet une borne polynômiale au temps de calcul d'une solution — qui est le temps incompressible d'une vérification formelle.

2 Notations

Dans toute cette thèse, nous noterons :

$\mathbb{R}$	l'ensemble des réels
$\mathbb{Z}$	l'ensemble des entiers relatifs
a	une donnée ou constante
x	une inconnue ou variable
$\lfloor a \rfloor$	la partie entière de a (avec arrondi par défaut)
X	un intervalle de $\mathbb{R}$ (également noté I)
ax	un monôme du premier degré
$\boldsymbol{a}$	un vecteur de constantes
$\boldsymbol{x}$	un vecteur de variables
$\boldsymbol{X}$	un produit cartésien d'intervalles
$\boldsymbol{ax}$	une forme linéaire ou produit scalaire
$\boldsymbol{ax} + b$	une forme affine
$\boldsymbol{ax} + b = 0$	l'équation d'un hyperplan affine
$\boldsymbol{ax} + b \geq 0$	la définition d'un demi-espace affine (supporté par un hyperplan)
$\mathbf{A}$	une matrice
${}^{t}\mathbf{A}$	la matrice transposée de $\mathbf{A}$ (également notée $\mathbf{T}$)
m	le nombre de lignes ou de contraintes
n	le nombre de colonnes ou de variables (dimension de l'espace)
i	un indice de ligne
j	un indice de colonne
I	l'ensemble des indices des lignes
J	l'ensemble des indices des colonnes
C	une contrainte d'inégalité (ou clause pour le problème SAT)
S	un système de contraintes $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$
P	un polyèdre
Sol(S)	l'ensemble des solutions du système de contraintes S
$\mathrm{E}_F(\mathrm{S}, x)$	le système obtenu par élimination de Fourier d'une variable x de S
Proj(P, x)	la projection de l'ensemble P parallèlement à l'axe des x
$\mathrm{proj}_j(\mathrm{P})$	la j-ième projection de P (ou projection sur l'axe des x_j)
$\Im$	une base d'approximation pour le calcul des intervalles
M	la limite d'approximation sur les entiers de la base finie $\Im_M$
apx(E)	l'approximation cartésienne, sur la base $\Im$, d'un ensemble $E \subset \mathbb{R}^n$ (plus petit produit cartésien d'intervalles de $\Im$ contenant E)
red_C	l'opérateur de réduction associé à une contrainte C
red_S	l'opérateur de réduction associé au système S.

Notations infixées propres à Prolog IV :

`plus`	opération d'addition pour le solveur d'intervalles
`pluslin`	opération d'addition pour le solveur linéaire
`ge`	relation d'inégalité large pour le solveur d'intervalles
`gelin`	relation d'inégalité large pour le solveur linéaire.

3 Plan de la thèse

Cette thèse s'articule en trois parties.

I

La première partie comprend trois chapitres généraux (numérotés de 2 à 4) qui établissent les principaux résultats théoriques. Ceux-ci sont relatifs à la résolution d'une classe particulière de systèmes linéaires, notamment diophantiens. Ils montrent que ces systèmes peuvent être efficacement résolus par des techniques formelles ou approchées.

Chapitre II
Aux origines du calcul des inégalités
Le procédé d'élimination de Fourier

Ce chapitre s'ouvre sur un rappel, tant historique que pratique, de l'analyse de Fourier pour la résolution formelle des systèmes linéaires. Nous l'avons préférée — pour sa simplicité, sa généralité et sa profondeur — à une évocation des algorithmes de Gauss et du simplexe que l'on retrouve dans les implantations des solveurs linéaires.

Cette méthode, qui élimine des variables en inférant de nouvelles inégalités, qui est applicable à des équations comme à des inéquations simultanées, et de laquelle on tire toute la théorie de la dualité, nous servira à la fois de procédure de décision et de technique de preuve.
Quant à sa complexité, que l'on juge rédhibitoire, elle tombera, pour deux des sous-cas qui nous intéressent, à linéaire et cubique — décrochant ainsi la meilleure performance dans ces catégories.

Nous exploiterons enfin une remarque donnant une condition suffisante pour que l'élimination d'une variable entière soit complète et produise un système linéaire équivalent.

Chapitre III
Des familles de polyèdres ayant un plus petit élément
Propriétés remarquables des inégalités monotones

Ce chapitre introduit la classe des systèmes linéaires monotones, objets centraux de toute cette étude, et pour lesquels on observe une conjonction remarquable entre propriétés syntaxiques et propriétés géométriques.

Syntaxiquement, ce sont des systèmes d'inégalités dont la matrice a au plus un coefficient > 0 par ligne (propriété *m*).
Géométriquement, ces systèmes définissent des polyèdres (éventuellement, en nombres entiers) qui ont une structure de demi-treillis.

Quand ils sont bornés, ces polyèdres ont un sommet distingué, plus petit (ou plus grand) que tous les autres. La détermination de la satisfaisabilité d'un système monotone $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ équivaut ainsi, dans tout translaté du cône positif $\mathbb{R}^{+n}$, à la vérification de l'existence d'un minimum global.

Ce plus petit sommet se trouve être à coordonnées entières pour tout une sous-classe de systèmes monotones à coefficients entiers, où les éléments > 0 de la matrice valent 1 (propriété 1-*m*).
L'existence de solutions réelles étant équivalente à l'existence de solutions entières, ces systèmes 1-monotones forment une classe polynômiale de problèmes en nombres entiers.

Nous montrons plus généralement que la résolution des systèmes monotones diophantiens dont le minimum se situe, lorsqu'il existe, entre des bornes connues, est également polynômiale.

Chapitre IV
Résolution approchée des systèmes monotones
Techniques de réduction d'intervalles

Ce chapitre présente un exposé — limité au linéaire — d'une méthode approchée pouvant servir d'alternative dans la résolution générale de systèmes de contraintes. Les calculs sur les valeurs sont remplacés par des calculs sur des intervalles de valeurs. S'appuyant sur un maillage fini de l'espace, ce calcul des intervalles détermine ici itérativement le plus petit encadrement rectangulaire commun aux solutions de chacune des inégalités du système.

A partir des formules de réductions d'intervalles applicables aux inégalités monotones, nous démontrons qu'en jouant sur la finesse ou l'étendue du maillage, nous pouvons :

- bien approcher le minimum global d'un système monotone sur les réels (*théorème d'approximation*)
- calculer de façon parfaite le minimum global d'un système 1-monotone.

La complétude de la résolution par les intervalles s'ensuit (sous réserve de maillage adéquat), pour les systèmes :

- 1-monotones et bornés, à variables réelles ou entières (*théorème de cohérence globale*)
- monotones et bornés, à variables entières.

II

Cette seconde partie réexamine, à la lumière des résultats de la première, deux problèmes fondamentaux d'algorithmique : la satisfaction de contraintes, pour le calcul booléen (chapitre 5) et l'optimisation sous contraintes, pour le calcul de plus courts chemins (chapitre 6).

Chapitre V
De Fourier à Davis et Putnam
Satisfaisabilité en logique propositionnelle

Nous réabordons ici, en le transposant dans le linéaire, le fameux problème de la satisfaisabilité d'un ensemble de clauses sur $\{0, 1\}^n$.

Chaque clause est exprimée par le biais d'une inégalité en variables bivalentes, dont les coefficients matriciels sont pris dans {–1, 0, +1}. Sous cette forme, les inégalités monotones s'identifient aux clauses de Horn. Leur résolution est effectivement polynômiale (et même linéaire).

On observe que la procédure de résolution unitaire positive découle directement de l'analyse de Fourier. De plus, les inférences produites par élimination de variables concordent avec celles obtenues par élimination de valeurs, avec les intervalles.

En remontant à l'algorithme de Davis et Putnam qui utilise la résolution unitaire comme technique de filtrage pour les systèmes de clauses plus généraux, on redécouvre d'autres similitudes partagées avec l'algorithme de Fourier. On évoque, pour conclure ce chapitre, une technique susceptible d'abaisser la complexité théorique d'un facteur exponentiel (et qui, accessoirement, redonne aux systèmes monotones un rôle de premier plan) : la relaxation par les clauses de Horn.

Chapitre VI
Cheminements et ordonnancements
Optimisation combinatoire sur les graphes

Où l'on montre que ce que l'on calcule classiquement comme plus courtes distances sur un graphe sont les coordonnées du plus grand élément d'un problème dual de tensions — où les contraintes d'inégalité sur les arcs, entre les potentiels définis aux noeuds, viennent remplacer les équations de conservation aux noeuds, entre les courants définis par les cheminements sur les arcs.

La matrice d'incidence et son opposée définissant des systèmes 1-monotones, ce passage au dual a pour avantages :

- de donner, avec Fourier, un algorithme complet et polynômial (cubique)
- de calculer, avec les intervalles, les projections exactes des solutions entières (l'ensemble des solutions étant un treillis, avec deux sommets distingués).

Un problème analogue dans le continu est celui de l'ordonnancement simple. Tant que les bornes restent entières, il n'y a pas de différence au niveau des calculs. La résolution par les intervalles est complète si les domaines sont bornés.

III

La troisième partie présente enfin une application plus technique (chapitre 7), où le calcul précédent s'inscrit au cœur d'une procédure de vérification automatique — hautement combinatoire — régie par un jeu de règles logiques.

Chapitre VII
Vérification de propriétés d'automates synchronisés
Analyse de l'accessibilité dans les Réseaux de Petri

En suivant une démarche symbolique en vérification de modèles, nous présentons une formulation déclarative du calcul des configurations accessibles, à l'aide de règles exprimant le non-déterminisme des transitions, et de contraintes raisonnant sur des ensembles de configurations.

Dans le cas de base, les ensembles manipulés sont des pavés de points entiers qui ont tous un plus petit élément. Les transitions du modèle expriment à la fois des intersections de pavés et les translations de ces intersections. Le test d'inclusion entre pavés sert à reconnaître les régions déjà explorées. Toutes ces opérations (qui se généralisent aux polyèdres de points entiers solutions d'un système de contraintes de tensions) peuvent être indifféremment réalisées (pour le cas de base) à l'aide d'un solveur linéaire ou d'intervalles.

L'utilisation de techniques d'accélération nous fait cependant sortir du cadre strict que nous nous sommes fixés, lequel nous garantit la complétude de la résolution de contraintes en nombres entiers. Nous montrons comment compenser cette faiblesse par une vérification *a posteriori* qui nous replace dans un contexte valide de preuve.

Chapitre VIII
Conclusion
Bilan, références, perspectives.

Après un dernier exemple en calcul numérique, nous dressons un tableau récapitulatif des déclinaisons faites des propriétés des systèmes linéaires monotones.

CHAPITRE II

Aux origines du calcul des inégalités

Le procédé d'élimination de Fourier

> «Les signes algébriques sont employés les uns à *dénommer* des quantités ; d'autres à indiquer des *affectations* de ces quantités ou des opérations qu'elles doivent subir dans le calcul ; d'autres enfin à *affirmer* certains rapports entre elles, tels que les signes de proportion, d'égalité et d'inégalité, celui de congruence adopté par certains géomètres dans la théorie des nombres, et d'autres semblables qu'on pourrait proposer. Le petit nombre des figures de cette dernière nature, analogues aux *verbes* des langues vulgaires, montre assez que, s'il faut regarder avec Condillac l'algèbre comme la plus parfaite des langues, elle est loin d'être la plus riche.»
>
> A. COURNOT

On assimile souvent l'Algèbre au maniement des équations, tant le signe d'égalité se prête avec facilité au mécanisme du calcul. Comme le notait Cournot, «encore maintenant, on ne se sert guère des signes d'inégalité que pour énoncer des conditions auxiliaires ; il est rare qu'on les emploie directement comme moyen de calcul.»

Fourier fut le premier à saisir l'importance pratique du maniement des inégalités. A la suite de ses recherches sur la Théorie de la chaleur, il mit au point une méthode de résolution directe des systèmes d'inéquations. Il fit connaître les principes ce *Calcul des conditions d'inégalités* dans une série de mémoires d'analyse, lus à l'Institut au cours des années 1823-1824. Fourier pensait pouvoir conclure avant sa mort son grand ouvrage sur la Théorie des équations par une étude détaillée sur ce sujet.

«Une question est, en général, déterminée lorsque le nombre des équations qui expriment toutes les conditions proposées est égal au nombre des inconnues. Dans la théorie dont il s'agit, les conditions ne sont pas exprimées par des équations ; c'est-à-dire qu'au lieu d'égaler, à une constante ou à zéro, une certaine fonction des inconnues, on indique au moyen des signes < ou > que cette fonction est plus grande ou moindre que la constante. C'est ce qui constitue une inégalité.»

L'objet du calcul est le suivant :

«Il s'agit de trouver des valeurs des inconnues qui, étant substituées simultanément, satisfont à toutes les conditions proposées, soit que ces conditions consistent seulement dans certaines inégalités, soit qu'elles comprennent aussi des équations. Une question de cette espèce admet une infinité de solutions ; elle est indéterminée.»

«Il est évident que les conditions proposées pourraient être telles que la question n'admît aucune solution possible. Dans ce cas, le calcul développe l'opposition réciproque des conditions, et montre l'impossibilité d'y satisfaire. Ainsi, la méthode a pour objet :

1° de reconnaître si la question peut être résolue ;

2° de trouver dans ce cas toutes les solutions qu'elle admet ;

3° de mesurer par un nombre l'étendue[1] propre à la question.»

Devançant de plus d'un siècle l'invention de la Programmation mathématique, le Secrétaire perpétuel de l'Académie des Sciences précise :

«Il arrive souvent aussi, dans ce genre de recherches, que l'objet principal n'est pas de trouver toutes les solutions, mais d'en reconnaître une ou plusieurs limites. Sous ce point de vue, la question n'est pas indéterminée ; et il en est de même de celle qui consiste à mesurer l'étendue ; mais ces questions dépendent de la même analyse.»

Nous nous intéresserons principalement avec Fourier à la première question, à savoir la détermination de la satisfaisabilité d'un système d'inéquations — à laquelle se ramène l'optimisation.

Quoique la méthode de Fourier soit plus générale, nous laisserons de côté les inégalités strictes au profit des inégalités larges $\leq$ et $\geq$. Une égalité $f = g$ pouvant être exprimée par une double inégalité $f \leq g$ et $g \leq f$, nous n'aurons besoin que d'un seul signe, $\leq$ ou $\geq$, pour uniformiser l'écriture des systèmes linéaires standard.

Ce chapitre rappelle la règle établie par Fourier pour un calcul «qui offre des applications variées à la Géométrie, à l'Analyse algébrique, à la Mécanique et à la Théorie des probabilités». Cette règle joue sur des combinaisons linéaires de contraintes pour éliminer des variables. Elle débouche sur un théorème de satisfaisabilité dont l'interprétation géométrique est immédiate : un polyèdre est vide dès lors que l'une de ses projections est vide.
On évoque ensuite l'extension possible aux espaces discrets de ce principe d'élimination. On donne enfin un aperçu de ses prolongements actuels en tant que méthode formelle.

1. Volume ou lieu solide des solutions, mesuré par une intégrale multiple.

1 L'Analyse des inégalités

Considérons un problème dont les conditions s'expriment par une série d'inégalités de la forme :

$$(1) \quad f(y, z, \ldots) \leq x \quad \text{ou} \quad (2) \quad x \leq F(y, z, \ldots)$$

Ces inégalités font apparaître qu'une même inconnue, x, est tantôt plus grande, tantôt plus petite qu'une fonction arbitraire des autres inconnues (y, z, …). De leur comparaison vont découler un certain nombre de nouvelles inégalités dans lesquelles x ne figurera plus :

$$(3) \quad f(y, z, \ldots) \leq F(y, z, \ldots)$$

Si l'on part de m inégalités, dont m' de forme (1) et $m - m'$ de forme (2), on aura, après élimination de x, $m'.(m - m')$ inégalités résultantes (3) en y, z, … — soit au pire $m^2/4$ inéquations.

«Elimination singulière», commente Cournot, «qui fait croître le nombre de conditions en diminuant celui des variables».

Conséquences logiques des premières, ces nouvelles inégalités sont autant de conditions nécessaires à la satisfaisabilité du système d'origine. Fourier note que, dans leur ensemble, ces conditions sont suffisantes pour les systèmes linéaires.

2 Elimination de Fourier dans le cas linéaire

Plaçons-nous dans le cas d'un système S, non vide, de m inéquations linéaires à n inconnues ($m, n \geq 1$). Sous forme matricielle, le système s'écrira : $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$.

2.1 Notations

On désignera par $\mathrm{I} = \{1, 2, \ldots, m\}$ les indices des lignes — correspondant aux contraintes, et par $\mathrm{J} = \{0, 1, 2, \ldots, n-1\}$ une permutation des indices des colonnes — correspondant aux variables.

Sans perdre de généralité, on supposera tous les coefficients de $\mathbf{A}$ et $\boldsymbol{b}$ entiers.
On décomposera $\mathbf{A}$ additivement en une différence $\mathbf{A}^+ - \mathbf{A}^-$ de deux matrices $m \times n$ positives donnant les valeurs absolues a_{ij} et les signes des coefficients de A.
Suivant cette convention, chaque contrainte C_i du système s'écrira :

$$\sum_{j \in \mathrm{J}_i^+} a_{ij} x_j - \sum_{j \in \mathrm{J}_i^-} a_{ij} x_i \geq b_i$$

avec $\mathrm{J}_i^+ \cap \mathrm{J}_i^- = \varnothing$, $\mathrm{J}_i^+ \cup \mathrm{J}_i^- \subset \mathrm{J}$ et $a_{ij} > 0 \ \forall j$.

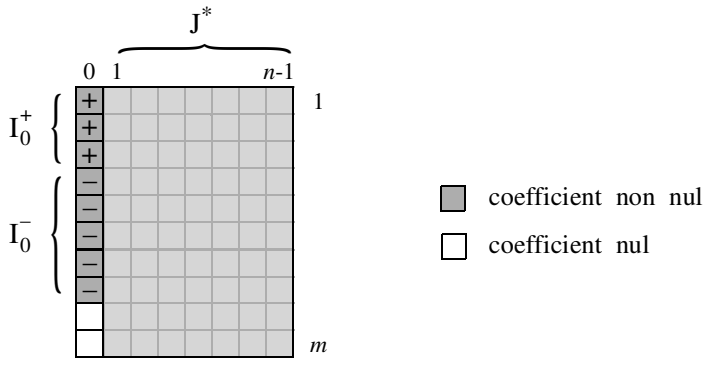
La variable à éliminer pouvant être l'une quelconque des n variables du système, on supposera que la permutation J la place en première colonne (x_0).

On partitionnera le système en trois classes d'inégalités :

- celles où x_0 apparaît supérieur à une forme affine des autres variables (I_0^+)
- celles où x_0 apparaît inférieur à une forme affine des autres variables (I_0^-)
- celles où x_0 n'apparaît pas.

Les inégalités étant toutes orientées dans le même sens, la partition s'effectuera suivant le signe des coefficients présents dans la première colonne de A.
Une inégalité C_i sera dans la première classe si x_0 a pour coefficient $a_{i0} > 0$; dans la seconde si x_0 a pour coefficient $-a_{i0} < 0$; dans la troisième si $a_{i0} = 0$.

Formellement, on aura : $I_0^+ = \{ i \in I \mid 0 \in J_i^+ \}$, $I_0^- = \{ i \in I \mid 0 \in J_i^- \}$
avec $I_0^+ \cap I_0^- = \varnothing$ et $I_0^+ \cup I_0^- \subset I$. On notera $J^* = J - \{0\} = \{1, 2, \ldots, n-1\}$.
A une permutation des lignes près, la matrice du système aura donc la forme :



2.2 Elimination d'une variable

Définition Deux contraintes C_k et C_l de S forment une paire d'*inégalités complémentaires* vis à vis de x_0 si $k \in I_0^-$ et $l \in I_0^+$.

Lorsque tel est le cas, on a :

$$\begin{array}{l|l} C_k : \sum_{j \in J_k^{+*}} a_{kj}x_j - a_{k0}x_0 - \sum_{j \in J_k^{-*}} a_{kj}x_j \geq b_k & \times a_{l0} \\ C_l : \sum_{j \in J_l^{+*}} a_{lj}x_j + a_{l0}x_0 - \sum_{j \in J_l^{-*}} a_{lj}x_j \geq b_l & \times a_{k0} \end{array}$$

On élimine x_0 en multipliant les deux membres de C_k par a_{l0}, les deux membres de C_l par a_{k0}, puis en sommant les inégalités produites. On obtient :

$$\sum_{j \in J_k^{+*}} a_{l0}a_{kj}x_j + \sum_{j \in J_l^{+*}} a_{k0}a_{lj}x_j - \left(\sum_{j \in J_k^{-*}} a_{l0}a_{kj}x_j + \sum_{j \in J_l^{-*}} a_{k0}a_{lj}x_j \right) \geq a_{l0}b_k + a_{k0}b_l$$

Cette troisième inégalité est également linéaire (et à coefficients entiers). Elle peut être vue comme une *combinaison linéaire à coefficients positifs* des deux premières, à savoir : $a_{l0}C_k + a_{k0}C_l$.

Définition Le produit de l'*élimination de Fourier* de la variable x_0 dans le système S est le système :

$$\mathrm{E}_F(\mathrm{S}, x_0) = \bigcup_{i \notin \mathrm{I}_0^- \cup \mathrm{I}_0^+} C_i \quad \cup \quad \bigcup_{(k,l) \in \mathrm{I}_0^- \times \mathrm{I}_0^+} a_{l0}C_k + a_{k0}C_l$$

2.3 Propriété de l'élimination

D'un point de vue syntaxique, l'élimination de Fourier transforme un système d'inégalités linéaires $\mathrm{S} = \bigcup_{i \in \mathrm{I}} C_i$ en un autre système linéaire, possédant une inconnue en moins. Tout l'intérêt de ce dernier est d'être équivalent à S du point de vue de la satisfaisabilité.

Définition On notera Sol(S) l'*ensemble des solutions* d'un système S.

Sémantiquement — ou géométriquement parlant, l'ensemble des solutions d'un système linéaire $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$, de taille $m \times n$, est un polyèdre convexe fermé $\mathrm{P} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$.

Dire que S est satisfaisable revient à dire que le polyèdre Sol(S) est non vide.

Le produit S' de l'élimination de Fourier d'une variable x dans S va décrire, dans $\mathbb{R}^{n-1}$, un nouveau polyèdre : $\mathrm{P}' = \{\boldsymbol{x}' \in \mathbb{R}^{n-1} \mid \mathbf{A}'\boldsymbol{x}' \geq \boldsymbol{b}'\}$.

Par une permutation des axes, on peut toujours se ramener, comme précédemment, au cas où la variable à éliminer, x, correspond à la première composante x_0 de $\boldsymbol{x}$. On posera donc, pour alléger l'écriture : $\boldsymbol{x} = (x, \boldsymbol{x}')$ avec $\boldsymbol{x}' = (x_1, \ldots, x_{n-1})$.

Théorème II.1 *Soit* S *un système linéaire non vide dont x est l'une des inconnues. Soit* $\mathrm{S}' = \mathrm{E}_F(\mathrm{S}, x)$. *Alors, on a l'équivalence :*

$$\boldsymbol{x}' \in \mathrm{Sol}(\mathrm{S}') \quad \Leftrightarrow \quad \exists\, x_0 \in \mathbb{R} \text{ t.q. } (x_0, \boldsymbol{x}') \in \mathrm{Sol}(\mathrm{S})$$

Démonstration Dans un sens, il est clair que si $\boldsymbol{x} = (x_0, \boldsymbol{x}')$ est solution de S, $\boldsymbol{x}'$ est solution de S' : toutes les inégalités de S étant vérifiées par $\boldsymbol{x}$, celles de S' qui en découlent sont nécessairement vérifiées par $\boldsymbol{x}'$.

Dans l'autre sens, si $\boldsymbol{x}'$ est solution de S', $\boldsymbol{x}'$ est solution du sous-système d'inégalités $\bigcup_{i \notin \mathrm{I}_0^- \cup \mathrm{I}_0^+} C_i$, commun à S et S'. Il reste à prouver que les inégalités $\bigcup_{i \in \mathrm{I}_0^- \cup \mathrm{I}_0^+} C_i$ de S où entre la variable x_0 sont également vérifiées.

Rappelons que la condition exprimée une inégalité reste inchangée si l'on multiplie les deux membres de cette inégalité par un même scalaire strictement positif. On peut ainsi diviser les inégalités $a_{l0}C_k + a_{k0}C_l$ de S' par le produit $a_{k0}a_{l0} > 0$. Ceci revient à remplacer les combinaisons entières de $E_F(S, x)$ par des combinaisons à coefficients rationnels : $\frac{C_k}{a_{k0}} + \frac{C_l}{a_{l0}}$.

Posons $F_{k0}(\boldsymbol{x'}) = \frac{1}{a_{k0}}\left(\sum_{j \in J_k^{+*}} a_{kj}x_j - \sum_{j \in J_k^{-*}} a_{kj}x_j - b_k \right)$

et $f_{l0}(\boldsymbol{x'}) = \frac{1}{a_{l0}}\left(b_l + \sum_{j \in J_l^{-*}} a_{lj}x_j - \sum_{j \in J_l^{+*}} a_{lj}x_j \right)$.

L'inégalité $\frac{C_k}{a_{k0}}$, équivalente à C_k, s'écrit : $x \le F_{k0}(\boldsymbol{x'})$.

L'inégalité $\frac{C_l}{a_{l0}}$, équivalente à C_l, s'écrit : $f_{l0}(\boldsymbol{x'}) \le x$.

L'inégalité $\frac{C_k}{a_{k0}} + \frac{C_l}{a_{l0}}$, équivalente à $a_{l0}C_k + a_{k0}C_l$, s'écrit : $f_{l0}(\boldsymbol{x'}) \le F_{k0}(\boldsymbol{x'})$.

Pour toute paire (C_k, C_l) d'inégalités complémentaires, $\boldsymbol{x'}$ vérifie ainsi :

$$f_{l0}(\boldsymbol{x'}) \le F_{k0}(\boldsymbol{x'}).$$

Cette condition étant vraie pour tout $k \in I_0^-$ et tout $l \in I_0^+$, on a donc, en prenant inf (resp. sup) dans $\mathbb{R} \cup \{+\infty\}$ (resp. $\{-\infty\}$) [1] :

$$\sup_{l \in I_0^+} f_{l0}(\boldsymbol{x'}) \le \inf_{k \in I_0^-} F_{k0}(\boldsymbol{x'}).$$

Lorsque $\boldsymbol{x'}$ désigne une solution particulière de S', ces deux bornes délimitent un intervalle non vide de $\mathbb{R}$. Il suffit de prendre x_0 dans cet intervalle pour vérifier :

$$f_{l0}(\boldsymbol{x'}) \le x_0 \ \ \forall l \in I_0^+ \ \text{ et } x_0 \le F_{k0}(\boldsymbol{x'}) \ \ \forall k \in I_0^-.$$

Les valeurs de x_0 et $\boldsymbol{x'}$ satisfont alors à toutes les contraintes de (P). ■

Des mêmes hypothèses, on déduit immédiatement le corollaire :

Corollaire *Le système* S *est satisfaisable si et seulement si le système* $S' = E_F(S, x)$ *l'est.*

1. On convient que le sup. (resp. l'inf.) d'un ensemble vide vaut $-\infty$ (resp. $+\infty$). Ainsi, lorsqu'il n'existe pas de paire complémentaire pour x, si la classe d'inégalités I^+(resp. I^-) est vide, on ne retient pour x_0 que le minimum des bornes par excès (resp. le maximum des bornes par défaut). Si ces deux classes sont vides, cela veut dire que la variable éliminée n'a que des coefficients nuls dans S : toute affectation est alors bonne pour x_0.

Remarque Si la variable x_0 n'est affectée que de coefficients positifs (resp. négatifs) dans S, et si S est satisfaisable, alors l'ensemble des solutions de S est non borné dans la direction des x_0 croissants (resp. décroissants).

Le système $E_F(S, x_0)$ s'obtient dans ce cas sans ajout de nouvelle inégalité, en retranchant simplement les lignes pour lesquelles $a_{i0} > 0$. A toute solution $\boldsymbol{x'}$ de $E_F(S, x_0)$, on peut associer une solution de S en prenant x_0 dans la demi-droite $[\max_{l \in \Gamma_0^+} f_{l0}(\boldsymbol{x'}) , +\infty[$ (resp. $]-\infty, \min_{k \in \Gamma_0^-} F_{k0}(\boldsymbol{x'})]$).

Si la variable x_0 n'est affectée que de coefficients nuls dans S, $E_F(S, x_0) = S$.

2.4 Interprétation géométrique

Définition Soit P un sous-ensemble de $\mathbb{R}^n$ et x le nom du premier axe.

La *projection de* P *parallèlement à* x est le sous-ensemble de $\mathbb{R}^{n-1}$ défini par :

$$(x_1, \ldots, x_{n-1}) \in \text{Proj}(P, x) \quad \Leftrightarrow \quad \exists\, x_0 \in \mathbb{R} \text{ t.q. } (x_0, x_1, \ldots, x_{n-1}) \in P$$

La projection de P suivant le j-ième axe se définit de façon similaire.

Propriété *Soient* S *un système linéaire,* x *une variable de* S *et* $S' = E_F(S, x)$. *Posons* $P = \text{Sol}(S)$ *et* $P' = \text{Sol}(S')$. *Alors,* $P' = \text{Proj}(P, x)$.

De façon générale, l'élimination de Fourier de la variable x_j du système S correspond à la projection du polyèdre Sol(S) parallèlement à l'axe de $\mathbb{R}^n$ numéroté par j.

L'élimination de Fourier montre, plus précisément :

1. que la projection d'un polyèdre convexe fermé P est un polyèdre convexe fermé P'
2. que ce polyèdre P' est défini par un ensemble de contraintes qui sont toutes combinaisons linéaires positives des contraintes de P.

2.5 Inégalités redondantes

L'élimination d'une variable (ou quantificateur) entre m inégalités est un procédé de nature éminemment combinatoire. Comme le souligne Fourier lui-même, les inégalités engendrées sont souvent redondantes.

«Il arrive presque toujours qu'un assez grand nombre de ces nouvelles inégalités subsistent évidemment, et qu'il est inutile de les écrire. Ces réductions se présentent d'elles-mêmes, et elles simplifient beaucoup le calcul.»

Parmi les inégalités produites, on notera que certaines sont sans variables, donc de la forme $0 \geq h$ avec h scalaire. Celles-ci sont trivialement redondantes si elles sont vraies (c'est-à-dire, si h est négatif). Si, en revanche, l'une d'elles est fausse ($h > 0$), le système est contradictoire et n'a pas de solution.

2.6 Procédure de décision

En procédant par éliminations successives, suivant la règle donnée par Fourier, on obtient algorithme complet pour de décider de la satisfaisabilité d'un système linéaire S. En effet, l'élimination de Fourier engendrant à son tour un système linéaire, on pourra, tant qu'il reste des variables, effectuer une nouvelle élimination sur le système produit.

On conviendra que $E_F(S, x) = S$ lorsque x ne figure pas dans le système S — situation analogue à celle où x n'a que des coefficients nuls. En particulier, si S est vide — donc forcément satisfaisable — $E_F(S, x) = \emptyset$.

Si n est le nombre d'inconnues du système initial $S = S_n$, on définira, en prenant pour ordre d'élimination des variables celui des indices de J, la suite $(S_{n-1-j})_{\,0 \le j \le n-1}$ des systèmes linéaires :

$$\begin{aligned} S_{n-1} &= E_F(S_n, x_0) \\ &\dots \\ S_{n-j-1} &= E_F(S_{n-j}, x_j) \\ &\dots \\ S_0 &= E_F(S_1, x_{n-1}) \end{aligned}$$

On débouchera ainsi, en au plus n étapes, sur un système d'inégalités S_0, équivalent à S du point de vue de la satisfaisabilité, et dans lequel n'entre plus aucune inconnue.

La décision résultera de l'un des cas suivants :

1. soit toutes les inégalités de S_0 (s'il en existe) sont trivialement vraies : en ce cas, S_0 est équivalent au système vide $\emptyset$ et S est satisfaisable.
2. soit S_0 contient une contradiction : $0 \ge h$ où h est une constante ≥ 1 : en ce cas, S_0 est équivalent au système $\{0 \ge 1\}$ et S est insatisfaisable.

Définition Soit S_n un système linéaire à n variables et $(x_0, x_1, \dots, x_{n-1})$ un ordre sur ces variables. La *procédure d'élimination de Fourier* correspond, pour cet ordre, à la construction de la suite de systèmes $(S_{n-1-j})_{\,0 \le j \le n-1}$ définie, pour j croissant de 0 à n–1, par : $S_{n-j-1} = E_F(S_{n-j}, x_j)$.

L'ultime élément de cette suite, S_0, sera noté $E_F(S_n, (x_0, x_1, \dots, x_{n-1}))$, ou encore $E_F(S_n)$ lorsqu'aucune ambiguïté n'est à craindre.

Remarque L'avant dernier élément de cette suite, $S_1 = E_F(S_n, (x_0, x_1, \dots, x_{n-2}))$, est un système linéaire (éventuellement vide de contraintes) à une seule inconnue : x_{n-1}. Le polyèdre $P_1 = \mathrm{Sol}(S_1)$ est un intervalle fermé[1] de $\mathbb{R}$ qui donne la projection du polyèdre $P_n = \mathrm{Sol}(S_n)$ sur l'axe des x_{n-1} : $P_1 = \mathrm{proj}_{n-1}(P_n)$.

1. *I.e.*: intervalle vide, segment, demi-droite fermée ou droite réelle.

L'arrêt avant son terme de la procédure permet de n'éliminer qu'une partie des variables du système et d'en donner une expression équivalente, S_{n-j}, en fonction des inconnues «principales» (celles d'indices $\geq j$ pour l'ordre ici choisi). Ce besoin apparaît fréquemment en programmation par contraintes pour la simplification des systèmes qui, du fait de leur construction incrémentale, contiennent de nombreuses inconnues superfétatoires.

2.7 Interprétation logique

Définition Soient S un système d'inégalités linéaires sur $\mathbb{R}^n$, $\boldsymbol{c}$ un vecteur de $\mathbb{R}^n$ et d un scalaire. On dira que l'inégalité linéaire $\boldsymbol{cx} \geq d$ est une *conséquence logique* de S si cette inégalité est vérifiée par toute solution $\boldsymbol{x}$ de S.

Toute combinaison linéaire à coefficients positifs d'inégalités de S donne une inégalité qui est une conséquence de S. On peut dire que Fourier utilise ces combinaisons comme règle d'inférence. De fait, nous montrerons au paragraphe sur la dualité que toute inégalité conséquence de S peut s'exprimer comme une combinaison linéaire positive d'inégalités de S.

Notons que dans le cas où Sol(S) est vide, n'importe quelle inégalité peut être vue comme une conséquence de S, notamment : $0 \geq 1$.

Définition On dira qu'un système S est *contradictoire* si $0 \geq 1$ est conséquence de S.

Le résultat suivant (lemme de Farkas, 1893) fait le lien entre les notions de résolution algébrique et de cohérence logique. Sa preuve découle de Fourier, comme l'a montré Kuhn (1956).

Théorème II.2 *Un système linéaire est soluble (ou satisfaisable) si et seulement si il est cohérent (ou non contradictoire).*

Démonstration Que S soit non contradictoire (et n'ait pas pour conséquence $0 \geq 1$) est clairement une condition nécessaire pour que S ait des solutions.

Réciproquement, quand S n'a pas de solution, la procédure d'élimination de Fourier exhibe une contradiction $C : 0 \geq h$ avec h constante ≥ 1. Comme toutes les inégalités de $E_F(S)$, C est obtenue par combinaisons linéaires positives d'inégalités de S. Donc $\frac{1}{h}C$, soit $0 \geq 1$, peut l'être également. ■

2.8 Complexité

Le nombre d'inégalités pouvant croître de façon exponentielle[1], la méthode de Fourier n'est pas polynômiale, ce qui constitue son principal désavantage par rapport aux méthodes d'optimisation de la Programmation linéaire.

1. Et même doublement exponentielle (voir paragraphe 6.2).

Impraticable sur des problèmes de grande taille, elle est pour cette raison peu utilisée, mais reste essentielle à la à simplification des systèmes linéaires.

Depuis sa rédécouverte par Motzkin en 1936, de nombreux efforts ont porté sur l'amélioration de son efficacité et la suppression des redondances — une nécessité cruciale pour lutter contre l'inflation du nombre d'inégalités. Les travaux de Lassez et d'Imbert dans le domaine de la Programmation en logique avec contraintes fournissent d'amples références à ce sujet.

Nous donnerons, dans les chapitres applicatifs, deux exemples de classes de problèmes où la croissance de la taille des systèmes peut être bridée *a priori*. Cette connaissance assure une efficacité optimale à l'élimination — la complexité de l'algorithme spécialisé chutant d'exponentielle à quadratique, voire à linéaire, sur ces cas très particuliers.

3 Dualité, optimisation et consécution

3.1 Théorème de la dualité

Un problème d'optimisation linéaire, ou «programme linéaire» selon la terminologie traditionnelle de la Recherche Opérationnelle, consiste à minimiser (ou maximiser) une forme linéaire sous des contraintes linéaires :

$$\min\{\boldsymbol{cx} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\} \tag{1}$$

L'analyse de Fourier permet de retrouver très simplement quelques résultats fondamentaux de la programmation linéaire.

Introduisons tout d'abord une variable z pour intégrer la fonction de coût comme contrainte. Soit S le système linéaire formé des m+1 inégalités $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ et $z - \boldsymbol{cx} \geq 0$. Sol(S) est le polyèdre $P = \{(z, \boldsymbol{x}) \in \mathbb{R}^{n+1} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}, z - \boldsymbol{cx} \geq 0\}$. La projection de P sur l'axe des z est un intervalle fermé.

Trois cas se présentent :

1. $\text{proj}_z(P)$ *est vide*
   S est insatisfaisable :
   $\inf\{\boldsymbol{cx} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\} = +\infty$
2. $\text{proj}_z(P)$ *est non vide et non borné inférieurement*
   S est satisfaisable mais $\{\boldsymbol{cx} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$ n'a pas de minimum :
   $\inf\{\boldsymbol{cx} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\} = -\infty$
3. $\text{proj}_z(P)$ *est non vide et admet une borne inférieure finie* z^*
   S est satisfaisable et le minimum de $\{\boldsymbol{cx} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$ existe :
   $\min\{\boldsymbol{cx} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\} = z^*$

Dans ce dernier cas, la méthode de Fourier produit au niveau de $E_F(S, \boldsymbol{x})$ l'inégalité C^* : $z \geq z^*$ (telle quelle ou multipliée par une constante positive).

C^* peut donc être obtenue comme une combinaison linéaire à coefficients positifs des inégalités $C_i : \boldsymbol{a}_i \boldsymbol{x} \geq b_i$ et $C_{m+1} : z - \boldsymbol{cx} \geq 0$ de S :

$$C^* = y_1 C_1 + y_2 C_2 + \ldots + y_m C_m + 1 C_{m+1}$$

(Le coefficient de la variable z étant 1 dans C^*, on a nécessairement $y_{m+1} = 1$.) Par identification des coefficients constants et annulation des coefficients des x_j, on déduit que les multiplicateurs y_i des contraintes C_i sont solution du système linéaire constitué des $n+1+m$ contraintes suivantes :

$$\sum_{i=1}^{m} y_i b_i = z^*, \quad \sum_{i=1}^{m} y_i a_{ij} = c_j \text{ pour } j = 1, \ldots, n \text{ et } y_i \geq 0 \text{ pour } i = 1, \ldots, m$$

soit : $\boldsymbol{y} \geq 0$, $\boldsymbol{yb} = z^*$ et ${}^t\mathbf{A}\boldsymbol{y} = \boldsymbol{c}$.

D'autre part, en multipliant à gauche par $\boldsymbol{y} \geq 0$ chacun des membres de l'inégalité $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$, il vient : $\boldsymbol{y}\mathbf{A}\boldsymbol{x} \geq \boldsymbol{yb}$. Par conséquent, si $\boldsymbol{y}\mathbf{A} = \boldsymbol{c}$ on a : $\boldsymbol{cx} \geq \boldsymbol{yb}$. Ainsi, le coût d'une solution en $\boldsymbol{x}$ du problème «primal» $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ fournit un majorant du coût d'une solution en $\boldsymbol{y}$ du problème «dual» $\boldsymbol{y}\mathbf{A} = \boldsymbol{c}$, $\boldsymbol{y} \geq 0$:

$$\max\{\boldsymbol{yb} \mid \boldsymbol{y}\mathbf{A} = \boldsymbol{c}, \boldsymbol{y} \geq 0\} \tag{2}$$

Lorsqu'un optimum existe, les égalités précédentes démontrent que :

$$\max\{\boldsymbol{yb} \mid \boldsymbol{y}\mathbf{A} = \boldsymbol{c}, \boldsymbol{y} \geq 0\} \quad = z^* = \quad \min\{\boldsymbol{cx} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$$

$$\textbf{Dual} \qquad \leq \qquad \textbf{Primal}$$

Théorème II.3 *Lorsque le problème primal (1) admet un minimum, le problème dual (2) admet un maximum et le minimum du primal coïncide avec le maximum du dual.*

Remarque Le passage du primal au dual est un jeu d'écriture : il intervertit le rôle des lignes et des colonnes qui viennent indicer les nouvelles variables et les nouvelles contraintes. Si l'on ajoute aux contraintes du primal les contraintes du dual, en prenant soin de lier les inconnues des deux problèmes par $\boldsymbol{cx} = z$ et $z = \boldsymbol{yb}$, alors le système résultant ne peut avoir qu'une solution en z. Si elle existe, cette solution est la solution optimale (primale ou duale).

La résolution d'un programme linéaire (problème OPT) est donc polynômialement équivalente à la résolution d'un système linéaire (problème SAT).

Corollaire *Réexprimons le primal en introduisant m variables d'écarts $s_i = \boldsymbol{a}_i \boldsymbol{x} - b_i \geq 0$. Alors, à l'optimum, ou bien $s_i = 0$ ou bien $y_i = 0$, $\forall\ i \in \{1, ..., m\}$.*

A l'optimum en effet, $\boldsymbol{yb} = \sum_{i=1}^{m} y_i \left(\sum_{j=1}^{n} a_{ij} x_j + s_i \right) = \boldsymbol{cx} = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} y_i a_{ij} \right) x_j$.

Donc $\sum_{i=1}^{m} y_i s_i = 0$, avec y_i et $s_i \geq 0$. ■

3.2 Application à la preuve de consécution

Nous revenons maintenant à la notion d'inégalité conséquence pour montrer que toute conséquence d'un système linéaire peut s'exprimer comme une combinaison linéaire positive de contraintes du système. Nous en tirerons une procédure de décision.

Théorème II.4 *Les propositions suivantes sont équivalentes :*

1. $\boldsymbol{cx} \geq d$ *est conséquence de* $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$
2. *il existe* $\boldsymbol{y} \geq 0$ *tel que* $\boldsymbol{y}\mathbf{A} = \boldsymbol{c}$ *et* $\boldsymbol{yb} \geq d$

Démonstration (2) implique (1) : si $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$, alors $\boldsymbol{cx} \geq d$ se déduit de $\boldsymbol{y}\mathbf{A}\boldsymbol{x} \geq \boldsymbol{yb}$.

Pour montrer que (1) implique (2), il faut examiner trois cas.

D'abord, si $\min\{\boldsymbol{cx} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$ existe et vaut z^*, alors $d \leq z^*$. En effet, si l'on avait $d > z^*$, la solution $\boldsymbol{x}$ satisfaisant à $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ et $\boldsymbol{cx} = z^*$ ne pourrait être solution de $\boldsymbol{cx} \geq d$. Donc $d \leq \max\{\boldsymbol{yb} \mid \boldsymbol{y}\mathbf{A} = \boldsymbol{c}, \boldsymbol{y} \geq 0\}$ et le point $\boldsymbol{y}$ satisfaisant à $\boldsymbol{y}\mathbf{A} = \boldsymbol{c}$, $\boldsymbol{y} \geq 0$ et à $\boldsymbol{yb} = z^*$ vérifie $\boldsymbol{yb} \geq d$.

Si le système d'inégalités $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ du primal est insatisfaisable, alors le dual $\max\{\boldsymbol{yb} \mid \boldsymbol{y}\mathbf{A} = \boldsymbol{c}, \boldsymbol{y} \geq 0\}$ est non borné supérieurement (on rappelle que le primal est le dual du dual). Il existe donc toujours $\boldsymbol{y} \geq 0$ tel que $\boldsymbol{y}\mathbf{A} = \boldsymbol{c}$ et $\boldsymbol{yb} \geq d$.

Enfin, si le primal $\min\{\boldsymbol{cx} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$ est non borné inférieurement, l'inégalité $\boldsymbol{cx} \geq d$ ne peut être une conséquence du système. ■

Ainsi, prouver qu'une inégalité $\boldsymbol{cx} \geq d$ est conséquence d'un système $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ équivaut à résoudre un système linéaire, en l'occurrence : $\boldsymbol{y}\mathbf{A} = \boldsymbol{c}, \boldsymbol{yb} \leq d, \boldsymbol{y} \geq 0$.

Remarque La tautologie $0 \geq 0$ est toujours conséquence d'un système, quel qu'il soit. Lorsque cette inégalité, C_0, est produite par élimination de Fourier, cela signifie qu'elle peut s'obtenir autrement que par une combinaison triviale (à coefficients nuls) de contraintes de S. Ce fait trahit une propriété cachée du système.

Définition Soit S un système linéaire et $C_i : \boldsymbol{a}_i\boldsymbol{x} \geq b_i$ une inégalité de S. On dit que C est une *égalité implicite* si l'inégalité $\boldsymbol{a}_i\boldsymbol{x} \leq b_i$ est conséquence de S.

Lassez et Maher démontrent que les propositions suivantes sont équivalentes:

1. C_k est une égalité implicite dans $\mathrm{S} = \bigcup_{i=1}^{m} C_i$
2. il existe des multiplicateurs $\boldsymbol{y} \geq 0$ vérifiant $\sum_{i=1} y_i C_i = C_0$ (*i.e.*, $0 \geq 0$), avec $y_i > 0$ pour $i = k$.

Exemple Soient les inégalités $C_1 : x \geq 1$ et $C_2 : -x \geq -1$. Dans le système $\{C_1, C_2\}$, l'égalité $x = 1$ est implicite et l'élimination de x donne : $C_1 + C_2 = C_0$.

Au système formé de toutes les *égalités implicites* de S correspond, géométriquement, un ensemble d'hyperplans dont l'intersection forme la plus petite *variété linéaire* contenant les solutions de S. La dimension de ce sous-espace de $\mathbb{R}^n$ définit la dimension du polyèdre des solutions.

4 Elimination de Fourier de variables entières

Nous abordons ici l'analyse de systèmes linéaires dont les inconnues ne portent plus simplement sur des nombres réels, mais sur des nombres entiers.

Selon Fourier, «si les conditions sont exprimées par des inégalités non linéaires, la question ne change pas de nature, et peut être encore traitée par les mêmes principes». On peut donc adapter la méthode sans se départir de son esprit.

Williams s'est intéressé en particulier à cette extension — difficile — du problème pour résoudre de programmes linéaires en nombres entiers. Dans une publication de 1976, il établit un rapprochement entre la procédure de décision de Presburger (pour le fragment existentiel de la théorie l'arithmétique sans multiplication) et l'élimination de Fourier. Il en tire une élégante généralisation (malheureusement, peu économique) qui nécessite l'introduction de relations de congruences (aux côtés des relations d'égalité et d'inégalité) dans des formules logiques où réapparaît la disjonction.

4.1 Satisfaisabilité sur les entiers

Définition On appelle *système linéaire en nombres entiers* tout système T défini par la donnée conjointe de contraintes linéaires $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ et de contraintes d'intégrité $\boldsymbol{x} \in \mathbb{Z}^n$. Lorsqu'on omet ces dernières, le système d'inégalités qui subsiste définit la *relaxation linéaire* du système T.

Définition On appelle *polyèdre en nombres entiers* l'ensemble Q des points à coordonnées entières d'un polyèdre convexe P.

La question est ici de savoir si un système $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ possède des solutions entières, autrement dit, si $Q = \{\boldsymbol{x} \in \mathbb{Z}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$ est non vide.

4.2 Incomplétude de l'élimination

Si l'élimination de Fourier répond parfaitement au problème de la satisfaisabilité d'un système linéaire sur les réels, elle est notoirement incomplète pour les systèmes en nombres entiers. Ceci peut s'appréhender géométriquement, à partir des propriétés de la projection.

Soient T un système linéaire en nombres entiers défini sur $\mathbb{R}^n$ et S sa relaxation linéaire. Posons Q = Sol(T) et P = Sol(S). Alors, $Q = P \cap \mathbb{Z}^n$.

La projection d'une intersection n'étant qu'incluse dans l'intersection des projections, on aura, pour tout axe x de $\mathbb{R}^n$:

$$\text{Proj}(Q, x) \subset \text{Proj}(P, x) \cap \mathbb{Z}^{n-1}.$$

On en tire l'implication :

$\text{Proj}(P, x) \cap \mathbb{Z}^{n-1} = \varnothing \quad \Rightarrow \quad Q = \varnothing$.

Mais sa réciproque est fausse :

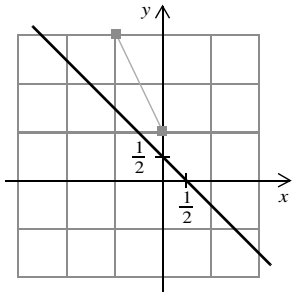
$\text{Proj}(P, x) \cap \mathbb{Z}^{n-1} \neq \varnothing \quad \not\Rightarrow \quad Q \neq \varnothing$

Pour que S soit satisfaisable sur les entiers, une condition nécessaire (mais non suffisante) est que l'élimination de Fourier de la variable x dans S produise un système $E_F(S, \boldsymbol{x})$ satisfaisable sur les entiers.

Exemple Prenons l'équation diophantienne $2x + 2y = 1$. Son membre gauche est pair, son membre droit impair. L'équation linéaire $2x + 2y = 1$ ne peut donc avoir de solution entière. Pourtant, les projections de la droite $2x + 2y = 1$ sur les axes du plan sont non bornées et contiennent intégralement $\mathbb{Z}$: l'élimination par Fourier d'une variable de $S = \{2x + 2y \geq 1, -2x - 2y \geq -1\}$ produit $E_F(S) = \{0 \geq 0\}$.

Considérons une autre équation diophantienne $2x + y = 1$, cette fois soluble. L'enveloppe convexe des solutions $\{(-1, 3), (0, 1)\}$ est définie par le polyèdre $P = \{(x, y) \in \mathbb{R}^2 \mid -1 \leq x \leq 0, 2x + y = 1\}$.
Posons $Q = \{(x, y) \in \mathbb{Z}^2 \mid -1 \leq x \leq 0, 2x + y = 1\}$. Par élimination de x, on obtient : $\text{proj}_y(P) = [1, 3]$. Or, $\text{proj}_y(Q) = \{1, 3\} \neq [1, 3] \cap \mathbb{Z} = \{1, 2, 3\}$.

La présence de «trous» dans l'enveloppe convexe montre, sur cet exemple, que la projection du polyèdre en nombres entiers $Q = P \cap \mathbb{Z}^2$ n'est pas — au sens de notre définition — un polyèdre en nombres entiers.



L'élimination de Fourier, qui procède d'une analyse sur les réels, donne, pour les systèmes linéaires en nombres entiers, une procédure de décision incomplète, similaire à la méthode SUP-INF de Bledsoe.

Celle-ci est capable de démontrer l'insatisfaisabilité d'un système ($Q = \varnothing$ dès lors que $\text{proj}_j(P) \cap \mathbb{Z} = \varnothing$), non sa satisfaisabilité.

4.3 Cause de l'incomplétude

Le problème vient de ce que les conditions qui subsistent après élimination d'une variable entière ne sont plus suffisantes pour garantir à celle-ci l'existence de valeurs compatibles.

Si l'on retourne à la démonstration de complétude de l'élimination sur les réels (théorème II.1), on voit que, dans le cas discret — contrairement au cas continu, on ne peut plus conclure à l'existence certaine d'une valeur entière entre les bornes rationnelles $\max_{l \in \mathrm{I}_0^+} f_{l0}(\boldsymbol{x}')$ et $\min_{k \in \mathrm{I}_0^-} F_{k0}(\boldsymbol{x}')$.

Il faudrait pour cela que l'on ait : $\lceil f_{l0}(\boldsymbol{x}') \rceil \leq \lfloor F_{k0}(\boldsymbol{x}') \rfloor$, $\forall k \in \mathrm{I}_0^-$ $\forall l \in \mathrm{I}_0^+$.

4.4 Conditions pour y remédier

Lors de l'élimination d'une variable x_0, il faut donc garantir l'existence d'une telle valeur entière par des conditions supplémentaires. Cela vaut pour pour toute paire C_k , C_l d'inégalités complémentaires vis à vis de x_0.

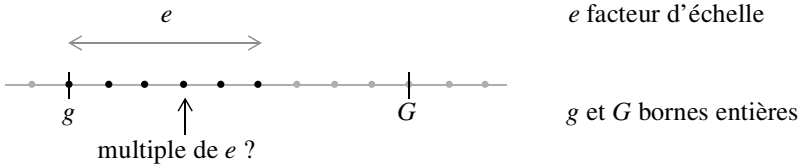
Puisque les membres de la combinaison entière $a_{l0}C_k + a_{k0}C_l$ sont tirés de la double inégalité :

$$a_{k0}a_{l0} \cdot f_{l0}(\boldsymbol{x}') \leq a_{k0}a_{l0} \cdot x_0 \leq a_{k0}a_{l0} \cdot F_{k0}(\boldsymbol{x}') ,$$

il faut exiger, pour être complet, la présence d'un multiple de $a_{k0}a_{l0}$ entre ces deux formes affines, gauche et droite, à coefficients entiers.

Or, comme le montre Williams, une condition comme «il existe un multiple d'un nombre entier e compris entre deux expressions entières g et G» peut être reformulée à l'aide d'inégalités et de congruences.

En effet, un tel multiple s'écrira $g + h$ (ou encore $G – h$), h étant un entier compris entre 0 et $G–g$ (on pourra choisir h strictement inférieur à e). Le fait que $g + h$ soit divisible par e se traduira par : $g + h \equiv 0 \pmod{e}$.



D'où les conditions à poser pour toute paire d'inégalités complémentaires :

$$h \in \{0, 1, \ldots, e–1\},\ h \leq G – g\ ,\ \ g + h \equiv 0 \pmod{e}$$

Notons qu'on introduit ici une variable entière h qui a au plus $e = a_{k0}a_{l0}$ valeurs[1] — alors que la variable entière éliminée pouvait ne pas être bornée. L'intérêt de l'opération sera d'autant plus grand que les valeurs a_{k0} et a_{l0} seront petites.

1. Au lieu de prendre le produit, on peut encore se limiter au PPCM de a_{k0} de a_{l0}.

Sans introduire de nouvelle variable entière, on peut encore écrire :

$$\{0 \leq G - g\ ,\ g \equiv 0 \pmod{e}\} \vee \ldots \vee \{e-1 \leq G - g\ ,\ g + e - 1 \equiv 0 \pmod{e}\}$$

Le résultat de l'élimination d'une variable entière d'un système linéaire devient une disjonction de systèmes, où chaque système s'enrichit d'une conjonction de congruences linéaires. On conçoit que cela renchérisse de beaucoup une élimination déjà passablement coûteuse[1].

4.5 Cas d'élimination licite

Intéressons-nous aux cas où le problème se simplifie. Sous quelles hypothèses peut-on pratiquer l'élimination de Fourier sans avoir à rajouter de conditions supplémentaires ?

Définition Soient T un système linéaire en nombres entiers et S sa relaxation linéaire. Si x est une variable de T et S' = E_F(S, x), on notera E_F(T, x) le système linéaire en nombre entier T' dont S' est la relaxation linéaire.

On dira que l'élimination de Fourier de la variable x de T est *licite* si elle respecte l'équivalence :

$$\exists\, x_0 \in \mathbb{Z} \text{ t.q. } (x_0, \boldsymbol{x'}) \in \text{Sol(T)} \quad \Leftrightarrow \quad \boldsymbol{x'} \in \text{Sol(T')}$$

En d'autres termes, et en reprenant les notations Q = Sol(T), P = Sol(S), Q = P $\cap \mathbb{Z}^n$, l'élimination de la variable entière x sera licite si et seulement si :

$$\text{Proj(Q, } x\text{)} = \text{Proj(P, } x\text{)} \cap \mathbb{Z}^{n-1}$$

Procéder à des éliminations licites garantira la complétude de la résolution sur des problèmes en nombres entiers.

Propriété *Une condition suffisante pour que l'élimination d'une variable entière x_0 soit licite est que cette variable ait au moins un coefficient a_{k0} ou a_{l0} égal à* 1 *dans toute paire (C_k , C_l) d'inégalités complémentaires.*

En effet, puisqu'il n'y a pas de division, l'une des deux formes affines f_{k0} ou F_{l0} est entière. Chaque encadrement $f_{l0}(\boldsymbol{x'}) \leq x_0 \leq F_{k0}(\boldsymbol{x'})$ présente donc au moins une borne entière (égale à sa partie entière). On en déduit :

$$\lceil f_{l0}(\boldsymbol{x'}) \rceil \leq \lfloor F_{k0}(\boldsymbol{x'}) \rfloor,\ \forall k \in \mathrm{I}_0^-\ \forall l \in \mathrm{I}_0^+$$

Ce qui suffit à prouver que T est satisfaisable quand E_F(T, x_0) l'est. ■

1. En termes de complexité, on passe d'une «simple» exponentielle (dans le cas continu) à une tour d'exponentielles (dans le cas discret).

5 Une procédure alternative pour l'élimination

L'élimination de variables entières nous a fait sortir du cadre des systèmes de contraintes purement conjonctifs pour aborder des problèmes non convexes. Si l'on revient à des variables réelles — en mettant de côté les contraintes d'intégrité, il apparaît, paradoxalement, que la gestion de disjonctions de systèmes d'inégalités linéaires permet d'abaisser la complexité théorique de l'élimination. Ce résultat, dû à Weispfenning (1988), conclura notre panorama.

5.1 La méthode des points-tests

Reprenons un système linéaire S et une variable x de S. Comme précédemment, S se décompose vis à vis de x en 3 sous-systèmes disjoints, avec des inégalités de la forme :

- $f_i \leq x$, pour $i \in \mathrm{I}^+$
- $x \leq F_i$, pour $i \in \mathrm{I}^-$
- $0 \leq G_i$, pour $i \in \mathrm{I}^0 = \mathrm{I} - (\mathrm{I}^+ \cup \mathrm{I}^-)$

f_i, F_i et G_i désignant des formes affines fonction des autres variables du système.

Pour éliminer x, réécrivons tout d'abord la formule logique définissant S :

$$\exists\, x \quad (\, \forall\, i \in \mathrm{I}^+ \forall\, j \in \mathrm{I}^- \forall\, k \in \mathrm{I}^0 \quad f_i \leq x \wedge x \leq F_j \wedge 0 \leq G_k \,)$$

en :

$$(\, \exists\, x \quad (\, \forall\, i \in \mathrm{I}^+ \forall\, j \in \mathrm{I}^- \quad f_i \leq x \leq F_j \,)\,) \wedge (\, \forall\, k \in \mathrm{I}^0 \quad 0 \leq G_k \,)$$

Le théorème d'élimination de Fourier donne :

$$\exists\, x \quad (\, \forall\, i \in \mathrm{I}^+ \forall\, j \in \mathrm{I}^- \quad f_i \leq x \leq F_j \,)$$
$$\Leftrightarrow$$
$$\forall\, i \in \mathrm{I}^+ \forall\, j \in \mathrm{I}^- \quad f_i \leq F_j$$

Lorsque I^+ est non vide, cette dernière formule peut s'écrire :

$$\forall\, j \in \mathrm{I}^- \max_{i \in \mathrm{I}^+} f_i \leq F_j$$

soit encore, puisque I^+ est fini :

$$\exists\, h \in \mathrm{I}^+ \forall\, i \in \mathrm{I}^+ \forall\, j \in \mathrm{I}^- \quad f_i \leq f_h \leq F_j$$

L'élimination de x dans S donne ainsi naissance à une disjonction de conjonctions.

La *méthode des points-tests* se fonde sur l'équivalence :

$$\exists x \left(\bigwedge_{i \in I^+} \bigwedge_{j \in I^-} f_i \leq x \leq F_j \right)$$

$$\Leftrightarrow$$

$$\bigvee_{h \in I^+} \bigwedge_{i \in I^+} \bigwedge_{j \in I^-} f_i \leq f_h \leq F_j$$

Elle permet de tester la satisfaisabilité d'un système linéaire en se limitant à une sélection finie de points. Les différentes formes f_h correspondent aux substitutions candidates pour x. On peut faire remonter à Skolem cette idée d'utiliser un ensemble fini de termes pour témoigner de l'existence d'une solution.

Symétriquement, en prenant comme valeurs test les min. des F_h au lieu des max. des f_h , on produira :

$$\bigvee_{h \in I^-} \bigwedge_{i \in I^+} \bigwedge_{j \in I^-} f_i \leq F_h \leq F_j$$

Une fois l'élimination d'un quantificateur effectuée, et avant de réitérer l'opération avec les autres variables, on reportera les disjonctions en tête, en distribuant les quantificateurs sur chacun des sous-problèmes :

$$\exists \boldsymbol{x'} \left(\bigvee_{h \in I^-} \varphi_h \right) \Leftrightarrow \bigvee_{h \in I^-} \left(\exists \boldsymbol{x'} \ \varphi_h \right)$$

Par le traitement séparé de l'élimination d'une variable dans chacun des systèmes d'une forme disjonctive, la méthode de Weispfenning développe un arbre de recherche dont la racine est le système d'origine et les feuilles des systèmes d'inégalités sans variable. Lorsqu'une de ces feuilles est vraie, on obtient une preuve de la satisfaisabilité du système d'origine (on peut alors en reconstituer une solution particulière : le point test[1]) ; lorsque toutes sont fausses, le système est insatisfaisable.

5.2 Complexité comparée

La principale source de complexité vient du nombre de variables du système (qui sont toutes quantifiées existentiellement).

Si l'on part de m inégalités linéaires, la méthode de Fourier est susceptible de produire $m^2/4$ contraintes, puis $m^4/16$, soit après n éliminations de variables, $\frac{m^{2^n}}{2^n}$ inégalités. La croissance du système est doublement exponentielle.

1. Comme pour le Simplexe, cette solution particulière correspond à un sommet du polyèdre des solutions (du moins, lorsque ce polyèdre est borné). La remontée de ce résultat s'effectue à partir d'un système d'équations mis sous forme «triangulaire». Son coût est celui d'une *back-substitution* dans l'élimination de Gauss.

A contrario, avec la méthode des points-tests, la taille, m, de chacun des systèmes ne croît pas. Mais leur nombre est multiplié, au pire par $m/2$ (puisqu'on a le choix entre $|I^+|$ et $|I^-|$ points tests). La progression en nombre d'inégalités se ralentit : $m.m/2$, $m.(m/2)^2$, …, $m.(m/2)^n$. Elle n'est plus que simplement exponentielle, en $O(m^{n+1})$ — ce qui reste considérable.

5.3 Un exemple

Soit à résoudre le programme linéaire :

$$\text{Minimiser } 2x - 3y$$
$$\text{sous les contraintes } \{-2x + y \geq -4,\ x - 2y \geq -2,\ 2x + y \geq 2,\ x \geq 0,\ y \geq 0\}$$

Après introduction d'une variable z, le système linéaire S s'écrit :

$$\begin{aligned}
C_0 &: & z - 2x + 3y &\geq 0\\
C_1 &: & -2x + y &\geq -4\\
C_2 &: & x - 2y &\geq -2\\
C_3 &: & 2x + y &\geq 2\\
C_4 &: & x &\geq 0\\
C_5 &: & y &\geq 0
\end{aligned}$$

5.3.1 Résolution par la méthode de Fourier

L'élimination de x dans S donne le système $E_F(S, x)$:

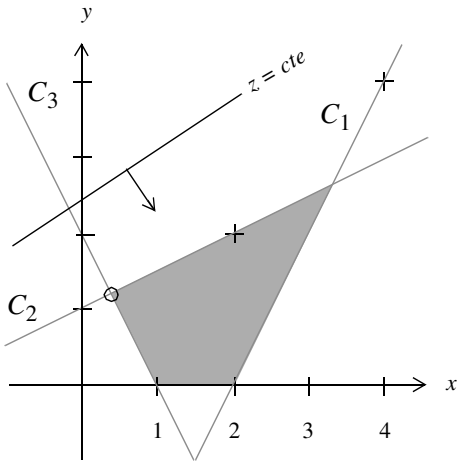
$$\begin{aligned}
C_6 = C_0 + 2C_2 &: & z - y &\geq -4 & \\
C_7 = C_0 + C_3 &: & z + 4y &\geq 2 & \\
C_8 = C_0 + 2C_4 &: & z + 3y &\geq 0 & \textit{redondante}\\
C_9 = C_1 + 2C_2 &: & -3y &\geq -8 & \\
C_{10} = C_1 + C_3 &: & 2y &\geq -2 & \textit{redondante}\\
C_{11} = C_1 + 2C_4 &: & y &\geq -4 & \textit{redondante}\\
C_5 &: & y &\geq 0 &
\end{aligned}$$

Puis l'élimination de y dans $E_F(S, x)$ donne un système équivalent à $z \in [-14/5, +\infty[$:

$$\begin{aligned}
C_{12} = 4C_6 + C_7 &: & 5z &\geq -14 & \\
C_{13} = 3C_6 + C_8 &: & 4z &\geq -12 & \textit{redondante}\\
C_{14} = 2C_6 + C_{10} &: & 2z &\geq -10 & \textit{redondante}\\
C_{15} = C_6 + C_{11} &: & z &\geq -8 & \textit{redondante}\\
C_{16} = C_6 + C_5 &: & z &\geq -4 & \textit{redondante}\\
C_{17} = 4C_9 + 3C_7 &: & 3z &\geq -26 & \textit{redondante}
\end{aligned}$$

$C_{18} = C_9 + C_8$:	$z \geq -8$	*redondante*
$C_{19} = 2C_9 + 3C_{10}$:	$0 \geq -22$	*redondante*
$C_{20} = C_9 + 3C_{11}$:	$0 \geq -20$	*redondante*
$C_{21} = C_9 + 3C_5$:	$0 \geq -8$	*redondante*

S est donc satisfaisable et le minimum vaut $z = -14/5$. Pour cette valeur, l'inégalité $C_{12} = 4C_6 + C_7 = 5C_0 + 8C_2 + C_3$ est une égalité. Donc les contraintes C_0, C_2, C_3, C_6, C_7 sont également saturées. De C_6, on tire $y = z + 4$ et de C_2, $x = 2y - 2$, ce qui correspond au sommet $(x, y) = (2/5, 6/5)$.



5.3.2 Résolution par la méthode des points-tests

On part à nouveau du système S :

C_0 :	$z + 3y \geq 2x$
C_1 :	$y + 4 \geq 2x$
C_2 :	$x \geq 2y - 2$
C_3 :	$2x \geq -y + 2$
C_4 :	$x \geq 0$
C_5 :	$y \geq 0$

Pour l'élimination de x, le plus petit nombre d'alternatives est donné par $|I^-|$, avec C_0 et C_1.

Si l'on choisit comme valeur témoin $2x = z + 3y$, on obtient le système S_1 :

$$\begin{array}{lrr}
C_0 : & z + 3y \geq z + 3y & 0 \geq 0 \\
C_1 : & y + 4 \geq z + 3y & -z + 4 \geq 2y \\
C_2 : & z + 3y \geq 4y - 4 & z + 4 \geq y \\
C_3 : & z + 3y \geq -y + 2 & 4y \geq -z + 2 \\
C_4 : & z + 3y \geq 0 & 3y \geq -z \\
C_5 : & y \geq 0 & y \geq 0
\end{array}$$

A partir de S_1, si l'on fait le choix de la valeur témoin $2y = -z + 4$ (tirée de C_1), on obtient le système S_{11} :

$$\begin{array}{lrr}
C_0 : & & \\
C_1 : & -z + 4 \geq -z + 4 & 0 \geq 0 \\
C_2 : & 2z + 8 \geq -z + 4 & 3z \geq -4 \\
C_3 : & -2z + 8 \geq -z + 2 & 6 \geq z \\
C_4 : & -3z + 12 \geq -2z & 12 \geq z \\
C_5 : & -z + 4 \geq 0 & 4 \geq z
\end{array}$$

A partir de S_1, si l'on fait le choix de l'autre valeur témoin $y = z + 4$ (tirée de C_2), on obtient le système S_{12} :

$$\begin{array}{lrr}
C_0 : & & \\
C_1 : & -z + 4 \geq 2z + 8 & -4 \geq 3z \\
C_2 : & z + 4 \geq z + 4 & 0 \geq 0 \\
C_3 : & 4z + 16 \geq -z + 2 & 5z \geq -14 \\
C_4 : & 3z + 12 \geq -z & 4z \geq -12 \\
C_5 : & z + 4 \geq 0 & z \geq -4
\end{array}$$

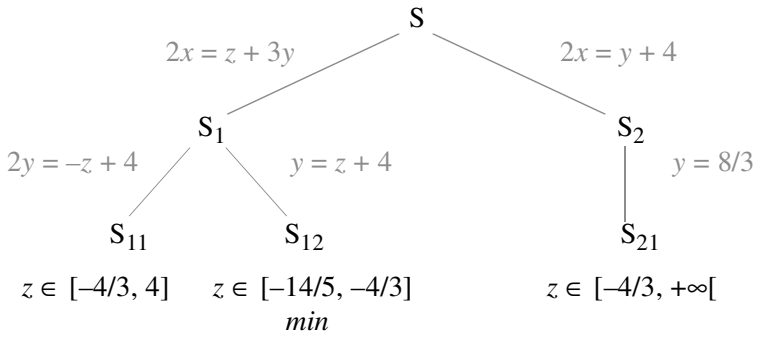
En remontant à S et en choisissant cette fois comme valeur témoin $2x = y + 4$, on obtient, après élimination de x, le système S_2 :

$$\begin{array}{lrr}
C_0 : & z + 3y \geq y + 4 & 2y \geq -z + 4 \\
C_1 : & y + 4 \geq y + 4 & 0 \geq 0 \\
C_2 : & y + 4 \geq 4y - 4 & 8 \geq 3y \\
C_3 : & y + 4 \geq -y + 2 & 2y \geq -2 \\
C_4 : & y + 4 \geq 0 & y \geq -4 \\
C_5 : & y \geq 0 & y \geq 0
\end{array}$$

Pour l'élimination de y, on peut alors se cantonner à la valeur témoin $y = 8/3$ (tirée de C_2). Le système substitué est S_{21} :

C_0 :	$16/3 \geq -z + 4$	$z \geq -4/3$
C_1 :		
C_2 :	$8 \geq 8$	$0 \geq 0$
C_3 :	$16/3 \geq -2$	$0 \geq 0$
C_4 :	$8/3 \geq -4$	$0 \geq 0$
C_5 :	$8/3 \geq 0$	$0 \geq 0$

La figure suivante représente l'arbre développé :



Eliminer z ne donnerait que des feuilles équivalentes à la valeur logique vrai.

On constate que l'union des intervalles solutions pour z reconstitue la projection de S sur cet axe : $\text{proj}_z(S) = [-14/5, +\infty[$.

Le point-test correspondant au minimum $z = -14/5$ est obtenu par résolution du système $\{2x = z + 3y, y = z + 4\}$. On retrouve bien : $y = 6/5$, $x = 2/5$.

D'une plus grande généralité, la méthode de Weispfenning s'applique encore à l'extérieur du cadre linéaire.

CHAPITRE III

Des familles de polyèdres ayant un plus petit élément

Propriétés remarquables des inégalités monotones

> «La composition des diverses machines employées dans les travaux des arts, présente à des yeux vulgaires une multitude d'inventions isolées, et, pour ainsi dire, de secrets dont chacun en particulier semble avoir nécessité une découverte ; mais pour le mécanicien habile, que des études théoriques ont éclairé, ces appareils si compliqués et si variés en apparence, se réduisent au fond à un petit nombre d'éléments simples, qui s'assemblent et se combinent les uns les autres d'une infinité de manières appropriées aux effets que l'on veut produire. Ce genre de combinaisons est lui-même soumis à des règles fixes, et les effets qui en résultent peuvent être généralement déterminés par le calcul. On conçoit combien l'étude des machines, présentée sous ce point de vue élevé, doit être utile à toutes les classes d'ingénieurs.»
>
> J. HACHETTE

Dans ce chapitre, nous nous focalisons sur une classe particulière de systèmes linéaires, les systèmes monotones, pour en étudier la résolution sur les réels et les entiers. Ces systèmes sont aisément identifiables au vu de leur structure syntaxique. Nous aurions pu les qualifier d'«unilatères» ou de «monotropes» car toutes les inégalités qui les composent sont tournées dans le même sens : vers un monôme à coefficient positif, lequel est par définition unique si ce coefficient est non nul.

Nous montrons que, dans l'espace produit muni de l'ordre $\geq$, les systèmes monotones décrivent des demi-treillis (continus ou discrets). Cette intéressante propriété de clôture algébrique permet de garantir l'existence d'un plus petit (ou plus grand) élément lorsque l'ensemble des solutions est non vide et borné (ou plus simplement minoré ou majoré).

Réciproquement, d'après un résultat de Veinott, tout polyèdre de $\mathbb{R}^n$ présentant cette particularité géométrique peut être défini par un système monotone.

Nous identifions également une sous-classe remarquable de systèmes monotones qui forme une classe polynômiale de problèmes linéaires en nombres entiers.

Quand les coefficients du système sont entiers et qu'en particulier les monômes positifs ont des coefficients égaux à 1, le calcul des inégalités dans $\mathbb{Z}^n$ ne diffère pas du calcul dans $\mathbb{R}^n$. Pour ces systèmes 1-monotones, la relaxation linéaire continue donne une procédure de décision complète.

De plus, l'extrêmum global qui caractérise un système monotone et borné construit avec de telles inégalités s'avère être le même pour les réels et les entiers : lorsqu'il existe, ce plus petit (ou plus grand) élément est à coordonnées entières. Cette solution particulière sera un témoin privilégié pour le test de satisfaisabilité.

Nous présentons enfin un algorithme de résolution pour les systèmes monotones à variables entières n'entrant pas dans la catégorie précédente. En procédant par approximations et troncatures successives, ce dernier converge, lorsque le système est borné, vers le plus petit (ou plus grand) élément solution — ou démontre sinon l'absence de solution entière.

1 Produits et treillis vectoriels

Rappelons, pour commencer, quelques définitions.

Définition L'*ordre produit* est la relation d'ordre partiel définie sur $\mathbb{R}^n$ ou $\mathbb{Z}^n$ par le système d'inégalités simultanées , :

$$\boldsymbol{x} \geq \boldsymbol{y} \quad \Leftrightarrow \quad x_j \geq y_j \,,\, \forall j \in \{1, \ldots, n\}$$

On écrira : $\boldsymbol{x} > \boldsymbol{y}$ lorsque $\boldsymbol{x} \geq \boldsymbol{y}$ et qu'au moins une des composantes x_j de $\boldsymbol{x}$ est strictement supérieure à y_j.

Définition On munit l'espace produit ($\mathbb{R}^n$ ou $\mathbb{Z}^n$) de *lois de composition internes* inf et sup qui associent à tout couple de vecteurs $(\boldsymbol{x}, \boldsymbol{y})$ le vecteur $\boldsymbol{z}$ dont les composantes sont les plus petites des composantes de $\boldsymbol{x}$ et de $\boldsymbol{y}$ (respectivement, des plus grandes) :

$$\inf(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{z} \quad \Leftrightarrow \quad \forall j \in \{1, \ldots, n\},\ z_j = \min\{x_j \,,\, y_j\}$$

$$\sup(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{z} \quad \Leftrightarrow \quad \forall j \in \{1, \ldots, n\},\ z_j = \max\{x_j \,,\, y_j\}$$

Remarque Ces lois sont commutatives, associatives et idempotentes.
On a en particulier : $\inf(\boldsymbol{x}, \boldsymbol{y}) = \inf\{\boldsymbol{x}, \boldsymbol{y}\}$ et : $\boldsymbol{x} \geq \boldsymbol{y} \ \Leftrightarrow \ \inf(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{y}$.

Exemple Les restrictions de ces lois à $\{0,1\}^n$ correspondent aux définitions de la conjonction logique $\wedge$ pour l'inf et de la disjonction logique $\vee$ pour le sup. L'ordre $y \geq x$ sur $\{0,1\}$ s'identifie à l'implication $x \Rightarrow y$.
$(\{0,1\}^n, \geq)$ n'est autre que le treillis booléen.

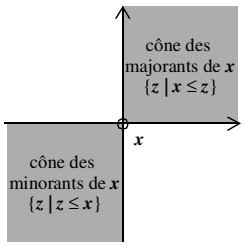
Définition Un ensemble partiellement ordonné dans lequel toute paire d'éléments admet une borne inférieure (resp. supérieure) est un *demi-treillis* inférieur (resp. supérieur).
Un *treillis* est à la fois un demi-treillis inférieur et un demi-treillis supérieur.

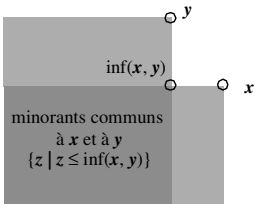
Propriété *Les lois* inf *et* sup *confèrent à* $(\mathbb{R}^n, \geq)$ *et* $(\mathbb{Z}^n, \geq)$ *une structure de treillis.*

En effet, toute paire $\{\boldsymbol{x}, \boldsymbol{y}\}$ d'éléments de l'espace produit admet pour borne inférieure $\inf\{\boldsymbol{x}, \boldsymbol{y}\}$, puisque l'on a : $\inf(\boldsymbol{x}, \boldsymbol{y}) \leq \boldsymbol{x}$, $\inf(\boldsymbol{x}, \boldsymbol{y}) \leq \boldsymbol{y}$ et quel que soit $\boldsymbol{z}$ vérifiant $\boldsymbol{z} \leq \boldsymbol{x}$ et $\boldsymbol{z} \leq \boldsymbol{y}$, $\boldsymbol{z}$ satisfait à l'inégalité $\boldsymbol{z} \leq \inf(\boldsymbol{x}, \boldsymbol{y})$.

De même, toute paire $\{\boldsymbol{x}, \boldsymbol{y}\}$ d'éléments de l'espace produit admet pour borne supérieure $\sup\{\boldsymbol{x}, \boldsymbol{y}\}$ (plus petit des majorants). ■

Géométriquement, la borne inférieure $\inf\{\boldsymbol{x}, \boldsymbol{y}\}$ se situe au sommet du cône des minorants de $\boldsymbol{x}$ et de $\boldsymbol{y}$.





2 Systèmes monotones

Définition Une inégalité linéaire est inf-monotone si elle est de la forme :
$x_j \geq \boldsymbol{a}\boldsymbol{x} + b$, avec $\boldsymbol{a} \geq 0$ et $a_j = 0$, ou encore de la forme : $0 \geq \boldsymbol{a}\boldsymbol{x} + b$, avec $\boldsymbol{a} \geq 0$.

L'inégalité exprime que la valeur d'une variable (ou d'une constante) borne supérieurement une fonction linéaire croissante (à coefficients rationnels) des autres variables du système.

Définition Une inégalité linéaire est sup-monotone si elle est de la forme :
$x_j \leq \boldsymbol{a}\boldsymbol{x} + b$, avec $\boldsymbol{a} \geq 0$ et $a_j = 0$, ou encore de la forme : $0 \leq \boldsymbol{a}\boldsymbol{x} + b$, avec $\boldsymbol{a} \geq 0$.

Exemple $2x - 3y - 5z - t \geq -8$ est monotone, alors que $2x - 3y - 5z + t \geq -8$ ne l'est pas.

Définition Un système linéaire est *monotone* si toutes les inégalités qui le composent sont soit toutes sup-monotones soit toutes inf-monotones.

Avec des coefficients entiers (et non plus rationnels) pour $\mathbf{A}$ et $\boldsymbol{b}$, les contraintes C_i d'un système inf-monotone se présentent, sous la forme :

$$(1)\quad a_{ij}x_j - \sum_{k \neq i} a_{ik}x_k \geq b_i \qquad \text{ou} \qquad (2)\quad -\sum_{1 \leq i \leq n} a_{ij}x_j \geq b_i$$

avec $a_{ij}, a_{ik} \in \mathbb{N}$ pour tout j et tout k, et $b_i \in \mathbb{Z}$.
Le cas sup-monotone s'obtient en remplaçant globalement le signe ≥ par ≤.

La définition précédente s'appuie essentiellement sur une propriété de structure de la matrice $m \times n$ du système d'inéquations.

Tout système monotone peut être normalisé sous la forme $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ ou $\mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}$ où la matrice $\mathbf{A}$ a *au plus un coefficient strictement positif par ligne*[1].

Exemple Le cube unité de $\mathbb{R}^n$ est défini, indifféremment, par l'un des systèmes monotones : $\begin{pmatrix} \mathbf{I} \\ -\mathbf{I} \end{pmatrix}\boldsymbol{x} \geq \begin{pmatrix} 0 \\ -1 \end{pmatrix}$ ou : $\begin{pmatrix} \mathbf{I} \\ -\mathbf{I} \end{pmatrix}\boldsymbol{x} \leq \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

Exemple Voici deux ensembles inscrits dans le carré unité.

Le losange $L = \{(x, y) \mid 1 \leq 2x + 2y, 2x + 2y \leq 3, 2y \leq 2x + 1, 2x \leq 2y + 1\}$ n'est pas défini par un système monotone.
Le polygone $P = \{(x, y) \mid 2x + 2y \leq 3, 2y \leq 2x + 1, 2x \leq 2y + 1, 0 \leq x, 0 \leq y\}$ représente en revanche l'ensemble des solutions d'un système monotone.
Ce système peut en effet se réécrire $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ avec :

$$\mathbf{A} = \begin{array}{|cc|} \hline -2 & -2 \\ +2 & -2 \\ -2 & +2 \\ +1 & 0 \\ 0 & +1 \\ \hline \end{array}$$

Le schéma de droite visualise bien une propriété de monotonie : le minimum de y est une fonction croissante de x.

Exemple Soient a et b sont deux entiers positifs tels que $a \geq b > 0$. Le PGCD, d , de a et b se définit comme le plus grand diviseur de b divisant a :

$$\begin{aligned} d &= \max\{z \in \mathbb{N}^* \mid \exists\, x \in \mathbb{N}^* \; \exists\, y \in \mathbb{N}^* \; a = xz, b = yz\} \\ &= \max\{\frac{b}{y} \mid \exists\, x \in \mathbb{N}^* \; \frac{a}{x} = \frac{b}{y}\} \end{aligned}$$

d est donc égal à l'entier $\frac{b}{\underline{y}}$, avec $\underline{y}$ défini par :

$$\begin{aligned} \underline{y} &= \min\{y \in \mathbb{N}^* \mid \exists\, x \in \mathbb{N}^* \; ay = bx\} \\ &= \min \mathrm{proj}_y\{(x, y) \in \mathbb{Z}^2 \mid x \geq 1, y \geq 1, ay \geq bx, bx \geq ay\} \end{aligned}$$

Dans cette définition, les inégalités $x \geq 1$, $y \geq 1$, $ay \geq bx$, $bx \geq ay$ forment un système monotone.

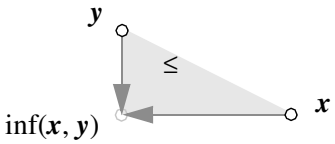
1. On trouve dans la littérature une définition relative à la transposée de la matrice $\mathbf{A}$: ${}^t\mathbf{A}$, dont chaque colonne a au plus un élément positif, est qualifiée de «pré-Léontief».

3 Propriétés générales des systèmes monotones

3.1 Monotonie

La monotonie exprime la stabilité de l'ensemble des solutions par passage à l'inf ou au sup.

Propriété *Soit E l'ensemble des solutions d'un système monotone* $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$. *Si* $\boldsymbol{x}$ *et* $\boldsymbol{y}$ *sont deux élements de E, alors leur borne inférieure* $\inf(\boldsymbol{x}, \boldsymbol{y})$ *appartient à E.*



Démonstration Posons $\boldsymbol{z} = \inf(\boldsymbol{x}, \boldsymbol{y})$. Pour toute contrainte C du système de forme (1), on a :

pour la solution $\boldsymbol{x}$: $x_j \geq \boldsymbol{a}\boldsymbol{x} + b \geq \boldsymbol{a}\boldsymbol{z} + b$,
pour la solution $\boldsymbol{y}$: $y_j \geq \boldsymbol{a}\boldsymbol{y} + b \geq \boldsymbol{a}\boldsymbol{z} + b$.
Par conséquent : $\min\{x_j, y_j\} = z_j \geq \boldsymbol{a}\boldsymbol{z} + b$, ce qui montre que $\boldsymbol{z}$ satisfait à C.

Il en va de même pour toute contrainte de forme (2) : $\boldsymbol{z}$ satisfait à toutes les contraintes définissant E. ■

Corollaire *L'ensemble des solutions réelles ou entières d'un système monotone* $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ *est un demi-treillis inférieur.*

Si l'on change le sens des inégalités du système, on a une propriété similaire (le suprémum remplaçant l'infimum) :

Propriété *L'ensemble des solutions réelles ou entières d'un système monotone* $\mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}$ *est un demi-treillis supérieur.*

Par la suite, sauf mention explicite, on ne parlera de système monotone que pour le sens ≥.

Remarque Plus intuitivement, les relations n-aires définies par des systèmes monotones $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$, ont pour tout $\boldsymbol{c} \in \mathbb{Z}^n$ la propriété suivante, où $\mathrm{P} = \{\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{c}\}$:

les n fonctions qui à $(x_1,..., x_{j-1}, x_{j+1},..., x_n)$ associent $\min\{x_j \mid \boldsymbol{x} \in \mathrm{P}\}$ sont (1) définies sur les projections de P et (2) croissantes.

3.2 Stabilité par intersection

Propriété *Si l'on rajoute des contraintes monotones à un système monotone — en respectant le sens d'orientation des inégalités, on conserve un système de contraintes monotone.*

Si $\mathbf{A}_1\boldsymbol{x} \geq \boldsymbol{b}_1$ et $\mathbf{A}_2\boldsymbol{x} \geq \boldsymbol{b}_2$ sont deux systèmes monotones, la matrice $\begin{pmatrix}\mathbf{A}_1\\ \mathbf{A}_2\end{pmatrix}$ résultant de la conjonction de ces systèmes a, par construction, au plus un coefficient strictement positif par ligne. ■

Exemple Si S est un système monotone, $S \cup \{\boldsymbol{x} \geq 0\}$ est également monotone.

Géométriquement parlant, l'intersection de deux demi-treillis inférieurs est un demi-treillis inférieur.

3.3 Existence d'un extrêmum global

Définition Un ensemble E possède un *plus petit élément* (ou minimum global) $\boldsymbol{m}$ si $\boldsymbol{m} \in$ E et si $\forall\, \boldsymbol{x} \in$ E, $\boldsymbol{x} \geq \boldsymbol{m}$.

Le plus petit élément est unique lorsqu'il existe. On le note min E.

Théorème III.1 *Soit* $\mathrm{E} = \{\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \geq 0\}$ *l'ensemble des solutions positives, réelles ou entières, d'un système monotone. Si* E *est non vide, alors* E *possède pour l'ordre produit un plus petit élement.*

Démonstration Si E est non vide, ses projections sur les axes de l'espace sont non vides.

- Si E est défini sur $\mathbb{R}^n$, ses projections sont des intervalles fermés inférieurement (toutes les inégalités étant larges).
- Si E est défini sur $\mathbb{Z}^n$, ses projections sont des sous-ensembles de $\mathbb{N}$.

Dans tous les cas, ces projections possèdent un plus petit élément.

Notons $m_j = \min \mathrm{proj}_j(\mathrm{E})$ pour $j = 1, \ldots, n$. Montrons que le vecteur $\boldsymbol{m} \geq 0$ ainsi construit est solution de E.

Par définition de la $j^{\text{ième}}$ projection, $m_j \in \mathrm{proj}_j(\mathrm{E})$ s'il existe des nombres $(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)$ tels que $(x_1, \ldots, x_{j-1}, m_j, x_{j+1}, \ldots, x_n) \in$ E.

Soit C_i une contrainte du système monotone définissant E.

Si C_i est de type (1), notons j l'indice du coefficient positif de C_i.

Comme $(x_1, \ldots, x_{j-1}, m_j, x_{j+1}, \ldots, x_n) \in$ E, on a :

$$a_{ij}m_j \geq \sum_{k \neq i} a_{ik}x_k + b_i \geq \sum_{k \neq i} a_{ik}m_k + b_i$$

ce qui montre que $\boldsymbol{m}$ est solution de C_i.

Si C_i est de type (2), on a, pour tout $\boldsymbol{x} \in$ E :

$$0 \geq \sum_{1 \leq j \leq n} a_{ij}x_j + b_i \geq \sum_{1 \leq j \leq n} a_{ij}m_j + b_i$$

ce qui montre que $\boldsymbol{m}$ est encore solution de C_i.

$\boldsymbol{m}$ satisfait donc à toutes les contraintes définissant E : $\boldsymbol{m} \in$ E.

Comme d'autre part $\boldsymbol{m}$ est un minorant de E, c'est la plus petite solution de E : $\boldsymbol{m}$ = min E. ■

Remarque La preuve, constructive, démontre que, sur les axes de l'espace : *les minimums des projections sont les projections du minimum global*.

$$\boldsymbol{m} = \min \mathrm{E} \quad \Leftrightarrow \quad m_j = \min \mathrm{proj}_j \mathrm{E},\ \forall j \in \{1, \ldots, n\}$$

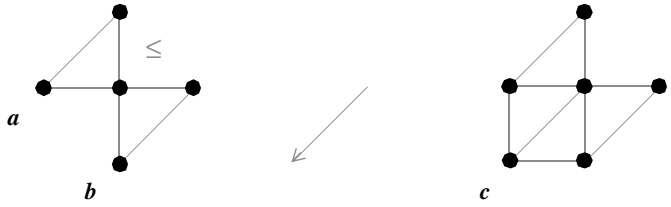
Corollaire *L'ensemble des solutions d'un système monotone possède, s'il est non vide et minoré, un plus petit élément.*

Soit $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ un système monotone et $\boldsymbol{c}$ un minorant de l'ensemble de ses solutions. Par le changement de variable $\boldsymbol{x} = \boldsymbol{y} + \boldsymbol{c}$, on déduit aisément :

$$\begin{aligned}\min \{\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\} &= \min \{\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{c}\} \\ &= \boldsymbol{c} + \min \{\boldsymbol{y} \geq 0 \mid \mathbf{A}\boldsymbol{y} \geq (\boldsymbol{b} - \mathbf{A}\boldsymbol{c})\}.\end{aligned}$$ ■

On a, bien entendu, un résultat symétrique en remplaçant ≥ par ≤, minoré par majoré et «plus petit» par «plus grand» élément.

Exemple Les diagrammes de Hasse suivants (dont les arêtes suivent la relation d'ordre produit) représentent, dans le plan discret, deux ensembles finis de points. L'un (à gauche) n'a pas de plus petit élément. L'autre (à droite) est un demi-treillis inférieur.



$\boldsymbol{a}$ et $\boldsymbol{b}$ sont deux éléments minimaux (incomparables pour l'ordre produit), alors que $\boldsymbol{c}$ = (0, 0) est un minimum global :

$\boldsymbol{c} = \min \{(x, y) \in \mathbb{Z}^2 \mid -x - y \geq -3,\ x - y \geq -1,\ -x + y \geq -1,\ x \geq 0,\ y \geq 0\}$.

4 Compléments sur les systèmes à variables réelles

4.1 Stabilité par projection

Les systèmes monotones forment une classe de problèmes linéaires stable par élimination de Fourier.

Propriété *Soient* S *un système monotone sur* $\mathbb{R}^n$ *et* x_0 *une variable de* S. *Alors* $\mathrm{E}_F(\mathrm{S}, x_0)$ *est un système monotone.*

Démonstration On supposera les inégalités orientées dans le sens $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ (l'autre sens s'obtenant par changement de variable $\boldsymbol{x'} = -\boldsymbol{x}$).

Soit (C_1, C_2) une paire d'inégalités complémentaires pour x_0. On a pour C_1 :

$$a_{10}x_0 \geq \sum_{k \neq 0} a_{1k}x_k + b_1$$

Si C_2 est de forme (1), on a $C_2 : a_{2j}x_j \geq a_{20}x_0 + \sum_{k \neq i,\, k \neq 0} a_{2k}x_k + b_2$

et $a_{20}C_1 + a_{10}C_2$:

$$(a_{10}a_{2j} - a_{20}a_{1j})x_j \geq \sum_{k \neq i,\, k \neq 0} (a_{20}a_{1k} + a_{10}a_{2k})x_k + (a_{20}b_1 + a_{10}b_2)$$

Si C_2 est de forme (2), on a $C_2 : 0 \geq a_{20}x_0 + \sum_{k \neq 0} a_{2k}x_k + b_2$

et $a_{20}C_1 + a_{10}C_2 : 0 \geq \sum_{k \neq 0} (a_{10}a_{2k} + a_{20}a_{1k})x_k + (a_{20}b_1 + a_{10}b_2)$

Dans les deux cas, les coefficients $a_{10}a_{2k} + a_{20}a_{1k}$ des variables apparaissant à droite de l'inégalité sont positifs. Seul le premier cas peut donner un coefficient positif du côté gauche de l'inégalité (si $a_{10}a_{2j} - a_{20}a_{1j} \geq 0$).

L'élimination de la variable réelle x_0 produit donc un système $\mathbf{A'x'} \geq \boldsymbol{b'}$ dans lequel chaque ligne de $\mathbf{A'}$ a au plus un coefficient positif. ■

Les propriétés géométriques de l'ensemble des solutions d'un système monotone sont donc préservées par toute opération de projection[1].

4.2 Caractérisation des polyèdres ayant un plus petit élément

Les conditions d'existence d'un plus petit élément sont étroitement liée au théorème de la dualité. L'énoncé suivant est dû à Cottle et Veinott (1972).

Théorème III.2 *Soit* $\mathrm{P} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$ *un polyèdre de* $\mathbb{R}^n$ *dont* $\boldsymbol{m}$ *est élément. Alors* $\boldsymbol{m}$ *est le plus petit élément de* P *si et seulement si la matrice* $\mathbf{A}$ *a pour inverse à gauche une matrice positive* $\mathbf{V}$ *et si* $\boldsymbol{m} = \mathbf{V}\boldsymbol{b}$.

Démonstration

Si $\boldsymbol{m}$ est le plus petit élément de P, alors, $\forall \boldsymbol{x} \in \mathrm{P}$, $\boldsymbol{x} \geq \boldsymbol{m}$ et pour tout vecteur $\boldsymbol{c} \geq 0$, on aura : $\boldsymbol{cx} \geq \boldsymbol{cm}$. Donc, $\forall \boldsymbol{c} \in \mathbb{R}^{+n}$, $\boldsymbol{cm}$ est solution optimale du problème linéaire $\min\{\boldsymbol{cx} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$.

Prenons pour $\boldsymbol{c}$ le $j^{\text{ième}}$ vecteur de la base canonique de $\mathbb{R}^n$: $\boldsymbol{c} = \boldsymbol{e}_j$. Le problème dual s'écrit alors : $\max\{\boldsymbol{by} \mid {}^{\mathrm{t}}\mathbf{A}\boldsymbol{y} = \boldsymbol{e}_j, \boldsymbol{y} \geq 0\}$ et admet une solution optimale $\boldsymbol{z}^j$ vérifiant $\boldsymbol{bz}^j = \boldsymbol{e}_j\boldsymbol{m}$.

En faisant varier j de 1 à n, on obtient ainsi n vecteurs $\boldsymbol{z}^j \geq 0$ vérifiant ${}^{\mathrm{t}}\mathbf{A}\boldsymbol{z}^j = \boldsymbol{e}_j$ et $\boldsymbol{bz}^j = m_j$. Posons alors ${}^{\mathrm{t}}\mathbf{V} = (\boldsymbol{z}^1, \ldots, \boldsymbol{z}^n)$. $\mathbf{V}$ est une matrice positive.

1. Ce point est évident pour un treillis.

On a, d'une part : ${}^t(\mathbf{VA}) = {}^t\mathbf{A}\,{}^t\mathbf{V} = ({}^t\mathbf{A}\boldsymbol{z}^1, \ldots, {}^t\mathbf{A}\boldsymbol{z}^n) = (\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n) = \mathbf{I}$
et d'autre part : ${}^t(\mathbf{V}\boldsymbol{b}) = {}^t\boldsymbol{b}\,{}^t\mathbf{V} = (\boldsymbol{b}\boldsymbol{z}^1, \ldots, \boldsymbol{b}\boldsymbol{z}^n) = (m_1, \ldots, m_n) = {}^t\boldsymbol{m}$.
Autrement dit, $\mathbf{VA} = \mathbf{I}$ et $\boldsymbol{m} = \mathbf{V}\boldsymbol{b}$.

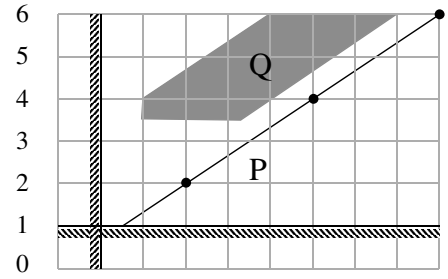
Ce qui prouve en particulier que $\mathbf{A}$ est inversible à gauche.

Réciproquement, si $\mathbf{V}$ est positive et vérifie $\mathbf{VA} = \mathbf{I}$, et si $\boldsymbol{m} = \mathbf{V}\boldsymbol{b} \in \mathrm{P}$, alors, $\forall \boldsymbol{x} \in \mathrm{P}$, on déduit de $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ en multipliant chacun des membres par $\mathbf{V} \geq 0$ $\mathbf{VA}\boldsymbol{x} \geq \mathbf{V}\boldsymbol{b}$, soit $\boldsymbol{x} \geq \boldsymbol{m}$.

Ce qui prouve que $\boldsymbol{m}$ est le plus petit élément de P. ■

Définition Soient $\mathrm{P}' = \{\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} \geq 0\}$ et $\mathrm{P} = \{\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$ ($\boldsymbol{b} \in \mathbb{Q}^m$) deux polyèdres. P étant défini comme l'intersection d'hyperplans parallèles à ceux de P', on dit que P et P' sont *parallèles*.

Exemple La figure suivante représente la demi-droite $\mathrm{P} = \{(x, y) \mid 6y = 4x, x \geq 1, y \geq 1\}$ (où l'on a mis en évidence les points à coordonnées entières) et un polyèdre Q parallèle à P dont le plus petit élément se situe au coin inférieur gauche de la surface grisée.



Le fait qu'un polyèdre $\{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{c}\}$ soit défini par un système monotone est une *condition suffisante* pour qu'il ait un plus petit élément — quand il n'est pas vide.

Réciproquement, Cottle et Veinott démontrent que la caractérisation précédente de $\mathbf{A}$ est une *condition nécessaire* pour que tous les polyèdres non vides parallèles au cône $\{\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} \geq 0\}$ aient dans $\mathbb{R}^{+n}$ un plus petit élément :

$\forall\, \boldsymbol{b} \in \mathbb{R}^m$, $\{\boldsymbol{x} \in \mathbb{R}^{+n} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$ a un plus petit élément lorsqu'il est non vide

$\Leftrightarrow$

$\mathbf{A}$ a au plus un coefficient strictement positif par ligne

Un théorème de Veinott (1989) montre encore que tout polyèdre formant un demi-treillis de $(\mathbb{R}^n, \geq)$ représente l'ensemble des solutions d'un système monotone[1].

1. La terminologie n'est pas celle de Veinott.

4.3 Références terminologiques

En remontant aux «sources», on trouve dans Varga (1962) la définition suivante (p. 87) :

Définition Une matrice $n \times n$ **A** est *monotone* (Collatz[1] 1952) si pour tout vecteur $\boldsymbol{r}$, $\mathbf{A}\boldsymbol{r} \geq 0$ implique $\boldsymbol{r} \geq 0$.

On trouve dans Berman & Plemmons (1979) une définition étendue aux matrices réelles $m \times n$ (p. 119) :

Définition Une matrice rectangulaire **A** est *monotone* si elle possède un inverse à gauche qui est une matrice positive.
(**A** *is* rectangular monotone *if it has a nonnegative left inverse*.)

5 Systèmes 1-monotones

Qu'ils soient à variables réelles, entières ou mixtes, les systèmes d'inégalités monotones ont des propriétés remarquables lorsque les coefficients positifs de la matrice du système ont pour valeur l'unité.

Définition Un système monotone $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ est *1-monotone* si **A** et $\boldsymbol{b}$ sont à coefficients entiers et si les coefficients strictement positifs de **A** sont tous égaux à 1.

Les contraintes du système sont donc de la forme (avec $a_{ij}, a_{ik} \in \mathbb{N}$ et $b_i \in \mathbb{Z}$) :

$$(1) \quad x_j - \sum_{k \neq i} a_{ik} x_k \geq b_i \quad \text{ou} \quad (2) \quad -\sum_{1 \leq i \leq n} a_{ij} x_j \geq b_i$$

Exemple Nous reproduisons ci-dessous une liste d'inégalités tirée d'une classification de problèmes combinatoires en variables bivalentes : $\boldsymbol{x} \in \{0,1\}^n$. Dans nos notations, tous les coefficients a_{ij} sont positifs (ce sont les valeurs absolues des coefficients de **A**). Toutes les inégalités présentées sont monotones, et pour les premières, 1-monotones. Lorsqu'elles sont de même sens, elles se combinent en systèmes monotones.

Un problème de couverture d'ensemble, $\mathbf{A}\boldsymbol{x} \geq \mathbf{1}$ (*set covering*), consiste ainsi à sélectionner une collection de sous-ensembles de $\{1,..., m\}$ dont l'union recouvre $\{1,..., m\}$. Chacune des n colonnes de A (faite de 0 et de 1) peut être vue comme l'un des sous-ensembles candidats pouvant entrer dans cette sélection.
Pour le problème de remplissage, $\mathbf{A}\boldsymbol{x} \leq \mathbf{1}$ (*set packing*), les sous-ensembles sélectionnés doivent être mutuellement disjoints.

1. L. Collatz. *Aufgaben monotoner Art*. Anch. Math. 3, 366-376, 1952.

Classe de contraintes	Expression	Forme normalisée	Sens	1-m
Knapsack	$\sum_i a_{ij}x_j \le b_i$	$-\sum_i a_{ij}x_j \ge -b_i$	inf	oui
Invariant Knapsack (Knapsack avec $a_{ij} = 1$)	$\sum_i x_j \le b_i$	$-\sum_i x_j \ge -b_i$	inf	oui
Set Packing (IK avec $b_i = 1$)	$\sum_i x_j \le 1$	$-\sum_i x_j \ge -1$	inf	oui
Set Covering	$\sum_i x_j \ge 1$	$-\sum_i x_j \le -1$	sup	oui
Bin Packing	$\sum_{i\neq k} a_{ij}x_j \le a_{ik}x_k$	$a_{ik}x_k - \sum_{i\neq k} a_{ij}x_j \ge 0$	inf	non
Facility location (BP avec $a_{ij} = 1$)	$\sum_{i\neq k} x_j \le a_{ik}x_k$	$a_{ik}x_k - \sum_{i\neq k} x_j \ge 0$	inf	non

Remarque Le système formé par les encadrements $0 \le x_j \le 1$ (1-monotones) et l'équation de partition d'ensemble $\sum_j x_j = 1$ regroupant deux inégalités 1-monotones mais de sens différents (Set Covering + Set Packing) n'est pas monotone.

5.1 Stabilité par projection

La classe des systèmes d'inégalités 1-monotones est stable par élimination de Fourier. De plus, la procédure d'élimination standard peut s'appliquer aux variables entières.

Théorème III.3 *Soient* S *un système 1-monotone et* x_0 *une variable entière de* S. *Alors l'élimination de Fourier de* x_0 *est licite et le système* $E_F(S, x_0)$ *est à son tour 1-monotone.*

Démonstration Considérons un système 1-monotone de la forme $\mathbf{A}\boldsymbol{x} \ge \boldsymbol{b}$, dans lequel x_0 désigne une des coordonnées de $\boldsymbol{x}$.
Soit (C_1, C_2) une paire d'inégalités complémentaires pour x_0. On a :

$C_1 : \quad x_0 \ge \sum_{k\neq 0} a_{1k}x_k + b_1$

et $C_2 : x_j \ge a_{20}x_0 + \sum_{k\neq i.\, k\neq 0} a_{2k}x_k + b_2$ si C_2 est de forme (1)

ou $C_2 : \ 0 \ge a_{20}x_0 + \sum_{k\neq 0} a_{2k}x_k + b_2$ si C_2 est de forme (2).

Dans les deux cas, on élimine x_0 par la combinaison linéaire $a_{20}C_1 + C_2$.

La caractère licite de l'élimination vient du coefficient 1 pour C_2.

Ce dernier prévient également toute croissance des coefficients positifs de la matrice du système dans les contraintes de forme (1).

Le seul coefficient pouvant rester positif dans $a_{20}C_1 + C_2$ est en effet celui de x_j qui devient, dans l'inégalité résultante, $1 - a_{20}a_{1j} \leq 1$. ■

Remarque Comme a_{20} est un entier plus grand que 1, $a_{1j} = 0$ est une condition nécessaire pour que le coefficient de x_j reste positif et égal à 1.

Corollaire *La relaxation linéaire continue des systèmes 1-monotones en nombres entiers est complète. Pour qu'il existe une solution entière, il suffit qu'il existe une solution réelle.*

Si $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ est un système 1-monotone :

$$\{\boldsymbol{x} \in \mathbb{Z}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\} = \varnothing \quad \Leftrightarrow \quad \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\} = \varnothing$$

La résolution des systèmes linéaires étant de complexité polynômiale, les systèmes 1-monotones en nombres entiers forment une *classe polynômiale*.

5.2 Intégrité des extrêmums globaux

Autre caractéristique essentielle des systèmes 1-monotones : les extrêmums globaux des solutions réelles et entières coïncident — lorsqu'ils existent.

Théorème III.4 *Soit $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ un système 1-monotone. Si le polyèdre* $\mathrm{P} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$ *est non vide et minoré, alors il possède un plus petit élément à coordonnées entières. Plus précisément :* $\min \mathrm{P} = \min \mathrm{Q}$, *avec* $\mathrm{Q} = \{\boldsymbol{x} \in \mathbb{Z}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$.

Démonstration Si P est minoré, P possède un plus petit élement, d'après le théorème **III.1**.

D'après le théorème **III.3** d'autre part, les projections de P sur chacun des axes de $\mathbb{R}^n$ sont définies par des systèmes 1-monotones. Toutes les bornes inférieures provenant d'inégalités de la forme $x_j \geq$ *constante entière*, on en déduit : $\forall j \in \{1, \ldots, n\}$, $\min \mathrm{proj}_j \mathrm{P} \in \mathbb{Z}$.

Par conséquent, $\min \mathrm{P} \in \mathbb{Z}^n$ et $\min \mathrm{P} = \min \mathrm{Q}$. ■

Autrement dit, les projections de P et Q qui se superposent ($\mathrm{proj}_j \mathrm{Q} \subset \mathrm{proj}_j \mathrm{P}$) se calent ici sur la même borne inférieure ($\min \mathrm{proj}_j \mathrm{P} = \min \mathrm{proj}_j \mathrm{Q}$).

Symétriquement, on aura : $\max \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}\} \in \mathbb{Z}^n$.

Remarque Dans les conditions du théorème, le polyèdre P des solutions réelles ne définit pas l'enveloppe convexe des solutions entières. Il en précise néanmoins un sommet remarquable.

5.3 Stabilité par troncature

Les systèmes 1-monotones définissent sur $\mathbb{R}^n$ des polyèdres dont les solutions ont la propriété de pouvoir être arrondies à leur partie entière.

Définition $\forall\, x \in \mathbb{R}, \exists!\, z \in \mathbb{Z}$ tel que $z \leq x < z+1$: z définit la *partie entière* de x, $\lfloor x \rfloor$.

Si $\boldsymbol{x} \in \mathbb{R}^n$, on note encore $\lfloor \boldsymbol{x} \rfloor$ le vecteur de $\mathbb{Z}^n$ dont les composantes sont les parties entières des composantes de $\boldsymbol{x}$.

Propriété $\lfloor \boldsymbol{x} \rfloor = \max \{\boldsymbol{y} \in \mathbb{Z}^n \mid \boldsymbol{y} \leq \boldsymbol{x}\}$

En effet, pour tout x_i, $\lfloor x_i \rfloor$ est le plus grand entier inférieur à x_i. ■

Théorème III.5 *Soit* $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ *un système 1-monotone et* $\mathrm{P} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$. *Si* $\boldsymbol{x} \in \mathrm{P}$ *alors* $\lfloor \boldsymbol{x} \rfloor \in \mathrm{P}$.

Démonstration Soit $\boldsymbol{y}$ une solution réelle du système. $\boldsymbol{y}$ est aussi solution de $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \geq \lfloor \boldsymbol{y} \rfloor$.

Le polyèdre $\mathrm{Q} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \geq \lfloor \boldsymbol{y} \rfloor\}$ est donc non vide ($\boldsymbol{y} \in \mathrm{Q}$) et minoré. Comme il est défini par un système 1-monotone, il admet un plus petit élément : min Q.

Du double encadrement : $\boldsymbol{y} \geq \min \mathrm{Q} \geq \lfloor \boldsymbol{y} \rfloor$ et du fait que min Q est à coordonnées entières, on déduit l'égalité : $\min \mathrm{Q} = \lfloor \boldsymbol{y} \rfloor$.

Ceci prouve que $\lfloor \boldsymbol{y} \rfloor$ appartient à Q, donc à P. ■

5.4 Caractérisations

De façon générale, l'analyse de Fourier indique que lorsqu'une variable x d'un système linéaire n'apparaît qu'avec des coefficients négatifs, l'ensemble des solutions, s'il est non vide, est non borné dans la direction des x décroissants. On en tire une condition nécessaire pour que ce polyèdre soit borné inférieurement : que la matrice du système présente au moins un coefficient positif dans chaque colonne.
Pour les systèmes 1-monotones, ce critère est très insuffisant : il faut au moins exiger la donnée explicite d'un minorant pour l'une des variable (ce qui veut dire, matriciellement, une ligne nulle à l'exception d'un coefficient 1).

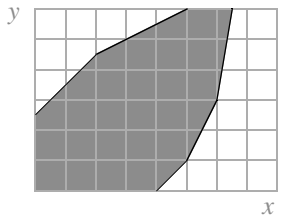
Nous observerons au chapitre VI, avec les systèmes de contraintes de tensions $\mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}$, qu'il suffit d'ajouter une inégalité $x_0 \leq 0$ (potentiel à l'origine) pour borner d'un seul coup l'ensemble des variables du système.

Propriété *Soit un* $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ *système 1-monotone. Si le système est soluble et ne contient pas d'encadrement de la forme* $x_j \geq$ *constante, alors aucune des valeurs des* x_j *n'est bornée inférieurement.*

Démonstration D'après les hypothèses, chaque contrainte de forme (1) met en relation au moins deux variables : $x_j \geq a_k x_k + \sum_{l \neq i,\, k} a_l x_l + b$ avec $k \neq j$ et $a_k \geq 1$.

$\forall\, \lambda > 0$, on a $0 \geq -\lambda \geq -a_l\lambda$ et $\sum_{l \neq i,\, k} a_l x_l \geq \sum_{l \neq i,\, k} a_l (x_l - \lambda)$. Donc, si $\boldsymbol{x}$ est solution, $(\boldsymbol{x} - \lambda \mathbf{1})$ est solution : $x_j - \lambda \geq a_k(x_k - \lambda) + \sum_{l \neq i,\, k} a_l (x_l - \lambda) + b$. ■

Dans le plan $\mathbb{R}^2$, les systèmes 1-monotones définissent des polygones dont toutes les limites basses, en dehors des contraintes d'encadrement sur *y*, sont des droites $y = ax + b$ de pente supérieure à 1 (*a* entier positif non nul) et toutes les limites hautes sont des droites de pente inférieure à 1.



Dans l'espace, les systèmes d'inégalités 1-monotones caractérisent des polyèdres qui, lorsqu'ils sont non vides et minorés, ont un plus petit élément dans $\mathbb{Z}^n$.

Un théorème de Chandrasekaran montre qu'il s'agit encore une fois de la classe la plus générale présentant cette propriété.

On peut l'énoncer comme suit :

Soit $\mathbf{A}$ *une matrice* $m \times n$ *à coefficients entiers. Définissons pour* $\boldsymbol{b} \in \mathbb{Z}^m$ *l'ensemble* $\mathrm{P}_{\boldsymbol{b}} = \{\boldsymbol{x} \in \mathbb{R}^{+n} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$. *Alors une condition nécessaire pour que tout polyèdre non vide de la famille des polyèdres* $(\mathrm{P}_{\boldsymbol{b}})_{\boldsymbol{b}}$ *possède un plus petit élément à coordonnées entières est que chaque ligne de* $\mathbf{A}$ *ait au plus un coefficient positif et que tous les coefficients positifs de* $\mathbf{A}$ *soient égaux à 1.*

Sa preuve repose sur le théorème de Cottle & Veinott[1].

1. D'après ce théorème en effet, chaque ligne de $\mathbf{A}$ possède au plus un élément strictement positif. Supposons que l'un de ces éléments, disons a_{ij}, ne soit pas égal à 1 et définissons $\boldsymbol{b}$ comme suit : $\boldsymbol{b} = \left\lfloor \mathbf{A}\frac{\boldsymbol{e}_j}{a_{ij}} \right\rfloor$ ($\mathbf{A}\boldsymbol{e}_j$ désigne la *j*ième colonne de $\mathbf{A}$, que l'on divise par $a_{ij} > 0$).
Alors $\boldsymbol{x}^* = a_{ij}^{-1}\boldsymbol{e}_j$ est un vecteur positif vérifiant $\mathbf{A}\boldsymbol{x}^* \geq \boldsymbol{b}$ qui appartient donc à $\mathrm{P}_{\boldsymbol{b}}$.
Or, si $\boldsymbol{x} \in \mathrm{P}_{\boldsymbol{b}}$, d'après la *i*ième inégalité, $a_{ij}x_j \geq \sum_{k \neq j} a_{ik}x_k + b_i \geq b_i = 1$. On en déduit que $x_j \geq x_j^*$ et, $\boldsymbol{x}$ étant positif, que $\boldsymbol{x} \geq \boldsymbol{x}^*.\boldsymbol{x}^*$ est ainsi le plus petit élément de $\mathrm{P}_{\boldsymbol{b}}$.
Mais ceci est en contradiction avec l'hypothèse selon laquelle ce plus petit élément est à coordonnées entières lorsque $\boldsymbol{b}$ est entier.

6 Compléments sur les systèmes à variables entières

6.1 Optimisation linéaire sous contraintes 1-monotones

Comme nous l'avons montré, la satisfaisabilité d'un système 1-monotone en nombres entiers est de complexité polynômiale. Bien que cette question soit résolue par le calcul d'un extrêmum global dans le continu, le problème de l'optimisation d'une forme linéaire quelconque requiert plus d'attention.

En effet, on cherche à calculer, avec $Q = P \cap \mathbb{Z}^n$ avec $P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$:

$$\min \{\boldsymbol{cx} \mid \boldsymbol{x} \in Q\} = \min \{z \in \mathbb{Z} \mid z = \boldsymbol{cx}, \boldsymbol{x} \in Q\}$$

Le rajout de l'égalité linéaire $z = \boldsymbol{cx}$ (qui correspond à la définition de la fonction objectif) ruine en général la propriété de monotonie du système, faisant ressurgir un risque d'explosion combinatoire.

Si l'on choisit toutefois $\boldsymbol{c} \geq 0$, c'est-à-dire une fonction objectif croissante, alors :

$$\min \{\boldsymbol{cx} \mid \boldsymbol{x} \in Q\} = \min \{z \in \mathbb{Z} \mid z \geq \boldsymbol{cx}, \boldsymbol{x} \in Q\}$$

et le problème se ramène à la démontration de la satisfaisabilité du système $\{(\boldsymbol{x}, z) \in \mathbb{Z}^{n+1} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}, z - \boldsymbol{cx} \geq 0\}$, lequel est 1-monotone si $\boldsymbol{c}$ est entier.

Dans ce cas, $\min \{\boldsymbol{cx} \mid \boldsymbol{x} \in Q\} = \min \{\boldsymbol{cx} \mid \boldsymbol{x} \in P\}$ et le calcul reste polynomial.

Tempérons d'emblée une interprétation trop optimiste de ce dernier résultat. Dès lors que la matrice est un peu dense, ses coefficients sont majoritairement négatifs. Ce n'est donc pas le min, mais le max qu'on cherchera le plus souvent à calculer...

Exemple On doit partir en randonnée et remplir son sac à dos (*knapsack*). On a le choix des articles. La seule contrainte est de ne pas emporter plus d'articles que la capacité du sac ne le permet.
Dans cet exercice d'optimisation combinatoire bien connu, le minimum est trivialement zéro : le choix du paresseux est de ne rien emporter.

Le problème difficile est le calcul du maximum — qui est NP-complet. Rappelons-en la formulation linéaire classique en $\{0,1\}$:

$$\max \{\boldsymbol{cx} \mid \boldsymbol{x} \in Q\}$$
$$\text{avec } Q = \{\boldsymbol{x} \in \{0,1\}^n \mid -\boldsymbol{ax} \geq -b\} \text{ et } \boldsymbol{a}, b, \boldsymbol{c} \geq 0.$$

6.2 Résolution générale par un algorithme de troncatures

La relaxation linéaire n'est pas suffisante pour résoudre à elle seule les systèmes monotones en nombres entiers dont les coefficients positifs sont quelconques. Mais elle peut être utilisée dans une méthode de coupes particulièrement simple pour un problème borné. En prenant l'arrondi supérieur du minorant fourni par cette relaxation, on peut en effet resserrer les bornes du problème jusqu'à atteindre une solution entière ou prouver qu'il n'en n'existe pas.

Soient un système d'inégalités monotone $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ et deux vecteurs $\boldsymbol{c}$ et $\boldsymbol{d}$. Considérons le polyèdre $P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{c} \leq \boldsymbol{x} \leq \boldsymbol{d}\}$ et l'ensemble de ses solutions entières $Q = \{\boldsymbol{x} \in \mathbb{Z}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{c} \leq \boldsymbol{x} \leq \boldsymbol{d}\}$.

De deux choses l'une :

- ou bien P est non vide ($\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{c} \leq \boldsymbol{x} \leq \boldsymbol{d}$ est satisfaisable) :
  dans ce cas, min P existe du fait que P est monotone et borné.
  Si toutes les coordonnées de min P sont entières, on a : min P = min Q ;
  sinon, $\boldsymbol{c} \leq \min P < \lceil \min P \rceil$ et l'on ne change pas l'ensemble des solutions entières en remplaçant dans la définition de Q les inégalités $\boldsymbol{c} \leq \boldsymbol{x}$ par $\lceil \min P \rceil \leq \boldsymbol{x}$.
- ou bien P est vide (le système n'est pas satisfaisable) :
  dans ce cas, le système n'a pas de solution entière.

A partir du système monotone d'origine, S_0, on constitue ainsi une suite $(S_i)_i$ de problèmes monotones et de polyèdres en nombres entiers $(Q_i)_i$, avec $Q_i = P_i \cap \mathbb{Z}^n$ et $P_i = \mathrm{Sol}(S_i)$.
Tant que S_i est satisfaisable et que $\min P_i \neq \lceil \min P_i \rceil$, on définit :

$$S_{i+1} = S_i \cup \{\lceil \min P_i \rceil \leq \boldsymbol{x}\}$$

Le système S_{i+1} est monotone. Les seules contraintes rajoutées au système sont des troncatures. Tous les plans de coupe sont perpendiculaires aux axes.

La suite $(\lceil \min P_i \rceil)_i$ étant strictement croissante dans $\mathbb{Z}^n$ et bornée par $\boldsymbol{d}$, le processus se termine nécessairement.

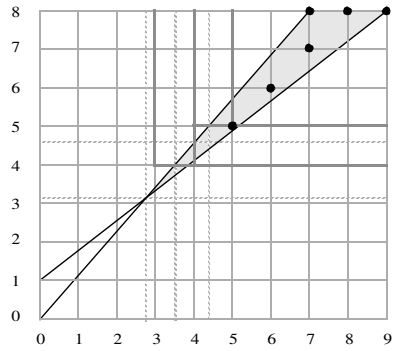
Les cas de terminaison sont au nombre de deux :

1. S_i n'a pas de solution réelle et donc pas de solution entière : $Q = \varnothing$.

2. $\min P_i = \lceil \min P_i \rceil$, d'où l'on déduit $Q \neq \varnothing$ et $\min Q = \min P_i$.

En notant $\| \boldsymbol{x} \|_1 = \sum_{j=1}^{n} |x_j|$, le nombre d'itérations est borné par la distance $\| \min Q - \lceil \min P \rceil \|_1$, donc par $\| d - c \|_1$. D'où une complexité polynômiale.

Exemple La figure suivante illustre la marche de l'algorithme dans le plan (x, y) pour $Q = \{(x, y) \in \mathbb{Z}^2 \mid 8x \geq 7y, 9y \geq 7x + 9\}$.



Le calcul relâché sur les réels donne min P = (63/23, 72/23) ≈ (2,74, 3,13). La progression de $\lceil \min P \rceil = (3, 4)$ à min Q = (5, 5) prend 3 itérations : (4, 4), (4, 5), (5, 5). Au total, on a besoin de calculer 4 relaxations linéaires et 4 troncatures.

Anticipons, pour conclure, sur le chapitre suivant et remplaçons la résolution linéaire de S_i par une résolution approchée avec les intervalles. Grâce à cette technique, on obtient — du fait de la monotonie — un bon encadrement du mininum de P_i .
Il suffit alors d'adjoindre à chacune des variables de S_i une contrainte d'intégrité (équivalente à $x = \lceil x \rceil$) pour effectuer les troncatures nécessaires. L'itération est automatique : min Q est obtenu sans énumération.

Cette propriété traduit la complétude de la résolution par les intervalles des systèmes monotones sur des domaines finis d'entiers.

Exemple Le PGCD de deux entiers $a \geq b > 0$ est le résultat d de la division exacte de b par la projection sur y de $\min\{(x, y) \in \mathbb{N}^2 \mid ay = bx,\ 1 \leq x \leq a,\ 1 \leq y \leq a\}$.

En voici une formulation en Prolog IV :

```
pgcd(A, B, D) :- A ge B, B ge 0,        % pré-conditions sur A et B
    pgcd_(A, B, D).

pgcd_(A, B, D) :-
    X ge 1, Y ge 1, int(X), int(Y),     % X le A, Y le A,
    A.*.Y = B.*.X,
    D ge 1, int(D),                     % D le B,
    B = D.*.Y,
    lub(D, D).                          % affecte y à son min. et d à son max.[1]
```

Quelques résultats d'exécutions :

```
>>  pgcd(1368250, 207025, D).
    D = 325.

>>  pgcd(56683, 12741, D).
    D = 1.
```

1. Lorsque les nombres entrés sont de grande taille, cette méta-instruction peut échouer du fait de la précision finie des calculs. Sans elle, on obtient toujours un encadrement du PGCD.

S'agissant de domaines finis, on peut reproduire l'expérience avec CHIP. Dans ce langage, le code devient :

```
pgcd_(A, B, D) :-
    D :: 1:B,                          % X, Y et D variables entières
    X :: 1:A, Y :: 1:A,                % A et B entiers donnés
    A*Y #= B*X,                        % (expression linéaire en X et Y)
    domain_info(Y,Y,_,_,_,_),          % unifie Y au min. du domaine
    B #= D*Y.                          % (expression non linéaire si Y est inconnu).
```

et pour de plus petits exemples :

```
?-  pgcd_(5668, 1274, D).
    D = 26                             1 call(s) delayed.
```

CHAPITRE IV

Résolution approchée des systèmes monotones

Techniques de réduction d'intervalles

> «La difficulté de résoudre exactement un grand nombre d'équations linéaires — comme c'est souvent le cas, quand on applique la méthode des moindres carrés — a conduit à penser à des méthodes d'approximation. Il en est une qui se présente d'elle-même quand, dans les différentes équations, ce n'est jamais la même variable qui est affectée d'un coefficient nettement supérieur aux autres.»
>
> C. JACOBI

Les méthodes formelles utilisées pour la résolution des systèmes linéaires exigent des calculs arithmétiques exacts, effectués en précision infinie. Or, au niveau logiciel, la manipulation de grands nombres s'avère coûteuse, en mémoire comme en temps.

L'utilisation d'une précision finie, limitée à quelques «mots» machine (comme celle des opérations en virgule flottante, implantées au niveau matériel) pose le problème traditionnel de l'arrondi des calculs «qui ne tombent pas juste». Une façon sûre de procéder consiste à remplacer le choix arbitraire d'un arrondi par un encadrement entre deux valeurs numériques. L'arithmétique des intervalles découle de cette préoccupation de correction des résultats. En systématisant ce calcul, elle reste économe en mémoire et en temps.

Dans cet esprit, la résolution de contraintes numériques se conçoit comme la preuve qu'une solution, si elle existe, se situe dans un intervalle restreint, ou, par contraposition, qu'il n'existe pas de solution en dehors de l'intervalle mono- ou multi-dimensionnel calculé.

La résolution approchée ne donne plus alors, en général, qu'une condition nécessaire de satisfaisabilité. Elle joue le rôle d'un filtre, qui ne perd pas de solution, mais peut laisser passer des systèmes insolubles. L'approximation obtenue, sous la forme d'un pavé (ou produit cartésien d'intervalles), est un sur-ensemble, assez grossier, des solutions possibles — ensemble que l'on peut éventuellement partitionner pour approfondir l'analyse.

Dans ce chapitre, nous présentons une théorie simplifiée de la résolution de contraintes avec les intervalles, en prenant Prolog IV (Colmerauer 1995) comme modèle de référence. Le fait de se cantonner aux systèmes linéaires — dont les projections exactes sont des intervalles — nous évite les problèmes de division par zéro qui surgissent dans le traitement des relations algébriques de degré supérieur.

Deux niveaux d'approximation se superposent dans la résolution. Le premier est lié à la discrétisation des axes et au maillage rectangulaire de l'espace qui s'ensuit. Le second est la conséquence d'une relaxation des conjonctions d'inégalités qui ignore les dépendances formelles entre variables au niveau d'un système. Si chaque contrainte linéaire est, en effet, très bien résolue localement, elle l'est presque indépendamment des autres. Seules les contraintes unaires — qui définissent les intervalles — sont prises en compte globalement. Elle répercutent les réductions successives opérées sur les domaines de variation des inconnues, lesquels, étant discrets, finissent par se stabiliser autour d'une maille plus ou moins large.

Nous démontrons qu'en dépit de cette coordination sommaire, le calcul d'approximation peut produire, avec des systèmes monotones, des encadrements assez fins — du côté où existe un extrêmum global. Ainsi :

1. la surestimation des bornes des projections peut être contrôlée au voisinage de ce point (et diminuée en augmentant la précision des calculs)
2. l'imbrication du pavé des intervalles réduits contre le polyèdre des solutions devient parfaite en ce point, pour les systèmes 1-monotones.

Ces résultats dispensent de compléter le filtrage par une phase d'énumération *ad hoc*.

1 Rappels sur les intervalles

Les intervalles sont les sous-ensembles convexes de l'espace ordonné $(\mathbb{R}, \geq)$.

Définition On appelle *intervalle* de $\mathbb{R}$ toute partie I de $\mathbb{R}$ vérifiant :

$$\forall\ (x, y) \in I^2 \quad \{z \in \mathbb{R} \mid x \leq z \leq y\} \subset I$$

Exemple L'ensemble vide, $\mathbb{R}$, tout point $\{x\}$ de $\mathbb{R}$ sont des intervalles.

Exemple L'ensemble $\{1,2,3\}$ n'est pas un intervalle de $\mathbb{R}$, mais pourrait être vu comme un intervalle de $\mathbb{N}$, dans la mesure où il contient tous les points compris entre toute paire de points. Nous réserverons dans la suite le mot intervalle pour les intervalles continus.

L'ensemble $\Im(\mathbb{R})$ des intervalles de $\mathbb{R}$ est stable par intersection.

Propriété *Soit $(I_k)_{k \in K}$ une famille (finie ou infinie) d'intervalles. Alors $\bigcap_{k \in K} I_k$ est un intervalle.*

Les notations classiques des intervalles se justifient par une extension topologique de la droite réelle aux deux infinis.

Définition On appelle *droite numérique achevée* l'ensemble $\mathbb{R} \cup \{-\infty, +\infty\}$ où $-\infty$ et $+\infty$ sont deux éléments distincts n'appartenant pas à $\mathbb{R}$.

La droite numérique achevée est totalement ordonnée par la relation $\leq$ qui prolonge l'ordre de $\mathbb{R}$ et vérifie : $\forall\, x \in \mathbb{R} \cup \{-\infty, +\infty\}$, $-\infty \leq x$ et $x \leq +\infty$.

Propriété *Toute partie de* $\mathbb{R} \cup \{-\infty, +\infty\}$ *admet une borne supérieure et une borne inférieure.*

En particulier, tout intervalle I de $\mathbb{R}$ admet une borne inférieure, $\inf(I)$, et une borne supérieure, $\sup(I)$, dans $\mathbb{R} \cup \{-\infty, +\infty\}$. On note ces bornes avec un crochet fermant ou ouvrant suivant qu'elles appartient ou non à l'intervalle.

Un intervalle non vide de $\mathbb{R}$ peut ainsi prendre l'une des 10 formes suivantes ($u, v \in \mathbb{R}$, $u < v$) :

Intervalle I	Inf I	Sup I	Min I	Max I	Définition
$[u, u]$	u	u	•	•	$\{u\}$
$[u, v]$	u	v	•	•	$\{x \in \mathbb{R} \mid u \leq x \leq v\}$
$[u, v[$	u	v	•		$\{x \in \mathbb{R} \mid u \leq x < v\}$
$]u, v]$	u	v		•	$\{x \in \mathbb{R} \mid u < x \leq v\}$
$]u, v[$	u	v			$\{x \in \mathbb{R} \mid u < x < v\}$
$[u, +\infty[$	u	$+\infty$	•		$\{x \in \mathbb{R} \mid u \leq x\}$
$]u, +\infty[$	u	$+\infty$			$\{x \in \mathbb{R} \mid u < x\ \}$
$]-\infty, v]$	$-\infty$	v		•	$\{x \in \mathbb{R} \mid x \leq v\}$
$]-\infty, v[$	$-\infty$	v			$\{x \in \mathbb{R} \mid x < v\}$
$]-\infty, +\infty[$	$-\infty$	$+\infty$			$\mathbb{R}$

Pour l'intervalle vide, on a $\inf(\varnothing) = +\infty$, $\sup(\varnothing) = -\infty$, ce qui revient à écrire : $\varnothing =]+\infty, -\infty[$.

$[u, v]$, $[u, v[$, $]u, v]$, $]u, v[$ représentent des segments, respectivement fermés, semi-fermés ou semi-ouverts et ouverts. $[u, +\infty[$ et $]-\infty, v]$ représentent des demi-droites fermées[1], $]u, +\infty[$ et $]-\infty, v[$, des demi-droites ouvertes.

Calculer avec des intervalles, c'est calculer avec leurs bornes. Pour cela, il faut étendre les opérations arithmétiques usuelles à certaines combinaisons incluant $+\infty$ ou $-\infty$.

Pour tout x de $\mathbb{R}$, on convient de poser : $x + (+\infty) = +\infty$ et $x + (-\infty) = -\infty$.
Pour tout $x > 0$, on convient également que : $x \times (+\infty) = +\infty$ et $x \times (-\infty) = -\infty$.
On définit enfin : $(-\infty) + (-\infty) = -\infty$, $(-\infty) - (+\infty) = -\infty$ et $\lfloor -\infty \rfloor = -\infty$.

1. Un intervalle est *fermé* si sa définition ne comprend que des inégalités larges.

2 Discrétisation de la droite et de l'espace

La discrétisation substitue aux calculs exacts sur la droite «continue» des calculs approchés sur une droite graduée ou «discrète». Les graduations de cette règle fournissent les arrondis, par défaut et par excès, pour encadrer les résultats d'un calcul.

Pour être homogènes, les opérations arithmétiques sur les réels doivent s'étendre aux ensembles encadrant les données et devenir des opérations sur des intervalles.

Définition $D \subset \mathbb{R} \cup \{-\infty, +\infty\}$ est un *ensemble discret* si $\forall (x, y) \in \mathbb{R}^2$, $[x, y] \cap D$ est fini.

Autrement dit, D est un ensemble discret si entre toute paire de points, il y a un nombre fini de points de D.

Exemple Toute partie finie de $\mathbb{R}$ est discrète.

Exemple L'ensemble $10^{-q}\mathbb{Z} = \{x \in \mathbb{R} \mid 10^q x \in \mathbb{Z}\}$ des nombres rationnels ayant au plus q décimales est discret.

Définition Une *base d'approximation* pour le calcul des intervalles est un sous-ensemble de $\mathbb{I}(\mathbb{R})$ contenant $\emptyset$ et $\mathbb{R}$, stable par intersection.

Remarque Cette définition peut être généralisée aux sous-ensembles de l'ensemble des parties de $\mathbb{R}$ afin de prendre en compte l'approximation par des unions d'intervalles (qui constitue un mode alternatif en Prolog IV).
Les intervalles simples suffiront ici à notre propos.

Définition Un *espace d'approximation* est un couple $(\mathbb{R}, \mathbb{I})$ où $\mathbb{I}$ est une base d'approximation. Cet espace est *fini* si $\mathbb{I}$ est fini.

Exemple L'ensemble $\mathbb{I}_\mathbf{F}$ des intervalles de $\mathbb{R}$ dont les bornes réelles sont des nombres «flottants»[1] est un ensemble fini contenant $\emptyset$ et $\mathbb{R}$, stable par intersection. $(\mathbb{R}, \mathbb{I}_\mathbf{F})$ est l'espace d'approximation retenu pour l'arithmétique réelle de Prolog IV.

Ce moyen de construire des espaces d'approximation peut être généralisé.

Définition On dira que l'espace $(\mathbb{R}, \mathbb{I})$ est (*finement*) *discrétisé* si sa base $\mathbb{I}$ est définie comme l'ensemble des intervalles de $\mathbb{I}(\mathbb{R})$ dont les bornes inf. et sup. appartiennent à $D \cup \{-\infty, +\infty\}$, où D est un ensemble discret donné.

Nous noterons $\mathbb{I}_\mathbf{Z}$ l'ensemble des intervalles de $\mathbb{R}$ dont les bornes sont entières lorsqu'elles sont finies et $\mathbb{I}_M$ le sous-ensemble fini de $\mathbb{I}_\mathbf{Z}$ dont les bornes entières sont inférieures en valeur absolue à un nombre $M \gg 1$.

1. Il s'agit, plus précisément, des nombres rationnels définis par la norme IEEE 754 pour la précision simple (codés sur 32 bits).

Exemple Le sous-ensemble des intervalles fermés de $\mathfrak{I}(\mathbb{R})$ comprenant $\emptyset$, $\mathbb{R}$ et les segments $[u, v]$ où u et v sont entiers ($u \leq v$) est un ensemble infini, stable par intersection. L'espace d'approximation qu'il forme avec $\mathbb{R}$ n'est pas, au sens de la définition précédente, finement discrétisé.
Il en est de même des espaces (finis) utilisés dans BNR-Prolog (Older & Vellino, 1990) ou Numerica (Van Hentenryck, Michel, Deville 1997) qui ne retiennent que les intervalles fermés de $\mathfrak{I}_{\mathbf{F}}$.

Un espace finement discrétisé permet de réaliser une partition de $\mathbb{R}$ en une suite d'*intervalles atomiques* de la forme :

$$..., \{n\},]n, n+1[, \{n+1\},]n+1, n+2[, \{n+2\}, ...$$

Cette suite est finie si D est fini, comme par exemple :

$$]-\infty, -M[, \{-M\},]-M, -M+1[, ... ,]M-1, M[, \{M\},]M, +\infty[$$

Dans ce cas, $|\mathfrak{I}_M|$ est de l'ordre de $2|\mathrm{D}|^2$ avec $|\mathrm{D}| \approx 2M$ et le nombre d'intervalles atomiques s'élève à peu près à $2|\mathrm{D}|$.

Remarque La distinction, dans une implémentation, entre intervalles ouverts et fermés remonte au papier de Cleary (1987) sur l'intégration de l'arithmétique des intervalles à la programmation en logique. Elle nous sera utile pour tester l'exactitude du calcul des projections d'un polyèdre convexe fermé de $\mathbb{R}^n$.

Si $(\mathbb{R}, \mathfrak{I})$ est un espace d'approximation, la discrétisation de chacun des axes de $\mathbb{R}^n$ suivant $\mathfrak{I}$ conduit à un maillage rectangulaire de l'espace.

Définition Une maille rectangulaire ou *pavé* de $\mathbb{R}^n$ est un produit cartésien $I_1 \times \ldots \times I_n$ d'intervalles de $\mathfrak{I}$. La maille est *atomique* si ces intervalles sont atomiques.

3 Approximation d'un ensemble de points

La donnée de $(\mathbb{R}, \mathfrak{I})$ permet d'encadrer tout point de $\mathbb{R}$ par un intervalle de $\mathfrak{I}$.

Lorsque, comme dans $(\mathbb{R}, \mathfrak{I}_{\mathbf{Z}})$, les graduations sont entières, les opérations d'arrondi correspondent aux fonctions parties entières.

Propriété *Si $x \in \mathbb{Z}$, on a : $\lfloor x \rfloor = \lceil x \rceil$ et $\{x\} \in \mathfrak{I}_{\mathbf{Z}}$; sinon, $\lfloor x \rfloor + 1 = \lceil x \rceil$ et $]\lfloor x \rfloor, \lceil x \rceil[$ définit le plus petit intervalle de $\mathfrak{I}_{\mathbf{Z}}$ encadrant x.*

On rappelle en effet que $\lfloor x \rfloor \leq x < \lfloor x \rfloor + 1$ et que $\lceil x \rceil - 1 < x \leq \lceil x \rceil$. ■

La notion d'encadrement dans $(\mathbb{R}, \mathfrak{I})$ se généralise à un ensemble de points.

Définition Soit E une partie de $\mathbb{R}$. L'intervalle $\bigcap \{I \in \mathfrak{I} \mid E \subset I\}$ est le plus petit intervalle de $\mathfrak{I}$ contenant E. On le note : $\mathrm{apx}_{\mathfrak{I}}(E)$.

Exemple $x = \frac{1}{2} \in \mathfrak{I}_{\mathbf{F}}$, par conséquent, $\mathrm{apx}_{\mathfrak{I}_{\mathbf{F}}}(\{x\}) = \{x\}$.

C'est le plus petit intervalle contenant x.

```
>> bounds(1/2, U, V).
    V = 1/2,
    U = 1/2.
```

$x = \frac{1}{3} \notin \mathfrak{I}_{\mathbf{F}}$: $\mathrm{apx}_{\mathfrak{I}_{\mathbf{F}}}(\{x\})$ est l'intervalle atomique ouvert $]u, v[$ contenant x.

```
>> bounds(1/3, U, V).
    V = 11184811/33554432,          % `>0.33333334`
    U = 5592405/16777216.           % `>0.3333333`
```

Propriété $\mathrm{apx}_{\mathfrak{I}}(I) = I \Leftrightarrow I \in \mathfrak{I}$.

On en déduit, en particulier :

- $\mathrm{apx}_{\mathfrak{I}}(\varnothing) = \varnothing$
- $\mathrm{apx}_{\mathfrak{I}}(\mathrm{apx}_{\mathfrak{I}}(E)) = \mathrm{apx}_{\mathfrak{I}}(E)$.

Si E est un sous-ensemble de $\mathbb{R}^n$, on définit de même $\mathrm{apx}_{\mathfrak{I}^n}(E)$ — que l'on notera pour simplifier $\mathrm{apx}(E)$ — comme le plus petit pavé de $\mathbb{R}^n$ contenant E :

$$\mathrm{apx}(E) = \bigcap \{\boldsymbol{I} \in \mathfrak{I}^n \mid E \subset \boldsymbol{I}\}$$

Propriété *On a par construction :*

$$\mathrm{apx}(E) = \mathrm{apx}(\mathrm{proj}_1(E)) \times \ldots \times \mathrm{apx}(\mathrm{proj}_n(E))$$

Autrement dit : $\mathrm{proj}_j(\mathrm{apx}_{\mathfrak{I}^n}(E)) = \mathrm{apx}_{\mathfrak{I}}(\mathrm{proj}_j(E))$.

L'approximation est un pavé qui est le produit cartésien de ses n projections. Chacune de ces projections contient la projection de E et donc sa plus petite approximation $\mathrm{apx}(\mathrm{proj}_j(E))$ à laquelle elle est, en définitive, égale. ■

Théorème IV.1 *Soit E un sous-ensemble de $\mathbb{R}^n$. Posons $I_1 \times \ldots \times I_n = \mathrm{apx}(E)$ dans un espace discrétisé. Alors, si l'intervalle I_j possède un minimum,* $\min(I_j) \in \mathrm{proj}_j(E)$. *Symétriquement, si l'intervalle I_j possède un maximum,* $\max(I_j) \in \mathrm{proj}_j(E)$.

Démonstration Supposons I_j de la forme $[u, u]$, $[u, v]$, $[u, v[$ ou $[u, +\infty[$ pour qu'il posséde un minimum u. Par définition, $I_j = \mathrm{apx}(\mathrm{proj}_j(E))$ et $\mathrm{proj}_j(E) \subset I_j$.

Si $u \notin \mathrm{proj}_j(E)$, alors $\mathrm{proj}_j(E) \subset I_j - \{u\}$. Or les hypothèses nous assurent que $I_j - \{u\}$ est un intervalle de $\mathfrak{I}$. Qu'il soit strictement contenu dans I_j et contienne la projection de E contredit la définition de I_j.
Donc $u \in \mathrm{proj}_j(E)$. ■

4 Mécanisme général de calcul

On se place dans un espace d'approximation ($\mathbb{R}$, $\Im$) fini[1]. S = $\bigcup_{i=1} C_i$ est un système de contraintes sur $\mathbb{R}^n$. Pour simplifier les notations, on se servira de la même lettre *C* pour désigner la contrainte élémentaire, la relation qu'elle définit ou encore le graphe de cette relation :

$$C = \{(x_1,\ldots, x_n) \in \mathbb{R}^n \mid C(x_1,\ldots, x_n)\}.$$

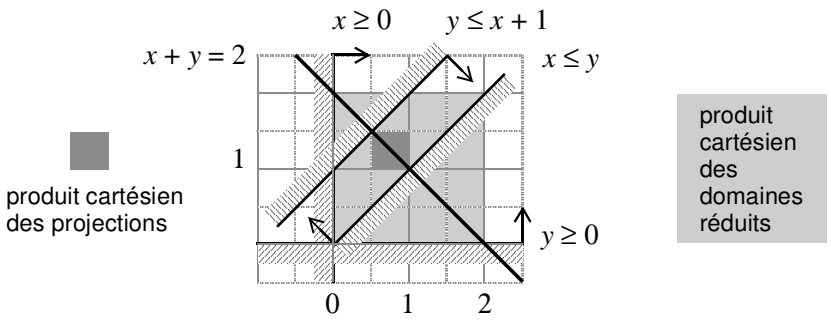
4.1 Résolution approchée par réduction d'intervalles

Résoudre S au sens des intervalles, c'est calculer une approximation de l'ensemble P des solutions de S par un pavé $\boldsymbol{X}$ de $\mathbb{R}^n$. Idéalement, ce devrait être le plus petit pavé contenant P : $\boldsymbol{X}$ = apx(P). Mais pour parvenir à un compromis précision - efficacité, on se contente d'un sur-encadrement :

$$P \subset apx(P) \subset \boldsymbol{X}.$$

Pour les systèmes linéaires[2], l'approximation des projections par le mécanisme de réduction d'intervalles détaillé dans ce qui suit est en général assez mauvaise, comme l'illustre le calcul suivant effectué en Prolog IV.

Exemple Soit $P = \{(x, y) \in \mathbb{R}^2 \mid x \geq 0, y \geq 0, x \leq y, y \leq x + 1, x + y = 2\}$. P correspond au segment de la droite $x + y = 2$ compris entre les points (1/2, 3/2) et (1,1). P est donc un polytope de dimension 1.



Les projections de P sont les intervalles fermés [1/2, 1] et [1, 3/2] (dont les bornes correspondent à des flottants).

Le mécanisme de réduction d'intervalles donne pour approximation de ces projections des intervalles, fermés également, mais beaucoup plus larges : [0, 2] et [0, 2].

```
>>  X ge 0, Y ge 0, X le Y, Y le X.+.1, X.+.Y = 2.
    Y ~ cc(0,2),
    X ~ cc(0,2).
```

1. Ces résultats se généralisent aux espaces d'approximation dans lesquels toute suite de $\Im$ strictement décroissante pour l'inclusion est finie.
2. Qui sont loin de constituer l'essentiel de ce pourquoi les techniques de réduction d'intervalles ont été introduites en Prolog IV.

4.2 Algorithme du plus grand point fixe

L'algorithme suivant correspond à l'une des formes pouvant être donnée à la vérification partielle de la cohérence d'un système de contraintes numériques. On le retrouve mentionné sous des noms divers : *bounds (2B-) consistency*, *hull consistency* ou encore *Partial Look Ahead Inference Rule*.

Le seul préalable qu'il pose est de savoir calculer, sur tout pavé $\boldsymbol{X}$ de $\mathbb{R}^n$, l'approximation $\boldsymbol{X'} = \text{apx}(C_i \cap \boldsymbol{X})$ de l'ensemble des solutions de chacune des contraintes C_i du système. Ce qui veut dire en particulier, savoir décider de la satisfaisabilité du sous-système : $C_i(\boldsymbol{x}) \wedge \boldsymbol{x} \in \boldsymbol{X}$.

Définition On note red_C l'*opérateur de réduction* associé à la contrainte C :

$$\text{red}_C(\boldsymbol{X}) = \text{apx}(C \cap \boldsymbol{X})$$

Ce principe de réduction d'intervalles s'applique à toutes sortes de contraintes élémentaires (Prolog IV en recense près de 100 dans $\mathbb{R}^3$).

Il permet de progresser dans la résolution d'un système en effectuant à tour de rôle les réductions permises par chacune des contraintes C_i, jusqu'à ce que les intervalles X_j attachés aux différentes variables se stabilisent.

Initialement, on se donne un encradrement des solutions de S par un pavé non vide $I_1 \times \ldots \times I_n$ qui est égal à $\mathbb{R}^n$ si rien n'est précisé ($\mathbb{R} \in \mathfrak{I}$).

Puis la simplification procède par réductions successives :

Tant que cela est possible, on sélectionne une contrainte C de S *et l'on transforme* $I_1 \times \ldots \times I_n$ *en un pavé* $J_1 \times \ldots \times J_n = \text{red}_C(I_1 \times \ldots \times I_n)$.

La réécriture $I_1 \times \ldots \times I_n \rightarrow \text{red}_C(I_1 \times \ldots \times I_n)$ ne compte que si $\text{red}_C(I_1 \times \ldots \times I_n)$ est strictement inclus dans $I_1 \times \ldots \times I_n$. On obtient ainsi une suite de pavés de $\mathfrak{I}^n$ strictement décroissante.
$\mathfrak{I}^n$ étant fini, la terminaison du processus est assurée.

Formellement, avec l'opérateur de réduction associé au système :

$$\text{red}_S(I_1 \times \ldots \times I_n) = \bigcap_{i=1} \text{red}_{C_i}(I_1 \times \ldots \times I_n)$$

l'algorithme s'écrit :

```
X  := ℝ^n ;
X' := red_S(X) ;
Tant que X' ≠ X faire
    début
    X  := Y ;
    X' := red_S(X)
    fin
```

A chaque itération, red_S applique m réductions en parallèle.

A l'issue d'une nombre fini d'itérations k, on obtient un pavé $\boldsymbol{X} = \text{red}_S^{(k)}(\mathbb{R}^n)$ vérifiant $\boldsymbol{X} = \text{red}_S(\boldsymbol{X})$, soit, pour toute contrainte C_i de S, $\boldsymbol{X} = \text{red}_{C_i}(\boldsymbol{X})$, c'est-à-dire :

$$\boldsymbol{X} = \text{apx}(C_i \cap \boldsymbol{X})$$

Le point fixe calculé correspond au plus grand élément de l'ensemble :

$$\{\boldsymbol{X} \in \mathbb{I}^n \mid \boldsymbol{X} = \text{apx}(C_1 \cap \boldsymbol{X}), \ldots, \boldsymbol{X} = \text{apx}(C_m \cap \boldsymbol{X})\}.$$

Ce plus grand point fixe définit l'approximation de la résolution de S par les intervalles de $\mathbb{I}$.

Propriété *Soit* P *l'ensemble des solutions d'un système de contraintes* S *sur* $\mathbb{R}^n$ *et* $\boldsymbol{X}$ *le pavé de* $\mathbb{R}^n$ *définissant l'approximation de la résolution de* S *par les intervalles de* $\mathbb{I}$. *Alors toutes les solutions* $\boldsymbol{x}$ *de* P *sont dans* $\boldsymbol{X}$. *En particulier, si* $\boldsymbol{X}$ *est vide,* P *l'est aussi.*

En effet, l'approximation est correcte puisque chacune des réécritures préserve l'ensemble des solutions (de la contrainte qui l'active et, *a fortiori*, du système). On ne perd donc pas de solution.

Comme $P \subset \boldsymbol{X}$, on obtient une condition suffisante d'insatisfaisabilité dès que le domaine d'une des variables se réduit à l'ensemble vide. ■

En définitive, le mécanisme de point fixe construit une approximation globale à partir d'un nombre fini d'approximations locales. Chacune des relations de base qui entre dans la définition du système contribue individuellement au filtrage des solutions. La résolution numérique s'effectue par élimination de valeurs — et non plus par élimination de variables.

- Si le problème exact a une solution, alors celle-ci se trouve dans le pavé des domaines réduits retournés par la résolution du problème approché.
- Si le problème approché est insoluble, alors le problème exact l'est aussi.

Définition Soit $\boldsymbol{X} = \text{red}_S^*(\mathbb{R}^n)$ le point fixe de l'approximation de S par les intervalles $\mathbb{I}$. Si $\boldsymbol{X}$ n'est pas vide, l'intervalle $X_j = \text{proj}_j(\boldsymbol{X})$ est appelé *domaine réduit* de la variable x_j.

La résolution approchée échoue lorsque le point fixe donne une maille vide. Dans les autres cas, les intervalles du pavé initial sont réduits jusqu'à ce que chaque face intersecte chaque contrainte :

$$\forall (i, j), \quad \text{proj}_j(C_i \cap \boldsymbol{X}) \neq \varnothing.$$

Remarque L'ensemble des solutions de l'équation implicite $\boldsymbol{X} = \text{apx}(C \cap \boldsymbol{X})$ définit une relation $\mathcal{C}$ sur $\mathbb{I}^n$ qui peut être vue comme une extension aux parties de $\mathbb{R}^n$ de la relation C définie sur $\mathbb{R}^n$. $\mathcal{C}$ est l'approximation «cartésienne» de C.

La résolution approchée échoue dès que le système de contraintes $\mathcal{S} = \bigcup_{i=1} \mathcal{C}_i$ devient satisfaisable sur $\mathfrak{I}^n$: la procédure de décision est donc complète, mais au niveau de l'approximation seulement.

5 Traitement des inégalités linéaires

Pour être à même de résoudre, avec le mécanisme précédent, un système d'inégalités linéaires, il reste à définir l'opérateur de réduction à associer à chacune des inégalités du système.

5.1 Réduction d'une inégalité linéaire

On cherche à approximer au mieux l'intersection d'un pavé $\boldsymbol{X}$ de $\mathfrak{I}^n$ et d'un demi-espace défini par une inégalité C sur $\mathbb{R}^n$.

La forme générale de C est : $\sum_{j \in J^+} a_j x_j \geq \sum_{j \in J^-} a_j x_j + b$ avec $a_j \geq 0, J^+ \cap J^- = \varnothing$ et $|J^+ \cup J^-| \leq n$.

Les contraintes de bornage résultant de la donnée du pavé $\boldsymbol{X}$ se traduisent par un système d'au plus $2n$ inégalités dans lequel les inégalités strictes donnent autant de contraintes de différence :

$$x_j \geq \inf(X_j) \qquad \sup(X_j) \geq x_j$$
$$x_j \neq \inf(X_j) \qquad \sup(X_j) \neq x_j$$

Lorsque les inégalités sont larges, on a $\inf(X_j) = \min(X_j)$ et $\sup(X_j) = \max(X_j)$.

L'élimination de Fourier d'une variable d'indice $k \in J^+$ de C remplace C par :

$a_k \sup(X_k) + \sum_{j \in J^+ - \{k\}} a_j x_j \geq \sum_{j \in J^-} a_j x_j + b$ (inégalité stricte si $\sup(X_k) \neq x_k$).

L'élimination de Fourier d'une variable d'indice $l \in J^-$ de C remplace C par :

$\sum_{j \in J^+} a_j x_j \geq \sum_{j \in J^- - \{k\}} a_j x_j + a_l \inf(X_l) + b$ (inégalité stricte si $x_l \neq \inf(X_l)$).

Les projections sur les axes sont donc définies par les contraintes : $x_j \in X_j$ et,

$$\text{si } j \in J^+ : a_j x_j + \sum_{k \in J^+ - \{i\}} a_k \sup(X_k) \geq \sum_{l \in J^-} a_l \inf(X_l) + b$$
$$\text{si } j \in J^- : \sum_{k \in J^+} a_k \sup(X_k) \geq \sum_{l \in J^- - \{j\}} a_l \inf(X_l) + a_j x_j + b$$

(ces contraintes devenant strictes dès lors que l'un des X_k n'a pas de max ou que l'un des X_l n'a pas de min).

On a donc : $\text{proj}_j(C \cap \boldsymbol{X}) = \boldsymbol{X} \cap \boldsymbol{Y}$, avec

si $j \in J^+$:

$\sup(Y_j) = +\infty$;

$$\inf(Y_j) = \frac{1}{a_j}\left(\sum_{l \in J^-} a_l \inf(X_l) - \sum_{k \in J^+ - \{j\}} a_k \sup(X_k) + b \right)$$

si $j \in J^-$:

$$\sup(Y_j) = \frac{1}{a_j}\left(\sum_{k \in J^+} a_k \sup(X_k) - \sum_{l \in J^- - \{j\}} a_l \inf(X_l) - b \right)$$

$\inf(Y_j) = -\infty$;

(les inégalités strictes donnant des intervalles ouverts).

Les *formules de réduction* définissant $\text{red}_C(\boldsymbol{X}) = \text{apx}(\text{proj}_j(C \cap \boldsymbol{X}))$ en fonction des valeurs extrêmes de $\boldsymbol{X}$ s'en déduisent.

En posant $\text{red}_C(\mathbf{X}) = \mathbf{X} \cap \mathbf{Z}$, on a, dans l'espace discrétisé $(\mathbb{R}, \Im_Z)$:

Si $j \in J^+$:

$\sup(Z_j) = +\infty$

$$\inf(Z_j) = \left\lfloor \frac{1}{a_j}\left(\sum_{l \in J^-} a_l \inf(X_l) - \sum_{k \in J^+ - \{j\}} a_k \sup(X_k) + b \right) \right\rfloor$$

Si $j \in J^-$:

$$\sup(Z_j) = \left\lfloor \frac{1}{a_j}\left(\sum_{k \in J^+} a_k \sup(X_k) - \sum_{l \in J^- - \{j\}} a_l \inf(X_l) - b \right) \right\rfloor$$

$\inf(Z_j) = -\infty$

L'intervalle Z_j sera fermé aux conditions suivantes :

si $j \in J^+$:	si $j \in J^-$:
$\inf(Z_j) = \inf(Y_j)$	$\sup(Z_j) = \sup(Y_j)$
$\forall k \in J^+ - \{j\}$, $\inf(X_k) = \min(X_k)$	$\forall k \in J^+$, $\inf(X_k) = \min(X_k)$
$\forall l \in J^-$, $\sup(X_l) = \max(X_l)$	$\forall l \in J^- - \{j\}$, $\sup(X_l) = \max(X_l)$

Les mises à jour des domaines sont similaires dans les espaces discrétisés $(\mathbb{R}, \Im_M)$ et $(\mathbb{R}, \Im_\mathbf{F})$, à condition d'interpréter $\lfloor x \rfloor$ comme «le plus grand élément de la droite discrète inférieur à x».

5.2 Pré-traitements particuliers

Les déductions précédentes ne sont pas l'apanage des techniques de résolution par intervalles. Elles sont utilisées depuis longtemps en programmation mathématique pour reformuler des modèles linéaires, en nombres entiers ou mixtes. Elles permettent de réduire, dans des phases de pré- ou post-processing, la dimension des problèmes soumis, par élimination de lignes (contraintes redondantes), de colonnes (variables figées) ou, plus simplement, par resserrement des bornes (que manipulent les simplexes en variables bornées). Mais ces techniques n'interviennent qu'en appoint de l'algorithme du simplexe, alors qu'elles sont, avec les intervalles, au cœur de l'algorithme du point fixe (en comparaison, bien pauvre).

Reprenons la contrainte C : $\sum_{i \in J^+} a_j x_j - \sum_{i \in J^-} a_j x_j - b \geq 0$ et les encadrements de $\boldsymbol{x}$ par le pavé $\boldsymbol{X}$ (que l'on supposera fermé).

L'élimination complète des variables donne comme condition nécessaire de satisfaisabilité :

$$SUP = \sum_{k \in J^+} a_k \sup(X_k) - \sum_{l \in J^-} a_l \inf(X_l) - b \geq 0$$

Symétriquement, on déduit :

$$INF = \sum_{i \in J^+} a_j \inf(X_j) - \sum_{i \in J^-} a_j \sup(X_j) - b \leq \sum_{i \in J^+} a_j x_j - \sum_{i \in J^-} a_j x_j - b$$

Par conséquent :

- si l'on a $INF \geq 0$, la contrainte C est redondante
- si l'on a $SUP < 0$, la contrainte C est insatisfaisable
- si l'on a $SUP = 0$, tous les x_j peuvent être fixés à leurs bornes, à savoir $\max(X_j)$ pour $j \in J^+$ et $\min(X_j)$ pour $j \in J^-$).

Historiquement, les articles fondateurs de Montanari pour la cohérence d'arc (en IA) et de Brearley, Mitra & Williams pour le pré-traitement des systèmes linéaires (en RO) datent de la même période (1974).

6 Application aux systèmes linéaires

6.1 Constat

Par rapport aux techniques de résolution, efficaces et complètes, de la programmation linéaire, la résolution par les intervalles d'un système linéaire est souvent décevante. A cela s'ajoute le fait que, mentalement, le passage d'une relation linéaire sur $\mathbb{R}^n$ à son extension sur $\mathbb{I}^n$ n'est pas des plus aisés, si bien qu'il est difficile de se figurer à l'avance les résultats d'un calcul.

Exemple Soit à résoudre le système suivant de trois équations à trois inconnues :
$P = \{(x, y, z) \in \mathbb{R}^3 \mid 3x = 2y + z,\ 3y = 2x + z,\ x + y + z = 1\}$.

La résolution linéaire donne la solution exacte :

```
>>  3*X = 2*Y + Z, 3*Y = 2*X + Z,  X + Y + Z = 1.
Z = 1/3, Y = 1/3, X = 1/3.
```

Les intervalles en donnent un encadrement assez lâche, dans [0,1] :

```
>>  3.*.X = 2.*.Y .+. Z, 3.*.Y = 2.*.X .+. Z,  X .+. Y .+. Z = 1,
    X ge 0, Y ge 0, Z ge 0.                  % contraintes imposant x, y, z ≥ 0.
Z ~ cc(0,1), Y ~ cc(0,1), X ~ cc(0,1).
```

Si les conditions deviennent incompatibles, la résolution linéaire le dit :

```
>>  3*X = 2*Y + Z, 3*Y = 2*X + Z, 3*Y = 2*X + 2*Z, X + Y + Z = 1.
false.
```

Mais pour les intervalles, l'ajout d'une quatrième équation passe inaperçu :

```
>>  3.*.X = 2.*.Y .+. Z, 3.*.Y = 2.*.X .+. Z, 3.*.Y = 2.*.X .+. 2.*.Z,
    X .+. Y .+. Z = 1, X ge 0, Y ge 0, Z ge 0.
Z ~ cc(0,1), Y ~ cc(0,1), X ~ cc(0,1).
```

L'exemple illustre l'incomplétude de la résolution par les intervalles.

Face à ce constat passablement négatif pour le linéaire, les résultats obtenus avec les systèmes monotones paraîtront appréciables.

Auparavant, nous devons insérer un bref développement sur les fonctions d'approximation dans $(\mathbb{R}, \Im_Z)$. Il nous sera utile pour analyser la précision de l'encadrement, à l'issue d'une réduction d'intervalles.

6.2 Approximation d'une fonction linéaire monotone

Considérons une forme affine $f: \mathbb{R}^{n-1} \to \mathbb{R}$, à coefficients rationnels, définie par : $f(\boldsymbol{x}) = \sum_{i=1}^{n-1} a_j x_j + b$, avec $a_j \geq 0$ et b de signe quelconque.

f est alors croissante en fonction de chacun des x_j.

Définition Soient $\underline{f}$ et $\bar{f}: \mathbb{R}^{n-1} \to \mathbb{Z}$ les fonctions en escalier définies par :

$$\underline{f}(\boldsymbol{x}) = \lfloor f(\lfloor \boldsymbol{x} \rfloor) \rfloor$$
$$\bar{f}(\boldsymbol{x}) = \lceil f(\lceil \boldsymbol{x} \rceil) \rceil.$$

$\underline{f}$ et $\bar{f}$ sont deux fonctions croissantes, l'une minorante, l'autre majorante de f :

$$\lfloor \boldsymbol{x} \rfloor \leq \boldsymbol{x} \leq \lceil \boldsymbol{x} \rceil \ \Rightarrow\ \underline{f}(\boldsymbol{x}) \leq f(\lfloor \boldsymbol{x} \rfloor) \leq f(\boldsymbol{x}) \leq f(\lceil \boldsymbol{x} \rceil) \leq \bar{f}(\boldsymbol{x})$$

En tout point $\boldsymbol{x}$ à coordonnées entières, l'encadrement $\underline{f}(\boldsymbol{x}) \leq f(\boldsymbol{x}) \leq \bar{f}(\boldsymbol{x})$ redonne l'intervalle délimité par $\lfloor f(\boldsymbol{x}) \rfloor$ et $\lceil f(\boldsymbol{x}) \rceil$ dont la largeur est 0 ou 1.

Pour un point à coordonnées fractionnaires, les relations entre les trois fonctions sont résumées dans le tableau suivant :

$$\begin{array}{ccccc} \underline{f}(\lceil \boldsymbol{x} \rceil) & \leq & f(\lceil \boldsymbol{x} \rceil) & \leq & \bar{f}(\lceil \boldsymbol{x} \rceil) \\ \wedge\!\vee & & \wedge\!\vee & & \| \\ \underline{f}(\boldsymbol{x}) & \leq & f(\boldsymbol{x}) & \leq & \bar{f}(\boldsymbol{x}) \\ \| & & \wedge\!\vee & & \wedge\!\vee \\ \underline{f}(\lfloor \boldsymbol{x} \rfloor) & \leq & f(\lfloor \boldsymbol{x} \rfloor) & \leq & \bar{f}(\lfloor \boldsymbol{x} \rfloor) \end{array}$$

De la définition de la partie entière et de la croissance de f on déduit :

$$\boldsymbol{x} < \lfloor \boldsymbol{x} \rfloor + \mathbf{1} \Rightarrow f(\boldsymbol{x}) \leq f(\lfloor \boldsymbol{x} \rfloor + \mathbf{1}) = f(\lfloor \boldsymbol{x} \rfloor) + \sum_{j=1}^{n-1} a_j .$$

En retranchant $\underline{f}(\boldsymbol{x})$ de l'inégalité obtenue, il vient :

$$f(\boldsymbol{x}) - \underline{f}(\boldsymbol{x}) = f(\boldsymbol{x}) - \lfloor f(\lfloor \boldsymbol{x} \rfloor) \rfloor \leq f(\lfloor \boldsymbol{x} \rfloor) - \lfloor f(\lfloor \boldsymbol{x} \rfloor) \rfloor + \sum_{i=1}^{n-1} a_j \leq 1 + \sum_{i=1}^{n-1} a_j .$$

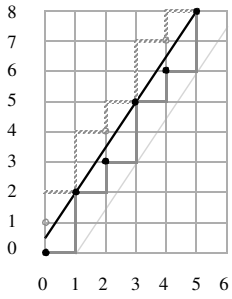
Par conséquent, en posant $\tau = 1 + \sigma$ avec $\sigma = \sum_{i=1}^{n-1} a_j$ et en définissant $\underline{\underline{f}}$ par $\underline{\underline{f}}(\boldsymbol{x}) = f(\boldsymbol{x}) - \tau$, on obtient, en plus de l'encadrement de f par $\underline{f}$ et $\bar{f}$, un encadrement de la fonction en escalier $\underline{f}$ entre $\underline{\underline{f}}$ et f :

$$\underline{\underline{f}}(\boldsymbol{x}) \leq \underline{f}(\boldsymbol{x}) \leq f(\boldsymbol{x}) \leq \bar{f}(\boldsymbol{x})$$

On peut donc minorer la fonction minorante par une forme affine monotone, parallèle à f et distante de f d'au plus τ.
De la même manière, on pourrait majorer la fonction majorante.

Remarque Si la forme affine f est décroissante, c'est-à-dire si tous ses coefficients a_j sont négatifs, il faut définir $\underline{f}(\boldsymbol{x})$ par $\lfloor f(\lceil \boldsymbol{x} \rceil) \rfloor$ et $\bar{f}(\boldsymbol{x})$ par $\lceil f(\lfloor \boldsymbol{x} \rfloor) \rceil$.

Exemple Droite $2y = 3x + 1$, d'équation : $y = f(x)$.



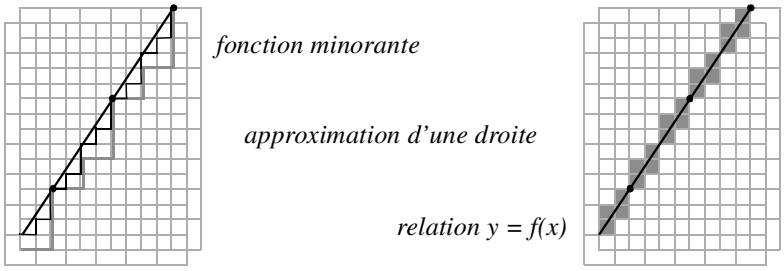
x	[0]	]0,1[	[1]	]1,2[	[2]	]2,3[	[3]	]3,4[	[4]	]4,5[	[5]
$\bar{f}(x)$	1	2	**2**	4	4	5	**5**	7	7	8	**8**
y	0.5		**2**		3.5		**5**		6.5		**8**
$\underline{f}(x)$	0	0	**2**	2	3	3	**5**	5	6	6	**8**

$\tau = 1 + 3/2$

Dans un espace discrétisé fini comme $(\mathbf{R}, \Im_M)$, la propriété d'encadrement de $\underline{f}$ entre f et une forme linéaire qui en est proche, $\underline{\underline{f}}$, n'est plus valable que «localement», pour des coordonnées comprises entre $-M$ et $+M$, puisqu'au delà on doit poser : $\forall\, x \in\,]-\infty, -M[,\ \lfloor x \rfloor = -\infty$.

Pour couvrir des points situés à la frontière de la grille de discrétisation, on peut soit jouer sur l'étendue de cette grille, soit effectuer une translation d'origine pour la repositionner.

Par ailleurs, un moyen d'affiner une approximation est d'augmenter la densité des points de calcul dans l'espace discrétisé.



Subdiviser la grille revient à effectuer un changement de variable au niveau du système. En calculant par exemple avec des nombres décimaux $z_j = 10^e x_j$ ($e \geq 0$) au lieu des entiers x_j, on aura : $\lfloor 10^e x_j \rfloor \leq 10^e x_j < \lfloor 10^e x_j \rfloor + 1$, soit :

$$10^{-e} \lfloor 10^e x_j \rfloor \leq x_j < 10^{-e} \lfloor 10^e x_j \rfloor + 10^{-e}$$

On parlera alors de résolution à l'*échelle* 10^e.

Propriété *Soit* $f(\boldsymbol{x}) = f_0 + b$, avec $f_0(\boldsymbol{x}) = \sum_{i=1}^{n-1} a_j x_j$, *une forme affine vérifiant* $a_j \geq 0$ *pour tout* j. *Soit* $e \in \mathbb{N}$ *un facteur d'échelle.*

En posant $\underline{f}^{(e)}(\boldsymbol{x}) = 10^{-e} \lfloor f_0(\lfloor 10^e \boldsymbol{x} \rfloor) + 10^e b \rfloor$ *et* $\tau = 1 + \sum_{i=1}^{n-1} a_j$ *on obtient dans* $(\mathbb{R}, \Im_{10^{-e}\mathbf{Z}})$ *l'encadrement à* $10^{-e}\tau$ *près :* $f(\boldsymbol{x}) - 10^{-e}\tau \leq \underline{f}^{(e)}(\boldsymbol{x}) \leq f(\boldsymbol{x})$.

Démonstration Tout d'abord, $\underline{f}^{(e)}$ est une fonction en escalier minorante à valeurs dans $10^{-e}\mathbb{Z}$.

f_0 étant linéaire et croissante, $f_0(\lfloor 10^e \boldsymbol{x} \rfloor) \leq f_0(10^e \boldsymbol{x}) = 10^e f_0(\boldsymbol{x})$.
Donc : $\lfloor f_0(\lfloor 10^e \boldsymbol{x} \rfloor) + 10^e b \rfloor \leq f_0(\lfloor 10^e \boldsymbol{x} \rfloor) + 10^e b \leq 10^e (f_0(\boldsymbol{x}) + b)$
et : $10^{-e} \lfloor f_0(\lfloor 10^e \boldsymbol{x} \rfloor) + 10^e b \rfloor \leq f(\boldsymbol{x})$, ce qui démontre l'inégalité de droite.

L'inégalité $\boldsymbol{x} < 10^{-e}(\lfloor 10^e \boldsymbol{x} \rfloor + \mathbf{1})$ et la croissance de f_0 donnent d'autre part :
$f_0(\boldsymbol{x}) \leq f_0(10^{-e} \lfloor 10^e \boldsymbol{x} \rfloor + 10^{-e}\mathbf{1}) = 10^{-e} f_0(\lfloor 10^e \boldsymbol{x} \rfloor) + 10^{-e}\sigma$
$f(\boldsymbol{x}) = f_0(\boldsymbol{x}) + b \leq 10^{-e}(f_0(\lfloor 10^e \boldsymbol{x} \rfloor) + 10^e b) + 10^{-e}\sigma$
$f(\boldsymbol{x}) - 10^{-e} \lfloor f_0(\lfloor 10^e \boldsymbol{x} \rfloor) + 10^e b \rfloor \leq 10^{-e}(1+\sigma) = 10^{-e}\tau$

D'où, l'inégalité de gauche : $f(\boldsymbol{x}) - \underline{f}^{(e)}(\boldsymbol{x}) \leq 10^{-e}\tau$. ■

Au bilan :

- l'effet positif d'un resserrement des mailles est une réduction de τ à $10^{-e}\tau$ unités de la largeur de la bande d'approximation inférieure délimitée par f et $\underline{f}$. En choisissant une résolution suffisamment fine, on peut ainsi faire tendre (localement) vers 0 la différence entre $\underline{f}$ et f.
- le prix à payer est un allongement des calculs, pour le point fixe notamment.

6.3 Approximation d'une relation linéaire monotone

Une inégalité monotone s'écrit $y \geq f(\boldsymbol{x})$ ou $0 \geq g(\boldsymbol{x})$, où f et g sont des formes affines croissantes. Seules les contraintes de la première forme interviennent dans la détermination du plus petit élément solution d'un système monotone.

6.3.1 Contrainte de forme (1)

Soit C_1 la relation définie sur $\mathbb{R}^n$ par : $C_1(x_1,\ldots,x_{n-1},y) \Leftrightarrow ay \geq \sum_{i=1}^{n-1} a_j x_j + b$

avec $a_j \geq 0$ et $a > 0$.

Propriété *Les formules de réduction donnent pour C_1 :*

$$\begin{array}{ll} y \in Y & \\ x_j \in X_j & \end{array} \Rightarrow \begin{array}{l} y \in Y' = Y \cap Z \\ x_j \in X'_j = X_j \cap Z_j \end{array}$$

avec :

$$\inf(Z) = \left| \frac{1}{a}\left(\sum_{j=1}^{n-1} a_j \inf(X_j) + b \right) \right| = \underline{f}(\inf(\boldsymbol{X})) \qquad (1.1)$$

$$\sup(Z_j) = \left| \frac{1}{a_j}\left(a\sup(Y) - \sum_{l \neq j} a_l \inf(X_l) - b \right) \right| , \forall\, j \in \{1,\ldots,n-1\} \qquad (1.2)$$

1. L'intervalle Z sera fermé si $\underline{f}(\inf(\boldsymbol{X})) = f(\inf(\boldsymbol{X}))$ et $\inf(X_j) = \min(X_j)$, $\forall j$; ouvert sinon.
2. L'intervalle Z_j sera fermé si $\sup(Z_j) = \frac{1}{a_i}\left(a\sup(Y) - \sum_{l \neq j} a_l \inf(X_l) - b \right)$ et si $\inf(Y) = \min(Y)$ et si $\forall l \neq j$, $\inf(X_l) = \min(X_l)$.

Remarque La forme affine croissante bornant inférieurement y atteint son minimum quand toutes ses variables atteignent leur minimum.

6.3.2 Contrainte de forme (2)

Soit C_2 la relation définie sur $\mathbb{R}^n$ par : $C_2(x_1,\ldots,x_n) \Leftrightarrow 0 \geq \sum_{i=1}^{n} a_j x_j + b$.

Propriété *Les formules de réduction donnent pour C_2 :*

$$x_j \in X_j \qquad \Rightarrow \qquad x_j \in X'_j = X_j \cap Z_j$$

avec :

$$\sup(Z_j) = \left| \frac{1}{a_j}\left(-\sum_{l \neq j} a_l \inf(X_l) - b \right) \right| , \forall\, j \in \{1,..., n\} \qquad (2)$$

L'intervalle Z_j sera fermé si $\sup(Z_j) = \frac{1}{a_j}\left(-\sum_{l \neq j} a_l \inf(X_l) - b \right)$ et si $\forall l \neq j$, $\inf(X_l) = \min(X_l)$.

Remarque Dans un langage comme Prolog IV, les contraintes C_1 et C_2 ne figurent pas parmi les relations de base. Elles se décomposent de manière équivalente en une conjonction de contraintes élémentaires formant un système monotone.

$\{z = ay, z_1 = a_1x_1, \ldots , z_{n-1} = a_{n-1}x_{n-1}, s_2 \geq z_1 + z_2, \ldots , s_{n-1} \geq s_{n-2} + z_{n-1}, s \geq s_{n-1} + b, z \geq s\}$

7 Résolution des systèmes monotones

L'intérêt des systèmes monotones provient de leurs bonnes propriétés pour le calcul.

Lorsque le polyèdre des solutions du système est non vide et minoré, il possède un plus petit élément. Le pavé des intervalles qui l'encadre possède une borne inférieure également, qui s'en approche d'autant plus que la finesse de résolution est grande. Le risque de «sur-approximation» du calcul par les intervalles, très pénalisant en d'autres circonstances, est, autour de ce point, écarté. Globalement, la qualité de l'approximation se transmet — des contraintes au système.

La précision finie utilisée pour le calcul ne permet guère d'espérer, dans le cas général, de résultat exact pour les bornes inférieures des projections sur les axes. On peut néanmoins énoncer quelques critères de complétude, particulièrement utiles pour la résolution des systèmes 1-monotones.

7.1 Condition suffisante de satisfaisabilité

La résolution par les intervalles d'un système monotone $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ retourne, pour chaque variable, un domaine réduit. Si ces domaines sont vides, le système n'a pas de solution. S'ils ne le sont pas, on ne peut en général rien conclure. Mais ici, un simple examen des bornes de ces intervalles peut apporter une réponse positive, sans énumération.

Théorème IV.2 *Soit* S *un système monotone sur* $\mathbb{R}^n$. *Soient* $X_1 \times \ldots \times X_n$ *les domaines réduits des variables de* S *dans un espace discrétisé. Si tous les* X_j *admettent un plus petit élément, alors* S *et soluble et a pour solution* $(\min(X_1), \ldots, \min(X_n))$.

Démonstration

Supposons que $\forall\, j \in \{1, \ldots, n\}$, $\min(X_j)$ existe. Posons P = Sol(S).
D'après le théorème 1, $\min(X_j) \in \text{proj}_j(\text{P})$. Donc P est non vide.
Comme X_j contient cette projection, $\min(X_j) = \min \text{proj}_j(\text{P})$ et P est minoré.

S étant un système monotone, $(\min(X_1), \ldots, \min(X_n))$ est le plus petit élément de P. ■

Corollaire *Lorsque les intervalles sont réduits à des points* $\{x_1\} \times \ldots \times \{x_n\}$*, le point* $(x_1, \ldots, x_n)$ *est solution du système.*

On retrouve là un résultat tout à fait général pour l'approximation des intervalles :

Si Prolog IV trouve une solution au problème approché dans laquelle toutes les variables prennent une valeur, alors cette solution est aussi solution du problème exact.

7.2 Qualité de l'approximation

Lorsque les systèmes monotones sont minorés de façon explicite, accroître la finesse de résolution améliore à coup sûr la qualité de l'approximation autour du plus petit élément du polyèdre des solutions.

Théorème IV.3 *Soient* ε *un réel positif et* S *un système monotone de* $\mathbb{R}^n$ *contenant un sous-système* $\boldsymbol{x} \geq \boldsymbol{c}$ *où* $\boldsymbol{c} \in \mathbb{R}^n$*. Soit* $X_1 \times \ldots \times X_n$ *le pavé des domaines réduits des variables de* S*. Si l'ensemble* P *des solutions de* S *est non vide, alors on peut toujours choisir une échelle de résolution pour que* $(\inf(X_1), \ldots, \inf(X_n))$ *s'approche de* min P *à moins de* ε*.*

Démonstration

Si P est non vide, P admet un plus petit élément puisqu'il est minoré. Notons $\mu = (\mu_1, \ldots, \mu_n)$ ce plus petit élément.

Le système S se décompose en deux sous-systèmes $\text{S}_1 \cup \text{S}_2$.

Toute contrainte de S_1 est de la forme $C_i : x_j \geq f_i(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)$.
D'après les formules de calcul **(1.1)**, on a :

$$\inf(X_j) \geq \underline{f_i}(\inf(X_1), \ldots, \inf(X_{j-1}), \inf(X_{j+1}), \ldots, \inf(X_n))$$

où $\underline{f_i}$ est la fonction d'approximation minorante de f_i. On en déduit :

$$\inf(X_j) \geq \underline{f_i}(\inf(X_1), \ldots, \inf(X_{j-1}), \inf(X_{j+1}), \ldots, \inf(X_n)) \qquad (1)$$

où la forme affine $\underline{f_i}$ est égale à $f_i - \tau_i$ ($\tau_i \geq 0$).

En particulier, $\inf(X_j) \geq \lfloor c_j \rfloor$ et $(\inf(X_1), \ldots, \inf(X_n)) \in \mathbb{R}^n$.

Toute contrainte de S_2 est de la forme $D_i : 0 \geq g_i(x_1, \ldots, x_n)$ et l'on a :

$$0 \geq g_i(\mu_1, \ldots, \mu_n) \geq g_i(\inf(X_1), \ldots, \inf(X_n)) \qquad (2)$$

(1) et (2) montrent que $(\inf(X_1), \ldots, \inf(X_n))$ est solution du système monotone $S' = S'_1 \cup S_2$, où les contraintes de forme (1) deviennent dans S'_1 : $C_i' = C_i - \tau_i : x_j \geq \underline{f_i}(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)$.

L'ensemble P' des solutions de S' est un polyèdre parallèle à P, non vide, puisqu'il contient P. Montrons que P' est également minoré.

Soit x_j une variable, $j \in \{1, \ldots, n\}$. L'inégalité $x_j \geq \mu_j$ est une conséquence du système S que l'on peut obtenir par élimination de Fourier des $n-1$ autres variables. Notons $\sum_{i \in I_1} \gamma_{ji} C_i + \sum_{i \in I_2} \gamma_{ji} D_i$ une combinaison à coefficients positifs d'inégalités de S conduisant à la connaissance du minimum de x_j. Alors, l'application de la même combinaison aux inégalités de S' donne également un minorant pour x_j : $x_j \geq \mu_j - \sum_{i \in I_1} \gamma_{ji} \tau_i$.

P', non vide et minoré, a donc un plus petit élément, $\nu = (\nu_1, \ldots, \nu_n)$.

On en déduit l'encadrement :

$$\nu = \min P' \leq \inf(X_1 \times \ldots \times X_n) \leq \mu = \min P$$

Plus précisément, on obtient, pour chaque coordonnée, la plage d'incertitude :

$$\mu_j - \sum_{i \in I_1} \gamma_{ji} \tau_i \leq \inf(X_j) \leq \mu_j$$

Il reste à montrer que la largeur $\lambda_j = \sum_{i \in I_1} \gamma_{ji} \tau_i$ de cette plage ne dépend que de S et de la finesse de la grille de résolution.

Lors d'une élimination de variable, on peut toujours choisir les coefficients de la combinaison linéaire parmi ceux de la matrice **A** du système[1] et il faut au plus $n-1$ éliminations successives pour arriver à un encadrement sur x_j. Donc chaque coefficient γ peut être majoré par α^{n-1} où $\alpha = \max(a_{ij})_{1 \leq i \leq m,\ 1 \leq j \leq n}$.

En prenant $\tau = \max(\tau_i)_{1 \leq i \leq m}$, on obtient la majoration :

$$\lambda_j \leq \alpha^{n-1} \sum_{i=1}^{m} \tau_i \leq m \alpha^{n-1} \tau$$

Pour une grille de pas égal à l'unité, on a $f_i = \frac{1}{a_{ij}} \left(\sum_{k \neq j} a_{ik} x_k + b_i \right)$ avec $a_{ij} \geq 1$.

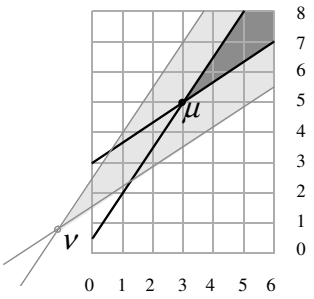
$\tau_i = \sum_{k=1}^{n} \frac{a_{ik}}{a_{ij}} \leq \sum_{k=1}^{n} a_{ik} \leq \max_i \sum_{k=1}^{n} a_{ik} = \| \mathbf{A} \|_\infty \leq n\alpha$. Donc : $\lambda_j \leq mn\alpha^n$.

1. Rappelons que, dans nos notations, les a_{ij} correspondent aux valeurs absolues.

Pour une grille de pas égal à $10^{-e}(e \geq 1)$, la largeur de la plage séparant v_j de μ_j est réduite d'un facteur 10^{-e}: $\tau_i = 10^{-e} \sum_{k=1}^{n} \frac{a_{ik}}{a_{ij}}$ et $\lambda_j = \leq 10^{-e} mn\alpha^n$.

Par conséquent, il suffit de prendre $e \geq \log_{10} m + \log_{10} n + n\log_{10}\alpha - \log_{10}\varepsilon$ pour avoir : $\max_j (\mu_j - v_j) = \| \mu - v \|_\infty \leq \varepsilon$. ■

Exemple $P = \{(x, y) \in \mathbb{R}^2 \mid 3x \geq 2y - 1, 3y \geq 2x + 9\}$ a pour minimum $\mu = (3,5)$.



$\tau_1 = \tau_2 = 1 + 2/3$

Si l'on ne déclare pas de minorant explicite de P, $(-\infty, -\infty)$ étant solution des inégalités dans l'extension des intervalles, on constate qu'il n'y a pas de réduction des domaines :

```
>>  3.*.X ge 2.*.Y .-. 1,
    3.*.Y ge 2.*.X .+. 9.

X ~ real, Y ~ real.                    % droite réelle : domaine implicite de x et de y.
```

En revanche, dès lors que l'on impose $(x, y) \geq$ constante, le minimum de P est approché à quelques flottants près (3 pour x, 2 pour y) :

```
>>  X ge -100, Y ge -100,
    3.*.X ge 2.*.Y .-. 1,
    3.*.Y ge 2.*.X .+. 9.

X ~ ge(`>2.9999992`), Y ~ ge(`>4.999999`). % demi-droites (théoriquement ouvertes).


>>  float_rank(`>2.9999992`,RX), float_rank(3, R3),
    float_rank(`>4.999999` ,RY), float_rank(5, R5),
    DX = R3-RX, DY = R5-RY.

DX = 3, DY = 2,                        % distances aux valeurs exactes, en nombre de flottants.
    RX = 1077936125, R3 = 1077936128,
    RY = 1084227582, R5 = 1084227584.
```

Détaillons les calculs sur une grille entière. Les formules actives pour la réduction des domaines sont celles qui suivent le sens des inégalités (1.1) :

$$\inf(X) \geq \left\lfloor \frac{2}{3}\inf(Y) - \frac{1}{3} \right\rfloor$$

$$\inf(Y) \geq \left\lfloor \frac{2}{3}\inf(X) + 3 \right\rfloor$$

Après 2 réductions, en partant de $x, y \geq 0$, un point fixe est atteint :

inf(X)	inf(Y)
0	0
0	3
1	3

Si l'on passe à l'échelle 10^1, les formules (1.1) deviennent :

$$\inf(X) \geq 10^{-1}\left\lfloor \frac{20}{3}\inf(Y) - \frac{10}{3} \right\rfloor \quad \text{et} \quad \inf(Y) \geq 10^{-1}\left\lfloor \frac{20}{3}\inf(X) + 30 \right\rfloor$$

Le nouveau point fixe se rapproche du minimum $\mu = (3{,}5)$:

inf(X)	inf(Y)
0	0
0	3
1,6	3
1,6	4
2,3	4
2,3	4,5
2,6	4,5
2,6	4,7
2,8	4,7
2,8	4,8

7.3 Cas des systèmes 1-monotones

Pour les systèmes 1-monotones, les calculs des bornes inférieures se simplifient puisqu'ils s'effectuent sans division. Les contraintes linéaires s'expriment directement à l'aide de formes affines f_i à coefficients entiers :

$$C_i : x_j \geq f_i(\boldsymbol{x}) = \sum_{k \neq i} a_{ik}x_k + b_i \quad (a_{ik}, b_i \text{ entiers}, a_{ik} \geq 0)$$

La mise à jour par C_i de la borne inférieure du domaine de la variable x_j donne lieu à une réécriture : $X_1 \times \ldots \times X_n \rightarrow X_1 \cap Z_1 \times \ldots \times X_n \cap Z_n$ dans laquelle $\inf(Z_j) = \left\lfloor \sum_{k \neq i} a_{ik}\inf(X_k) + b_i \right\rfloor$ et $\inf(Z_k) = -\infty$ si $k \neq j$.

Si l'on part d'un pavé fermé dont les bornes inférieures $\min(X_j)$ sont entières, la formule **(1.1)** précédente se simplifie sur $(\mathbf{R}, \Im_Z)$ en :

$$\min(Z_j) = \sum_{k \neq i} a_{ik}\min(X_k) + b_i = f_i(\min(X_1 \times \ldots \times X_n))$$

Si l'intersection $X_j \cap Z_j$ n'est pas vide, alors le domaine réduit de x_j possède un plus petit élément $\min(X_j \cap Z_j) = \max(\min(X_j), \min(Z_j))$ qui est entier. Tant qu'aucun de ces intervalle n'est vide, l'existence d'un minimum entier reste donc une propriété invariante du pavé des domaines réduits.

Cette remarque justifie l'utilisation d'une grille de calcul entière sur laquelle le min peut être calculé exactement.

La terminaison du calcul de point fixe n'étant pas garantie sur $(\mathbb{R}, \Im_Z)$, on se place en pratique dans l'espace d'approximation $(\mathbb{R}, \Im_M)$. Les bornes inférieures des intervalles d'approximation croissent et restent entières tant qu'elles ne dépassent pas la limite supérieure de la grille, M.

Théorème IV.4 *Soit* $P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{c}\}$ *l'ensemble des solutions d'un système 1-monotone explicitement minoré par* $\boldsymbol{c} \in \mathbb{Z}^n$. *Si* min P *existe, alors, pour tout entier M suffisamment grand,* $\min P = \min(X_1 \times \ldots \times X_n)$ *où* $X_1 \times \ldots \times X_n$ *est le pavé des domaines réduits des variables dans* $(\mathbb{R}, \Im_M)$.

Démonstration Le polyèdre P n'étant pas vide, aucun des X_j ne l'est. Initialisée avec des demi-droites $[c_j, +\infty[$ pour $j = 1, \ldots, n$, la procédure de point fixe réduit des intervalles dont la borne inférieure croît tout en restant un minimum entier.

Pour s'assurer que les valeurs calculées avec les formules (1.1) ne tombent jamais hors de l'intervalle $]-M, +M[$, il suffit de choisir M plus grand que :

- $\| f_i(\boldsymbol{c}) \|_\infty$, pour toute contrainte de forme (1), $x_j \geq f_i(\boldsymbol{x})$, dans $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$,
- $\| \boldsymbol{c} \|_\infty = \max\{| c_j |\}_{1 \leq j \leq n}$ (conditions pour amorcer les calculs) ;
- $\| \min P \|_\infty$ (condition pour terminer les calculs).

Le point fixe vérifie alors : $\inf(X_1 \times \ldots \times X_n) = \min(X_1 \times \ldots \times X_n)$ et le résultat découle du théorème **IV.2**. ■

Si min P existe, alors min P figure sur la grille entière et, à ce niveau, le pavé des domaines réduits s'ajuste parfaitement à P.

Corollaire *Soit* $P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{d} \geq \boldsymbol{x} \geq \boldsymbol{c}\}$ *l'ensemble des solutions d'un système 1-monotone explicitement borné (*$\mathbf{A}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}$ *à coefficients entiers). Alors, pour tout entier M entier suffisamment grand, la résolution du système dans l'approximation* $(\mathbb{R}, \Im_M)$ *est complète.*

Démonstration Il s'agit de montrer que si aucun des domaines réduits $X_1, \ldots, X_n$ n'est vide, alors P n'est pas vide — la réciproque étant toujours vraie.

Tout d'abord, si $X_1 \times \ldots \times X_n$ est non vide, tous les X_j ont un minimum $\min(X_j)$ entier (positif). L'argument est le même que précédemment : les calculs des bornes inférieures seront exacts si M est supérieur à $\max(\| \boldsymbol{c} \|_\infty, \| \boldsymbol{d} \|_\infty)$. Notons $\xi_j = \min(X_j)$ et montrons que ξ est solution de (P).

Pour toute contrainte C_i de forme (1) : $x_j \geq f_i(\boldsymbol{x})$, on a d'après **(1.1)** :

$$\xi_j \geq f_i(\xi) = f_i(\xi)$$

et ξ est solution de C_i.

Considérons maintenant une contrainte D_i de forme (2) : $0 \geq \sum_{j=1}^{n} a_{ij}x_j + b_i$.

D'après cette fois la formule **(2)** de mise à jour des bornes, on a, pour tout indice j compris entre 1et n :

$$\xi_j \leq \sup(X_j) \leq \left| \frac{1}{a_{ij}}\left(- \sum_{k \neq j} a_{ij}\xi_k - b_i \right) \right|$$

Si l'on prend soin de choisir M supérieur à $- \sum_{k \neq j} a_{ij}c_k - b_i$, toutes les parties entières sont calculées exactement dans $(\mathbb{R}, \mathfrak{I}_M)$ puisqu'elles sont finies. Alors, de deux choses l'une :

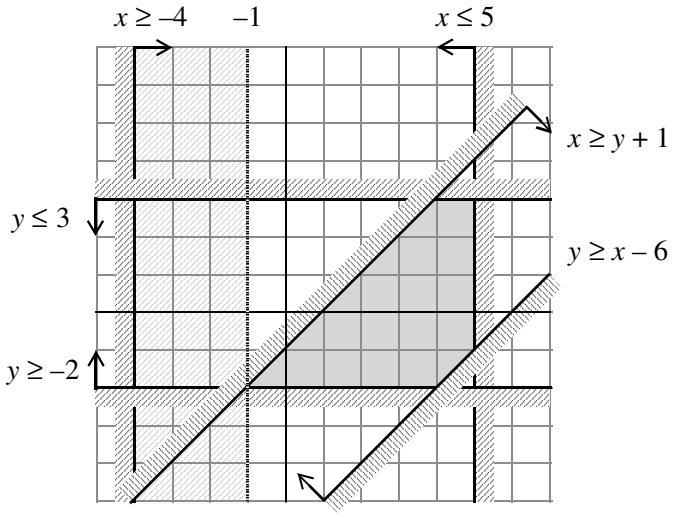
- soit $\frac{1}{a_{ij}}\left(- \sum_{k \neq j} a_{ij}\xi_k - b_i \right)$ est entier et dans ce cas : $\xi_j \leq \frac{1}{a_{ij}}\left(- \sum_{k \neq j} a_{ij}\xi_k - b_i \right)$
- soit $\frac{1}{a_{ij}}\left(- \sum_{k \neq j} a_{ij}\xi_k - b_i \right) < \left| \frac{1}{a_{ij}}\left(- \sum_{k \neq j} a_{ij}\xi_k - b_i \right) \right|$ mais alors X_j est ouvert à droite et $\xi_j < \sup(X_j)$; comme ξ_j est entier, on en déduit nécessairement : $\xi_j \leq \left| \frac{1}{a_{ij}}\left(- \sum_{k \neq j} a_{ij}\xi_k - b_i \right) \right| < \frac{1}{a_{ij}}\left(- \sum_{k \neq j} a_{ij}\xi_k - b_i \right)$

  L'inégalité obtenue par transitivité montre que ξ est encore solution de D_i.

ξ satisfait donc à toutes les contraintes définissant P. ■

Lorsqu'un système 1-monotone est borné, sa résolution par les intervalles permet de dire sans ambiguïté s'il est ou non satisfaisable, et si oui, d'en donner la plus petite solution.

Exemple $-4 \leq x \leq 5$, $-2 \leq y \leq 3$, $x \leq y + 6$, $y + 1 \leq x$ est un système à deux variables, borné, à coefficients dans $\{-1,0,1\}$ dont la matrice et son inverse ont au plus un coefficient positif par ligne. Il admet donc un minimum et un maximum entier.



Les formules de réduction donnent les réécritures $(X, Y) \rightarrow (X', Y')$ avec,

- pour le calcul du minimum :
  $\min(X') = \max(\min(X), \min(Y) + 1)$
  $\min(Y') = \max(\min(Y), \min(X) - 6)$
- pour le calcul du maximum :
  $\max(X') = \min(\max(X), \max(Y) + 6)$
  $\min(Y') = \max(\min(Y), \min(X) - 1)$

Après rectification du domaine de x à $[-1,5]$ par l'effet de la dernière contrainte, on vérifie que les couples formés des minimums $(-1, -2)$ et maximums $(5, 3)$ sont bien des solutions.

7.4 Résolution sur les entiers

La complétude de la résolution des systèmes monotones sur des domaines finis d'entiers découle :

1. de celle sur les réels pour les problèmes 1-monotones bornés (qui calcule exactement l'extrêmum global entier)
2. de l'application de l'algorithme de troncature exposé au chapitre précédent à la relaxation linéaire continue du problème.

Le simple ajout des contraintes d'intégrité $\boldsymbol{x} \in \mathbb{Z}^n$ au système linéaire produit en effet les coupes désirées.

Propriété *L'opérateur de réduction associé à la relation* $x \in \mathbb{Z}$ (`int/1` *en Prolog IV*) *définit un intervalle fermé* $X' = \mathrm{red}_{\mathrm{int}}(X) = \mathrm{apx}(X \cap \mathbb{Z})$ *dont les bornes sont les parties entières de celles de X :*

$$\inf(X') = \lceil \inf(X) \rceil$$
$$\sup(X') = \lfloor \sup(X) \rfloor$$

Le mécanisme de point fixe se charge implicitement des itérations.
Comme dans l'exemple de la formulation linéaire du PGCD, la résolution se fait sans programmer d'énumération.

8 Complexité

Concluons ce chapitre sur un bref rappel de la complexité de la résolution d'un système de contraintes par les intervalles.

Sa mesure, dans le pire des cas, dépend peu de la nature du système résolu si l'on considère, comme le fait N'Dong (1999), chaque réduction comme une opération unitaire. Elle résulte essentiellement du nombre de contraintes, m, et du nombre de variables, n.

Propriété *Le nombre maximal de réductions de l'algorithme du point fixe croît comme* $O(mn)$.

Démonstration

Soit S un système de contraintes résolu avec les intervalles. Chaque itération de l'opérateur de réduction red_S fait décroître strictement le pavé définissant les domaines des variables. Ce qui veut dire qu'au moins un intervalle du produit décroît strictement.

Prenons pour mesure de la taille d'un intervalle le nombre d'intervalles atomiques qu'il contient. Ce nombre est compris entre 1 et le nombre total d'intervalles atomiques qui est de l'ordre de $2|D|$, soit $4M$ pour $(\mathbb{R}, \mathcal{I}_M)$ — 2.2^{32} pour Prolog IV (exactement 8 556 380 159).

La décroissance d'un intervalle se traduit par une décroissance de sa taille. La somme des tailles des intervalles donne une mesure de la taille d'un pavé qui décroît à chaque itération. Le nombre total d'itérations de l'algorithme du plus grand point fixe est donc borné par $2n|D|$.

Chaque application de red_S correspond en fait à l'exécution en parallèle de m réductions par des contraintes élémentaires. Le nombre de réductions élémentaires est donc égal à $2nm|D|$. ■

Si l'on admet d'affecter un coût forfaitaire u à chaque opération de mise à jour des domaines (ce qui paraît raisonnable quand les contraintes de base ont un petit nombre de variables), on obtient, pour le temps de résolution, une borne en $2|D|u \times nm$.

Lorsque les contraintes de base on plus de 3 variables, comme dans l'expression littérale d'un système linéaire de m inégalités à n inconnues, il faut faire entrer un facteur n^2 supplémentaire (puisqu'il y a n formules de réduction par contraintes et que chacune implique n additions). L'avantage est que le nombre de contraintes élémentaires, m, est dans ce cas moins élévé, puisque l'on évite la décomposition d'une inégalité en contraintes à 3 variables. Ceci donne un chiffrage en $O(mn^3)$, pessimiste quand le nombre de variables par contrainte est réduit par rapport au nombre total de variables. Si le nombre maximal de variables par contrainte est inférieur à une constante v, l'expression de la complexité théorique se ramène à $2|D|uv^2 \times nm$.

Par comparaison, Dantzig, Chvátal rapportent que le nombre d'itérations de l'algorithme du simplexe (qui, dans le pire des cas, peut être exponentiel) croît *en moyenne* proportionnellement à m. Le rapport empirique est donné voisin de 3.

Le coût d'un pivotage étant de l'ordre de mn opérations arithmétiques, on en déduit que le coût opérationnel moyen est comparable à la complexité d'une élimination de Gauss, à savoir $O(n^3)$ lorsque $n = m$.

A ce nombre d'additions et de multiplications, vient s'ajouter le surcoût dû à la gestion de la précision infinie, que l'on peut à son tour borner par $n^2(\log n)^2$ opérations élémentaires en temps constant. De sorte que le coût moyen de la résolution d'un système linéaire par le simplexe s'élève grosso modo à $O(n^5)$ si l'on n'opère pas en calcul flottant.

L'avantage relatif de la résolution par les intervalles en termes de complexité théorique ne doit pas faire oublier l'importance du coefficient associé à *nm*. Cette constante de temps vaut, pour Prolog IV : 8 556 380 159 — et encore ne s'agit-il que d'une implantation en 32 bits !

II

CHAPITRE V

De Fourier à Davis et Putnam

Satisfaisabilité en logique propositionnelle

> «All logic in the sense of *Principia*, and hence all mathematics as well, can be translated into a language which consists only of an infinity of variables '*x*', '*y*', '*z*', '*x*'', etc., and these three modes of notational composition: *membership* ; *alternative denial* ; and *universal quantification*.»
>
> W. QUINE

La logique propositionnelle est la plus simple des logiques. Elle ne s'embarrasse ni d'appartenance ensembliste ni de quantification universelle — notions propres aux logiques d'ordres supérieurs. Elle ne manipule à travers ses variables que des *propositions*, soit *vraies* (1), soit *fausses* (0).

Une *formule* logique à plusieurs variables $x_1, x_2, \ldots, x_n$ est l'expression syntaxique d'une *fonction* de ces variables, définie sur $\{0,1\}^n$ et à valeur dans l'ensemble $\{0,1\}$ des valeurs booléennes. Par construction, toute formule bien formée est automatiquement pourvue de sens.

Quoiqu'un seul connecteur binaire (NAND) suffise à engendrer, par induction, toutes les formules propositionnelles, il est plus courant de retenir pour fonctions de base le jeu formé par la négation '$\neg$' (unaire), la conjonction '$\wedge$' et la disjonction '$\vee$' (binaires).

Les quatre fonctions unaires ($\mathbf{0}$, x, $\neg x$, $\mathbf{1}$) s'expriment à l'aide de ces trois connecteurs. Et toute fonction logique f de n+1 variables se décompose, par étude de cas, en une disjonction :

$$f(x, \boldsymbol{x'}) \;=\; (x \wedge f_1(\boldsymbol{x'})) \vee (\neg x \wedge f_0(\boldsymbol{x'}))$$

où f_0 et f_1 sont des fonctions de n variables exprimables à l'aide de ces trois connecteurs.
On rappelle que parmi les 16 fonctions binaires, $x \Rightarrow y$ correspond à $\neg x \vee y$ ou encore à $\neg(x \wedge \neg y)$: une implication est vraie si et seulement si elle ne part pas du vrai pour aboutir au faux.

Dans ce chapitre, nous réexaminons le problème classique de la satisfaisabilité d'une formule de logique propositionnelle en le transposant dans le calcul des inégalités. Nous l'envisageons d'abord sur des classes de systèmes monotones, «faciles à résoudre» sans énumération, puis, le champ s'élargissant, nous avons recours à l'énumération, mais modérément, en nous appuyant autant que possible sur des sous-systèmes monotones.

Partant de la transcription d'une forme normale conjonctive en un système de contraintes linéaires en {0,1}, nous nous intéressons en premier aux clauses de Horn utilisées dans les systèmes experts.

Les systèmes de Horn sont, dans notre terminologie, précisément 1-monotones. L'équivalence de l'élimination sur les entiers et les réels nous offre une panoplie de procédures de décision complètes pour tester la satisfaisabilité d'un tel système :

- résolution unitaire (une spécialisation du procédé d'élimination de Fourier),
- relaxation linéaire sur [0,1] (généralement synonyme de l'algorithme du simplexe)
- réduction par les intervalles (qui par ailleurs réalise la résolution unitaire).

Ces procédures, au pouvoir de déduction équivalent, restent complètes pour les systèmes Horn-renommables dont la complexité est quadratique.

Leur généralisation à SAT n'est plus complète car le problème, bien entendu, n'est plus polynômial.
L'élimination de Fourier sur les entiers redonne — si l'on prend soin de resserrer les inégalités produites, la procédure de résolution de Robinson.

Or cette dernière est moins efficace qu'une procédure avec énumération. La référence en la matière est celle de Davis & Putnam, que l'on peut présenter comme une méthode de *branch and cut* avant la lettre.

La combinaison filtrage + énumération, où le filtrage est effectué par la résolution unitaire (ou une technique équivalente) nous conduit à l'idée de la Horn-relaxation.

Les travaux de Gallo confirment cette intuition que l'on peut tirer parti d'une sous-structure de Horn pour mieux maîtriser la combinatoire de SAT. Méthodologiquement, cela s'entend après reformulation du modèle.

La mise sous forme normale étant un problème en soi, les langages de programmation par contraintes offrent, avec les domaines finis, une approche plus flexible (pas nécessairement plus efficace). Le programmeur y compose librement des conjonctions de relations booléennes, à deux ou trois variables, avant de programmer une énumération.

Nous montrons que chacune de ces relations de base se décompose en un petit système monotone — ou Horn-renommable — qui en définit donc une implantation.

1 Satisfaisabilité d'une formule

Définition Une formule propositionnelle est *satisfaisable* s'il existe une affectation de ses variables dans {0,1} qui la rende vraie, c'est-à-dire égale à 1.

Dans l'acception logique du terme, un *modèle* d'une formule f est une solution de l'équation $f(\boldsymbol{x}) = 1$.

Définition Une formule identiquement vraie, c'est-à-dire vraie quelle que soit l'affectation de ses variables, est une tautologie ou *théorème* de la logique propositionnelle.

Exemple **Si** la première, **alors** la seconde ;
or la première ;
donc la seconde.

Ce syllogisme, qui servait de schéma d'inférences dans la dialectique stoïcienne, ancêtre de la logique propositionnelle, est fondé sur le théorème : $((x_1 \Rightarrow x_2) \wedge x_1) \Rightarrow x_2$.

Propriété *Une condition nécessaire et suffisante pour qu'une formule soit un théorème est que sa négation soit insatisfaisable (identiquement fausse).*

Cette question centrale de la satisfaisabilité d'une formule est décidable en logique propositionnelle (les combinaisons à tester étant en nombre fini).

2 Formes normales

Pour normaliser l'écriture des expressions booléennes, on peut exiger :

1. que tout symbole de négation précède immédiatement une variable, par application des lois de De Morgan ;
2. que les toutes conjonctions précèdent les disjonctions (ou vice versa), par application de la distributivité de la disjonction sur la conjonction.

On parle de *forme normale* conjonctive ou disjonctive suivant que l'expression résultante est une conjonction de disjonctions ou une disjonction de conjonctions.

Définition On appelle *littéral* toute occurrence d'une proposition x ou de sa négation $\neg x$ dans une forme normale. Lorsque la proposition est précédée du signe de négation, le littéral est dit *négatif*. Il est *positif* sinon.

2.1 Forme normale disjonctive

D'une expression donnée sous forme normale disjonctive, on sait déduire immédiatement tous ses modèles.

En effet, une disjonction généralisée $f = \bigvee_{i \in \mathrm{I}} D_i$ est vraie dès lors que l'un de ses membres D_i est vrai (I désigant un ensemble fini d'indices).

Les modèles de f découlent donc de l'union des modèles des D_i. Or, chaque D_i étant une conjonction de littéraux, D_i n'est vraie que si chacun de ses membres est vrai. Il suffit, pour obtenir un modèle de D_i (et donc de f), d'égaler à vrai toutes les variables intervenant dans les littéraux positifs de D_i et à faux toutes les variables intervenant dans les littéraux négatifs de D_i (l'affectation des autres variables de f étant indifférente).

Une forme normale disjonctive donne ainsi une vision *extensionnelle* d'une fonction, en énumérant — parfois avec redondance — ses modèles.

2.2 Forme normale conjonctive

Une forme normale conjonctive correspond, en revanche, à une expression *intensionnelle*, en général plus condensée (plus propice à la représentation des connaissances), mais aussi à partir de laquelle il est plus difficile de statuer sur la satisfaisabilité d'une formule.

Définition Une *clause* est une disjonction de littéraux.

Une forme normale conjonctive $f = \bigwedge_{i \in \mathrm{I}} C_i$ est une conjonction de clauses.

La satisfaisabilité de la formule f définit la satisfaisabilité du *système de clauses* $(C_i)_{i \in \mathrm{I}}$.

Sous réserve de simplifications évidentes au niveau de la forme clausale ($x \vee x = x$, $x \vee \neg x = 1$), on pourra faire l'hypothèse qu'une variable propositionnelle n'a jamais plus d'une occurrence au sein d'une clause.

Montrer qu'un système de clauses n'a pas de modèle (ou les énumérer tous) revient, en quelque sorte, à passer d'une forme normale conjonctive à une forme normale disjonctive. Or, cette opération est éminemment combinatoire. Dans le cas général, sa complexité est exponentielle en fonction du nombre de variables que renferme la formule. Le problème de la satisfaisabilité d'une formule booléenne, ou problème SAT en abrégé, est d'ailleurs la figure emblématique des problèmes classés NP-complets par Cook (1971), en référence à la difficulté de leur résolution algorithmique.

Nous nous intéresserons principalement dans ce qui suit à des classes de problèmes SAT exprimés sous forme clausale[1] où l'on peut montrer que, même dans le pire des cas, cette complexité reste polynômialement bornée.

1. La mise sous forme normale conjonctive d'une formule quelconque est également combinatoire : la taille de la forme clausale peut croître de façon exponentielle. Mais on peut rendre cette transformation linéaire en rajoutant des variables (Siegel 1987).

3 Représentation des connaissances

Généralement, il est plus parlant de présenter une clause sous la forme d'une règle d'implication. Il suffit pour cela de regrouper les littéraux négatifs dans la partie prémisses et les littéraux positifs dans la partie conclusion.

$\neg x_1 \vee \ldots \vee \neg x_n \vee x_{n+1} \vee \ldots \vee x_{n+p}$ équivaut en effet, pour $n, p \geq 0$, à :
$x_1 \wedge \ldots \wedge x_n \Rightarrow x_{n+1} \vee \ldots \vee x_{n+p}$.

Beaucoup de problèmes appliqués de logique peuvent être exprimés à l'aide de clauses plus directives, ne présentant pas d'alternative en conclusion.

Définition Une clause ayant au plus un littéral en conclusion est appelée *clause de Horn.*

En logique du premier ordre, les clauses de Horn servent de fondement au langage de programmation Prolog.

En reprenant la terminologie des systèmes à base de règles, on obtient la classification suivante des clauses de Horn propositionnelles.

Forme	Clause			Dénomination	Littéraux négatifs	Littéraux positifs
1	$x_1 \wedge \ldots \wedge x_n$	$\Rightarrow$	x_{n+1}	règle	$n \geq 1$	$p = 1$
2		$\Rightarrow$	x_1	fait	$n = 0$	$p = 1$
3	$x_1 \wedge \ldots \wedge x_n$	$\Rightarrow$		requête	$n \geq 1$	$p = 0$
4		$\Rightarrow$		clause vide	$n = 0$	$p = 0$

La clause (1) se lit : si x_1 et ... x_n sont vrais, alors x_{n+1} est vrai.
La clause (2) affirme (inconditionnellement) que x_1 est vrai.
La clause (3) exprime que x_1 et ... x_n sont mutuellement incompatibles (ou, tout simplement, si $n = 1$, que x_1 est faux).

Une base de connaissances est d'ordinaire constituée de règles «de production» et de faits, c'est-à-dire de clauses ayant un et un seul littéral positif. Le terme de requête, but ou question pour désigner une clause où tous les littéraux sont négatifs vient de l'usage en mode interrogatif de ces bases (qui s'apparente à une démonstration par l'absurde[1]).

Exemple L'affirmation «S'il fait jour, alors il fait clair» est une règle.
L'affirmation «il fait jour» est un fait.
La proposition «il fait clair» est une conséquence logique de ces deux affirmations. Un mécanisme à la Prolog en donnerait une preuve par réfutation, en montrant que la négation de cette dernière (prise comme question) est incompatible avec les deux affirmations précédentes.

1. On peut toujours se ramener au cas où chaque règle à une conclusion. Il suffit de rajouter, à droite du signe d'implication, une variable fictive égale à 0 dans toute interprétation, ce qui signifie «conclusion absurde».

La *clause vide* ($1 \Rightarrow 0$) que nous mentionnons ici parmi les clauses de Horn est un artéfact de calcul. Elle n'est jamais présente à l'origine dans un système de clauses ou dans une base de connaissances. Elle correspond à une disjonction ne contenant aucun litteral, autrement dit à faux (l'élement neutre pour la disjonction).

Un système contenant une clause vide n'est donc jamais satisfaisable[1].

4 Traduction dans le calcul des inégalités

Avoir identifié le vrai à 1 et le faux à 0 nous incite à analyser de plus près les similitudes entre calcul booléen et calcul en nombres entiers.

Une fois le domaine de calcul plongé dans l'hypercube unité, nous pouvons réinterpréter tout système de clauses comme un système de contraintes linéaires en $\{0,1\}$. Nous montrerons aux paragraphes suivants quel parti tirer de cette reformulation pour une résolution pratique de ces systèmes, et comment l'application du calcul des inégalités conduit presque naturellement aux méthodes algorithmiques du domaine de la démonstration automatique.

4.1 Transcription d'un système de clauses

On constate tout d'abord que la fonction négation s'identifie à la restriction de l'application qui à x associe $1 - x$ sur $\{0,1\}$. Considérant deux variables propositionnelles, x et y, comme des inconnues dans $\{0,1\}$, on se convainc aisément que $x \vee y$ est vrai si et seulement si $x + y$ est supérieur à 1.

Plus généralement, affirmer : $\bigvee_{j \in J} x_j$ revient à affirmer : $\sum_{i \in J} x_j \geq 1$.

Les correspondances pour les connecteurs les plus courants sont résumées dans le tableau suivant (dans lequel $\oplus$ désigne le «ou exclusif») :

x	y	$x \vee y$ $\equiv$ $(x + y \geq 1)$	$x \wedge y$ $\equiv$ $(x + y \geq 2)$	$x \Rightarrow y$ $\equiv$ $(x \leq y)$	$x \Leftrightarrow y$ $\equiv$ $(x = y)$	$x \oplus y$ $\equiv$ $(x \neq y)$
0	0	0	0	1	1	0
0	1	1	0	1	0	1
1	0	1	0	0	0	1
1	1	1	1	1	1	0

Toute clause C : $\neg x_1 \vee \ldots \vee \neg x_n \vee x_{n+1} \vee \ldots \vee x_{n+p}$ se traduit ainsi par une inéquation $(1-x_1) + \ldots + (1-x_n) + x_{n+1} + \ldots + x_{n+p} \geq 1$.

La règle $x_1 \wedge \ldots \wedge x_n \Rightarrow x_{n+1} \vee \ldots \vee x_{n+p}$ devient :

$$x_1 + \ldots + x_n \leq x_{n+1} + \ldots + x_{n+p} + (n - 1)$$

où le signe d'inégalité prend la place du signe d'implication.

1. A ne pas confondre avec une *formule* ou système de clauses *vide*, qui est une conjonction ne contenant aucune clause, et qui, par conséquent, est toujours satisfaisable.

Un système de m clauses est donc l'équivalent d'un système de m inéquations linéaires $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ à résoudre dans $\{0,1\}^n$, si n est le nombre de ses variables. Les coefficients de la matrice $\mathbf{A}$ du système sont les entiers –1, 0, 1 (suivant le signe du littéral dans la clause dont chaque ligne est issue). Les coefficients du membre constant $\boldsymbol{b}$ sont des entiers relatifs dont la valeur est dictée par le nombre de littéraux négatifs dans chaque clause.

Avant de poursuivre, précisons nos notations.

La contrainte $\sum_{i \in J^-} x_j \leq \sum_{i \in J^+} x_j + (|J^-| - 1)$ *est la transcription d'une clause C.*

J est l'ensemble des indices de variables ayant des occurrences dans C. J est la réunion disjointe de J^+et J^-, ensembles (enventuellement vides) des indices des littéraux positifs et négatifs. $|J^-|$ désigne le nombre de littéraux négatifs.

4.2 Transcription d'un système de Horn

En particularisant l'inégalité précédente aux clauses de Horn ($|J^+| \leq 1$), on obtient les déclinaisons suivantes.

Forme	Contrainte	Dénomination	Commentaire
1	$1 + \sum_{i \in J^-} x_j \leq x_k + \|J^-\|$	règle	$\|J^-\| \geq 1, k \notin J^-$
2	$1 \leq x_k$	fait	$x_k = 1$
3	$1 + \sum_{i \in J^-} x_j \leq \|J^-\|$	requête	$\|J^-\| \geq 1$
4	$1 \leq 0$	clause vide	contradiction

Ces différents cas peuvent être rassemblés sous la forme :

$$\sum_{i \in J^-} x_j \leq \delta_k x_k + (|J^-| - 1) \qquad (1)$$

avec δ_k coefficient donné dans $\{0,1\}$, $k \notin J^-$ et $x_j \in \{0,1\}$ pour tout $j \in J$.

Toutes les inégalités étant orientées dans le même sens,

Propriété *Un système de clauses de Horn forme un système 1-monotone* $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$.

Remarque Les systèmes dont les clauses ont au plus un littéral négatif ($|J^-| \leq 1$) sont également monotones (de la forme $\mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}$). Ils ne constituent pas des systèmes de Horn, mais des systèmes *renommables* en systèmes de Horn (par le changement de variables : $\boldsymbol{y} = \mathbf{1} - \boldsymbol{x}$).

5 Résolution d'un système de Horn

5.1 Complétude de la relaxation linéaire

Soit $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ la transcription linéaire d'un système de clauses de Horn H.
Posons $Q = \{\boldsymbol{x} \in \{0,1\}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\} = \{\boldsymbol{x} \in \mathbb{Z}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\ ,\ \mathbf{0} \leq \boldsymbol{x} \leq \mathbf{1}\}$.

Par définition, H est satisfaisable si et seulement si $Q \neq \emptyset$.

La relaxation linéaire continue de H dans l'hypercube unité définit le polytope $P = \{\boldsymbol{x} \in [0, 1]^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\ \} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\ ,\ \mathbf{0} \leq \boldsymbol{x} \leq \mathbf{1}\}$.

Théorème V.1 *Le système de Horn* H *est satisfaisable si et seulement si le polytope* P *est non vide, auquel cas il admet un plus petit élément qui vérifie :* min P = min Q.

En effet, $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\ ,\ \mathbf{0} \leq \boldsymbol{x} \leq \mathbf{1}$ est un système 1-monotone et d'après le corollaire du théorème III.3, la condition de satisfaisabilité $Q \neq \emptyset$ équivaut à $P \neq \emptyset$.

La seconde partie de la proposition découle du théorème III.4. ■

Tout algorithme de résolution de systèmes linéaires (le simplexe notamment) est donc à même de résoudre la question de la satisfaisabilité d'un système de clauses de Horn.

Remarque Pour obtenir, par optimisation, la solution particulière min P, il suffit de prendre pour fonction à minimiser la somme des variables. Minimiser chacun des x_i revient à minimiser leur somme : $\min P = \min \{\mathbf{1}.\boldsymbol{x} \mid \boldsymbol{x} \in P\}$.

Lorsque H est satisfaisable, la solution min P est un modèle de H. C'est le *plus petit modèle* de H pour l'ordre correspondant à la structure de treillis booléen.

Corollaire *Tout système de Horn admet, lorsqu'il est satisfaisable, un plus petit modèle.*

5.2 Complétude de la relaxation dans l'approximation des intervalles

La résolution par réduction d'intervalles du système linéaire relâché définit, dans l'espace discrétisé $(\mathbb{R}, \Im_M)$, un sous-pavé R de l'hypercube unité.

Ces relaxations successives donnent la chaîne d'inclusions : $Q \subset P \subset R$.

La proposition suivante (pour laquelle on suppose $M \gg 1$) découle du théorème IV.4 :

Théorème V.2 *Le système de Horn* H *est satisfaisable si et seulement si le pavé* R *des domaines réduits des variables est non vide, auquel cas il admet un plus petit élément,* min R*, à coordonnées entières.*

On a donc en définitive : $Q \neq \emptyset \Leftrightarrow P \neq \emptyset \Leftrightarrow R \neq \emptyset$.

Le plus petit modèle de H est la solution particulière obtenue en prenant le minimum du domaine réduit de chacune des variables.

Remarque Les formules de réduction des domaines qui découlent de l'inégalité (1) s'écrivent :

$$(1 - |J^-|) + \sum_{i \in J^-} \min(X_j) \leq \delta_k \min(X_k) \quad (1.1)$$

$$\max(X_j) \leq \delta_k \max(X_k) + (|J^-| - 1) - \sum_{l \in J^- - \{i\}} \min(X_l) \quad (1.2)$$

Partant de domaines égaux à [0,1], les réductions d'intervalles n'opèrent ici que par propagation de bornes entières — puisqu'il n'y a pas de division — par conséquent égales à 0 ou à 1[1]. Tant que le système reste satisfaisable, le domaine réduit X_j d'une variable x_j n'a que 3 trois formes possibles : [0,1], {0} ou {1}.

Cette remarque reste valable pour tout système de clauses en général : les domaines réduits ont toujours un minimum et un maximum entier. Au niveau de la réduction des domaines, l'ajout de contraintes d'intégrité $x_j \in \mathbb{Z}$ n'apporte rien de plus. Il n'y a donc pas de différence, pour les booléens, entre domaines finis et intervalles.

5.3 Application à la vérification de cohérence de règles

La cohérence d'une base de règles est fondamentale pour un système expert. Vérifier cette cohérence revient à statuer sur un problème «Horn-SAT». Pour ce faire, on a le choix des solveurs (Simplexe ou Intervalles) .

Exemple 1 Extrait d'un rapport d'analyse chimique dont on se demande s'il est cohérent[2]:

- En présence de cobalt et en l'absence de nickel, une couleur verte apparaît.
- On n'observe ni présence de nickel, ni présence de manganèse.
- On observe une présence de cobalt, mais seule une couleur brune apparaît.

En notant *c*, *n*, *m* les propositions *présence de cobalt*, *absence de nickel*, *absence de manganèse*, *v* et *b* les propositions (antinomiques) *couleur de réaction verte* et *couleur de réaction brune*, on écrit :

Clauses booléennes		Contraintes linéaires		
$c \wedge n$	$\Rightarrow v$	$c + n$	$\leq$	$v + 1$
	$\Rightarrow n$	1	$\leq$	n
	$\Rightarrow m$	1	$\leq$	m
	$\Rightarrow c$	1	$\leq$	c
	$\Rightarrow b$	1	$\leq$	b
$b \wedge v$	$\Rightarrow$	$b + v$	$\leq$	1

1. Nous en donnerons une interprétation plus loin avec la résolution unitaire.
2. Exemple inspiré de W. Hodges.

Ce système de clauses est incohérent puisqu'on en tire à la fois v et $\neg v$ (soit $v \geq 1$ et $v \leq 0$). Il est impossible à satisfaire.

Confirmation par programme (en Prolog IV) :

Relaxation linéaire (problème P) :

```
>>  C gelin 0, C lelin 1, N gelin 0, N lelin 1, M gelin 0, M lelin 1,
    V gelin 0, V lelin 1, B gelin 0, B lelin 1,
    pluslin(C, N)  lelin  pluslin(V, 1),
    1 lelin N,
    1 lelin M,
    1 lelin C,
    1 lelin B,
    pluslin(B, V)  lelin  1.

false.
```

Relaxation au sens des intervalles (problème R) :

```
>>  C ge 0, C le 1, N ge 0, N le 1, M ge 0, M le 1,
    V ge 0, V le 1, B ge 0, B le 1,
    plus(C, N)  le  plus(V, 1),
    1 le N,
    1 le M,
    1 le C,
    1 le B,
    plus(B, V)  le  1.

false.
```

Exemple 2 Considérons maintenant le rapport d'analyses modifié :

- En présence de cobalt et en l'absence de nickel, une couleur verte apparaît.
- On observe une présence de nickel ou de manganèse.
- On observe une présence de cobalt, mais seule une couleur brune apparaît.

On a toujours un système de Horn :

Clauses booléennes		Contraintes linéaires	
$c \wedge n$	$\Rightarrow v$	$c + n$	$\leq v + 1$
$m \wedge n$	$\Rightarrow$	$m + n$	≤ 1
	$\Rightarrow c$	1	$\leq c$
	$\Rightarrow b$	1	$\leq b$
$b \wedge v$	$\Rightarrow$	$b + v$	≤ 1

Ces affirmations sont cohérentes. En particulier, le système a pour plus petit modèle $(b, c, m, n, v) = (1, 1, 0, 0, 0)$. Et l'on peut même conclure, par les deux relaxations, à la présence de nickel ($n = 0$).

Relaxation linéaire :

```
>>  C gelin 0, C lelin 1, N gelin 0, N lelin 1, M gelin 0, M lelin 1,
    V gelin 0, V lelin 1, B gelin 0, B lelin 1,
    pluslin(C, N)  lelin  pluslin(V, 1),
    pluslin(M, N)  lelin  1,
    1 lelin C,
    1 lelin B,
    pluslin(B, V)  lelin  1.

B = 1, V = 0, N = 0, C = 1, M ~ real.        % 0 lelin M, M lelin 1.
```

Relaxation au sens des intervalles :

```
>>  C ge 0, C le 1, N ge 0, N le 1, M ge 0, M le 1,
    V ge 0, V le 1, B ge 0, B le 1,
    plus(C, N)  le  plus(V, 1),
    plus(M, N)  le  1,
    1 le C,
    1 le B,
    plus(B, V)  le  1.

B = 1, V = 0, N = 0, C = 1, M ~ cc(0,1).     % 0 le M, M le 1.
```

Il se trouve ici que les deux valeurs de la proposition indéterminée *absence de manganèse* ($m = 0$ et $m = 1$) sont solution[1]. Les projections données par les intervalles sont donc exactes. Pour un système de Horn en général, seule la plus petite valeur est garantie pour solution.

Noter que si l'on avait pris comme variables propositionnelles *présence de nickel* et *présence de manganèse*, le système de clauses ne serait plus un système de Horn, mais un système *renommable* en un système de Horn.

Avec les propositions *présence de nickel* et *absence de couleur verte*, le système de clauses reste de Horn :

Clauses booléennes		Contraintes linéaires	
$c \wedge v'$	$\Rightarrow n'$	$c + v'$	$\leq n' + 1$
m	$\Rightarrow n'$	m	$\leq n$
	$\Rightarrow c$	1	$\leq c$
	$\Rightarrow b$	1	$\leq b$
b	$\Rightarrow v'$	b	$\leq v'$

Son plus petit modèle est $(b, c, m, n', v') = (1, 1, 0, 1, 1)$, ce qui explique les bornes supérieures trouvées[2] pour n et v : $\max(n) = 1 - \min(n') = 0$.

1. On peut s'en assurer en lançant un `enum` ou `enumlin(M)` en Prolog IV.
2. Nous verrons que les propriétés de complétude de la relaxation linéaire et de la relaxation au sens des intervalles s'étendent aux systèmes Horn-renommables.

6 De Fourier à la résolution unitaire

Puisque la satisfaisabilité d'un système de Horn se ramène à la résolution d'un programme linéaire, les problèmes Horn-SAT sont polynômialement solubles. Nous allons montrer, en dérivant du schéma d'élimination de Fourier un algorithme spécifique, que la complexité de ce problème de décision est linéaire par rapport à la taille du système donné en entrée (mesurée au nombre de littéraux apparaissant dans les clauses). L'algorithme obtenu a un nom : la *résolution unitaire*.

On supposera que le système de départ H_0, défini par $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$, $\mathbf{0} \leq \boldsymbol{x} \leq \mathbf{1}$, ne comporte pas de clause vide qui le rendrait immédiatement insatisfaisable (il suffit pour cela d'exiger que chaque contrainte fasse au moins intervenir une variable). On rappelle qu'il y a équivalence, pour un système 1-monotone, entre élimination sur les réels et les entiers.

6.1 Simplifications liminaires

Au niveau d'un système, les clauses redondantes peuvent être ignorées (certaines sont triviales) — et l'élimination peut s'abstenir d'en produire.

Proposition *Si un système clausal contient un fait portant sur la variable propositionnelle x_0, alors ce fait rend redondantes toutes les règles donnant x_0 en conclusion.*

En effet, $\sum_{j \in J^-} x_j \leq x_0 + \sum_{j \in J^{+*}} x_j + (|J^-| - 1)$ est la somme des inégalités $1 \leq x_0$, $\sum_{j \in J^-} x_j \leq |J^-|$ et $0 \leq \sum_{j \in J^{+*}} x_j$ (qui découlent des encadrements sur les x_j). ■

Proposition *Lors de l'élimination d'une variable propositionnelle, il est inutile de prendre en compte les inégalités tirées du sous-système $\mathbf{0} \leq \boldsymbol{x} \leq \mathbf{1}$.*

Considérons en effet une paire d'inégalités impliquant le sous-système $\boldsymbol{x} \leq \mathbf{1}$:

$C_1 : x_0 \leq 1$ et $C_2 : \sum_{j \in J^-} x_j \leq x_0 + \sum_{j \in J^{+*}} x_j + (|J^-| - 1)$.

Alors l'inégalité résultant de l'élimination de x_0 est trivialement redondante :

$C_1 + C_2 : \sum_{j \in J^-} x_j \leq \sum_{j \in J^{+*}} x_j + |J^-|$.

Même chose pour les paires d'inégalités impliquant le sous-système $\mathbf{0} \leq \boldsymbol{x}$. ■

6.2 Algorithme de résolution unitaire positive

Définition Une *clause unitaire* est une clause réduite à un seul littéral, positif (x) ou négatif ($\neg x$).

Un fait correspond donc à une clause unitaire positive.

Proposition *Soit* S *un système de clauses comprenant un fait portant sur une variable* x_0. *Alors le système* $E_F(S, x_0)$ *issu de l'élimination de Fourier de la variable* x_0 *est également un système clausal.*

En effet, les seules paires d'inégalités complémentaires à considérer sont celles comprenant l'inégalité $C : 1 \leq x_0$ (toute autre inégalité donnant en x_0 conclusion est redondante). Soit donc D une clause de S de la forme :

$$x_0 + \sum_{i \in J^{-*}} x_j \leq \sum_{i \in J^+} x_j + (|J^{-*}| + 1 - 1) \text{ avec } k \neq 0 \text{ et } k \notin J^-.$$

$$C + D : \sum_{i \in J^{-*}} x_j \leq \sum_{i \in J^+} x_j + (|J^{-*}| - 1) \text{ a la structure syntaxique d'une clause.}$$ ■

Remarque Dans le nouveau système, toutes les contraintes où x_0 n'apparaît pas sont reconduites (y compris donc les encadrements des autres variables). Les clauses où x_0 apparaissait en conclusion sont effacées. Ne subsistent par ailleurs que des clauses «raccourcies», où x_0 apparaissait précédemment en prémisse.

Proposition *Un système de Horn qui ne comprend pas de fait (et ne contient pas de clause vide) admet une solution triviale* $\boldsymbol{x} = \mathbf{0}$.

Dans ce cas en effet, les clauses sont de la forme : $\sum_{i \in J^-} x_j \leq \delta_k x_k + (|J^-| - 1)$

avec $|J^-| \geq 1$ (chaque clause possédant au moins un littéral négatif).
On a donc toujours : $0 \leq |J^-| - 1$. ■

A partir d'un système de Horn, H_0, on construit par éliminations successives une suite $(H_k)_{k \in K \subset J}$ de systèmes de Horn, de la façon suivante.

Tant qu'il existe dans H_k une clause unitaire positive x_j, calculer :

$$H_{k+1} = E_F(H_k, x_j)$$

L'algorithme d'élimination s'arrête à une étape f comprise entre 0 et n (si n désigne le nombre de variables du système initial). Il produit un système de Horn, H_f, ne contenant plus de fait et équivalent à H_0 du point de vue de la satisfaisabilité.

L'étude de cas suivante permet de conclure sur la satisfaisabilité de H_f (et donc de H_0) :

1. Soit le système H_f est vide
   et dans ce cas, le système H_0 est satisfaisable.
2. Soit le système H_f contient la clause vide ($1 \leq 0$)
   et dans ce cas, le système H_0 est insatisfaisable.
3. Soit le système H_f est non vide et exempt de clause vide
   et d'après la proposition précédente, le système H_0 est satisfaisable.

Définition La *résolution unitaire positive* correspond à la procédure d'élimination de Fourier itérée sur des variables intervenant dans des clauses unitaires positives.

L'étude de cas vient de montrer :

Théorème V.3 *Un système de Horn est insatisfaisable si et seulemement si sa résolution unitaire positive produit la clause vide.*

Exemple En reprenant le système de clauses de l'exemple 1 :

l'élimination de la variable b donne : $v \Rightarrow$ (soit $v \leq 0$) ;
l'élimination de la variable c donne : $n \Rightarrow v$;
l'élimination de la variable n donne : $\Rightarrow v$ (soit $v \geq 1$) ;
enfin, l'élimination de la variable v donne la clause vide : $1 \leq 0$.

Pour le système de clauses de l'exemple 2 :

l'élimination de la variable b donne : $v \Rightarrow$;
l'élimination de la variable c donne : $n \Rightarrow v$;
puis l'élimination s'arrête, puisqu'il n'y a plus de faits. On est dans le cas 3 : le système est satisfaisable et son plus petit modèle est obtenu en affectant à 0 les variables restantes : m, n, v.

6.3 Complexité

Chaque élimination de variable propositionnelle entraîne nécessairement la radiation de clauses ayant un littéral positif (lignes) et éventuellement d'occurrences de littéraux négatifs dans les autres (colonnes). La taille du système décroissant strictement, l'algorithme est clairement polynômial.

Dowling & Gallier (1984) montrent qu'avec un codage de type matriciel à double chaînage (des variables vers les clauses et des clauses vers les variables) on peut rendre les opérations de mise à jour du système, de H_k en H_{k+1}, peu coûteuses et obtenir un algorithme linéaire en fonction du nombre total de littéraux apparaissant dans les clauses. La mise en place de compteurs des occurrences de littéraux négatifs dans chaque règle est un élément clé pour éviter de scanner inutilement la base et obtenir cette complexité linéaire. Ces compteurs ne sont autres que les coefficients constants des inégalités : $|J^-| = 0$ signale un fait déduit, prêt à être propagé.

6.4 Interprétation graphique

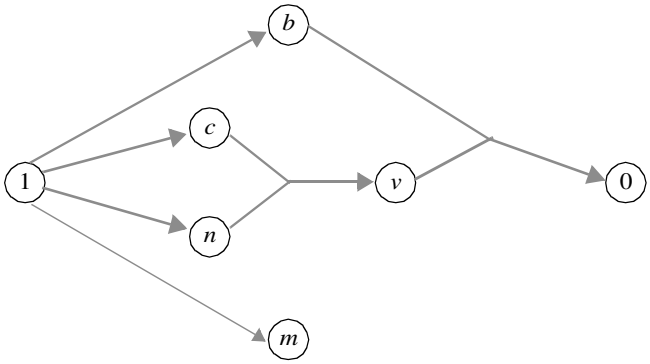
Pour décrire la structure particulière d'un système de Horn, il est intéressant d'introduire la notion d'hypergraphe.

Définition Un *hypergraphe* (N, A) est constitué d'un ensemble fini de sommets ou nœuds), N, et d'un ensemble, A, de parties de N appelées arcs (ou hyperarcs).
Un *sommet* est dit *incident* à un arc lorsqu'il appartient à cet arc.
Un hypergraphe est *orienté* lorsqu'on distingue, dans un arc, les sommets de départ des sommets d'arrivée.

La représentation du problème par un hypergraphe dont les hyperarcs (clauses de Horn) ne pointent que vers un seul sommet (variable propositionnelle ou, à défaut, 0) conduit à une interprétation de la résolution unitaire en termes de *data flow*.

Un système de Horn est insatisfaisable s'il existe une suite d'arcs dans l'hypergraphe qui «propage» la valeur vraie du sommet 1 jusqu'au sommet 0. Le test de la satisfaisabilité d'un système de Horn s'apparente ainsi à la recherche d'un «chemin» entre le vrai et le faux — qui ferait court-circuit.

Exemple Le schéma suivant restitue la structure d'hypergraphe de l'exemple 1, où l'on voit que 0 est accessible à partir de 1 ($1 \Rightarrow 0$).



L'algorithme de Dowling &Gallier correspond à un parcours en largeur d'abord de l'hypergraphe, dans lequel chaque nœud est visité au plus une fois.

Remarque Dans les systèmes experts, le «chaînage avant» s'effectue selon le même principe :

On part de la base de faits initiale.
On déclenche toutes les règles dont les prémisses sont satisfaites.
On ajoute les faits obtenus.
Et l'on poursuit ainsi jusqu'à saturation.

La procédure de chaînage avant implémente la résolution unitaire. Elle est donc complète pour les systèmes de Horn.
Son principe d'inférence est une forme d'hyper-résolution (Robinson 1965) :

$$\Rightarrow x_1$$
$$\ldots$$
$$\Rightarrow x_n$$
$$x_1 \wedge \ldots \wedge x_n \Rightarrow x_{n+1}$$
$$_____$$
$$\Rightarrow x_{n+1}$$

6.5 Transparence des résultats

Les systèmes experts accordent une place considérable à l'explication du raisonnement et à la justification des conclusions — ce que les méthodes numériques généralement occultent.

Néanmoins, il possible, grâce à la théorie de la dualité, de fournir l'explication d'un calcul. C'est le cas, en particulier, lorsque le programme primal $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$, $\mathbf{0} \leq \boldsymbol{x} \leq \mathbf{1}$ affecte la valeur d'une variable x_j à 0 (resp. à 1). Alors $x_j \leq 0$ (resp. $x_j \geq 1$) est conséquence logique du système de clauses et la résolution du programme dual donne une description détaillée de la preuve.

En notant $\boldsymbol{e}_j$ le $j$ième vecteur de la base canonique de $\mathbb{R}^n$, ce programme s'écrit : $\boldsymbol{y}\mathbf{A} = -\boldsymbol{e}_j$, $\boldsymbol{y} \geq 0$, $\boldsymbol{by} \geq 0$ (resp. $\boldsymbol{y}\mathbf{A} = \boldsymbol{e}_j$, $\boldsymbol{y} \geq 0$, $\boldsymbol{by} \geq 1$).

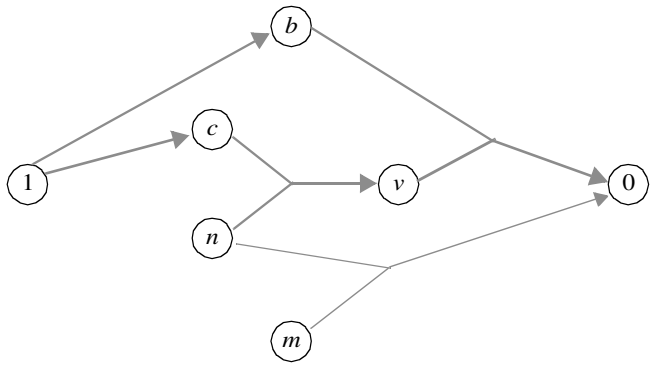
Exemple La matrice d'incidence de l'exemple 2, pour le sous-système $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$, $\mathbf{0} \leq \boldsymbol{x}$, a la forme suivante :

b	c	m	n	v
	−		−	+
		−	−	
	+			
+				
−				−
+				
	+			
		+		
			+	
				+

$\geq$

−1	y_1 √
−1	y_2
+1	y_3 √
+1	y_4 √
−1	y_5 √
0	z_1
0	z_2
0	z_3
0	z_4
0	z_5

Exemple (1,0,1,1,1,0,0,0,0,0) est une solution particulière (en nombres entiers) du problème dual qui explique la conclusion $n = 0$. La figure suivante fait apparaître en gras les arcs de l'hypergraphe sélectionnés.



L'explication se résume à un chemin reliant n à 0 : $n \Rightarrow v \Rightarrow 0$, ce qui donne $n \leq v \leq 0$.

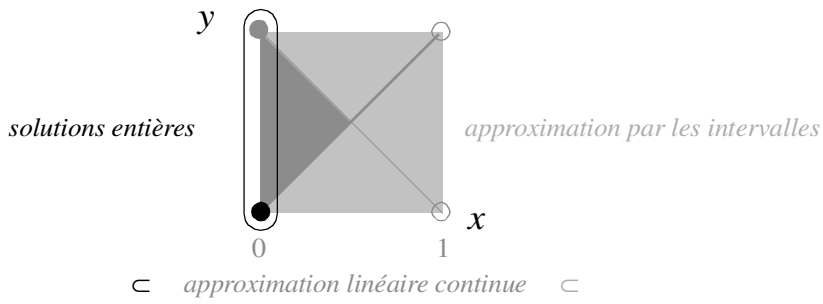
6.6 Calcul des projections exactes

Lorsque le système de Horn H est satisfaisable, la résolution unitaire positive — comme d'ailleurs la relaxation linéaire et la réduction d'intervalles — exhibe une solution particulière (le plus petit modèle de H) mais n'explicite pas les projections de l'ensemble des solutions. On n'est donc pas assuré que les variables «figées» à 0 soient détectées.

Exemple Prenons pour H le système formé des deux clauses de Horn suivantes :

Clauses booléennes	Contraintes linéaires
$x \Rightarrow y$	$x \leq y$
$x \wedge y \Rightarrow$	$x + y \leq 1$

H admet pour plus petit modèle le couple $(x, y) = (0, 0)$. La résolution linéaire positive n'effectue aucune inférence, puisqu'il n'y a pas de fait. La relaxation par les intervalles donne pour pavé des domaines réduits : $[0,1]\times[0,1]$. La relaxation linéaire donne une approximation plus fine ($x \leq 1/2$) — qui délimite cette fois les solutions entières, sans pour autant conclure à $x = 0$.



Pour calculer les projections exactes, il suffit de tester l'affectation à 1 de toute variable x_j ayant 0 pour minimum. Le système $H \cup \{x_j \geq 1\}$ restant de Horn, le test de satisfaisabilité peut être effectué par l'un quelconque des trois algorithmes étudiés. Lorsqu'il est positif, 0 et 1 sont solutions. Lorsqu'il est négatif, x_j est figée à 0.

Dans son ensemble, la procédure reste polynômiale.

6.7 Algorithme de résolution unitaire

La *résolution unitaire* généralise la résolution unitaire positive en réservant un traitement symétrique à toutes les clauses unitaires, y compris négatives. Elle permet de résoudre des systèmes de clauses qui, s'ils ne sont pas syntaxiquement de Horn, peuvent le devenir par inversion de certains axes.

Le procédé d'élimination de Fourier est juste poussé un peu plus loin, l'étape de base devenant :

Tant qu'il existe dans H_k une clause unitaire x_j ou $\neg x_j$, calculer :

$$H_{k+1} = E_F(H_k, x_j)$$

On construit ainsi une suite de systèmes équivalents du point de vue de la satisfaisabilité et renommables en systèmes de Horn :

- L'inégalité $1 \leq x_j$ rend redondantes toutes les autres clauses où x_j intervient en conclusion : seules les seules clauses où x_j intervient en prémisse sont alors à prendre en compte pour l'élimination de x_j.
- L'inégalité $x_j \leq 0$ rend redondantes toutes les autres clauses où x_j intervient en prémisse : seules les clauses où x_j intervient en conclusion sont alors à prendre en compte pour l'élimination de x_j.
- Chaque élimination produit, par suppression des littéraux portant sur x_j, une sous-clause de H_k.

Donc si H_k est un système de clauses Horn-renommable, H_{k+1} reste de Horn pour le même renommage.

L'étude de cas finale est inchangée (elle porte désormais sur un système, H_f, ne contenant plus de clause unitaire) :

1. Si le système H_f est vide :
   le système initial est satisfaisable.
2. Si le système H_f contient la clause vide ($1 \leq 0$) :
   le système initial est insatisfaisable.
3. Si le système H_f est non vide et exempt de clause vide :
   le système initial est satisfaisable.

La complexité de l'algorithme passe de linéaire à quadratique[1].

Propriété *Un système Horn-renommable est insatisfaisable si et seulement si sa résolution unitaire produit la clause vide.*

Remarque Avec les intervalles, il y a toujours, pour chaque clause, une formule de réduction d'intervalles qui redonne la formule **(1.1)** que l'on obtiendrait avec le bon renommage — et dont l'effet, en termes de propagation, est celui de la résolution unitaire positive[2].

Résolution unitaire, relaxation linéaire et approximation par les intervalles ont un pouvoir comparable :

- elles sont complètes pour les systèmes Horn-renommables
- elle sont incomplètes plus généralement.

1. En $O(|V|.|S|)$ avec $|V|$ = nombre de variables et $|S|$ = nombre total de littéraux dans le système de clauses.
2. Formellement, un renommage est un changement de variable, de $\boldsymbol{x}$ en $\boldsymbol{x'}$, dans lequel, selon le cas, $x'_j = x_j$ ou $1–x_j$. Lorsque l'axe est inversé, les bornes le sont également : $\min(X'_j) = 1 – \max(X_j)$ et $\max(X'_j) = 1 – \min(X_j)$.

Exemple La résolution unitaire ne détecte pas l'insatisfaisabilité du système de clauses à 2 variables : $(x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$.

Propriété *La relaxation linéaire continue d'un système de clauses non vide, exempt de clause vide et sans clause unitaire, a toujours* **1/2** *pour solution.*

Chaque clause $\sum_{i \in J^-} x_j \leq \sum_{i \in J^+} x_j + (|J^-| - 1)$ donne $\frac{1}{2}|J^-| \leq \frac{1}{2}|J^+| + (|J^-| - 1)$, soit $1 \leq \frac{1}{2}|J|$, puisque $|J| = |J^+| + |J^-|$. Comme par hypothèse, ces clauses ont au moins 2 variables, $|J| \geq 2$ et **1/2** est solution. ■

7 Résolution dans le cas général

Les systèmes de Horn se résolvent facilement mais leur expressivité est limitée (l'information négative n'est guère prise en compte). Or on a souvent besoin d'exprimer des règles comme $x \wedge \neg y \wedge z \Rightarrow t$, autrement dit, des clauses $x \wedge z \Rightarrow y \vee t$.

Passer à SAT dans toute sa généralité à partir de Horn-SAT, c'est quitter une classe polynômiale pour effectuer un grand saut dans l'échelle de la complexité. Comme on l'a vu, l'existence de solutions réelles n'est plus ici garante de l'existence de solutions entières.

Nous allons montrer qu'avec des coupes appropriées, l'élimination de Fourier des variables entières aboutit au principe de résolution — dans sa forme propositionnelle. Ce rapprochement entre méthodes logiques et numériques est, entre autres, dû à Williams.

Il est toujours licite d'éliminer par Fourier une variable x_0 d'un système de clauses S. Cela donne pour toute paire de clauses complémentaires :

$$C_1 : \quad x_0 + \sum_{i \in J^-} x_j \leq \sum_{i \in J^+} x_j + ((|J^-| + 1) - 1)$$

$$C_2 : \quad \sum_{i \in K^-} x_j \leq x_0 + \sum_{i \in K^+} x_j + (|K^-| - 1)$$

$$C_1 + C_2 : \sum_{i \in J^-} x_j + \sum_{i \in K^-} x_j \leq \sum_{i \in J^+} x_j + \sum_{i \in K^+} x_j + (|J^-| + |K^-| - 1)$$

En revanche, d'un point de vue syntaxique, le système d'inégalités obtenu n'est plus uniquement formé de clauses. Le problème vient de la présence éventuelle d'occurrences multiples de variables dans l'inégalité résultante. Ce problème se simplifie en logique grâce à l'idempotence de la disjonction ($x \vee x = x$) et à la loi du tiers exclu ($x \vee \neg x = 1$).

Propriété *Le système* $E_F(S, x_0)$ *est équivalent sur les entiers à un système de clauses* S'.

Démonstration En notant : $J = J^+ \cup J^-$ et $K = K^+ \cup K^-$, l'inéquation $C_1 + C_2$ se réécrit :

$$\sum_{J^- - K} x_j + 2\left(\sum_{J^- \cap K^-} x_j\right) + \sum_{K^- - J} x_j \leq \sum_{J^+ - K} x_j + 2\left(\sum_{J^+ \cap K^+} x_j\right) + \sum_{K^+ - J} x_j + |J^-| + |K^-| - 1$$

En rajoutant des inégalités de la forme : $0 \leq \sum_{i \in E} x_j$ et : $\sum_{i \in E} x_j \leq |E|$ qui résultent des encadrements des variables, on obtient l'inégalité :

$C_3 : 2\left(\sum_{i \in L^-} x_j\right) \leq 2\left(\sum_{i \in L^+} x_j\right) + l$ où $l = |J^- - K| + |K^- - J| + |J^-| + |K^-| - 1$, avec

$L^- = (J^- - K) \cup (J^- \cap K^-) \cup (K^- - J)$,

$L^+ = (J^+ - K) \cup (J^+ \cap K^+) \cup (K^+ - J)$.

On ne change par les solutions du système d'inégalités en remplaçant $C_1 + C_2$ par C_3.

Comme $J^+ \cap J^- = K^+ \cap K^- = \emptyset$, on a :

$|J^-| = |J^- - K| + |J^- \cap K^+| + |J^- \cap K^-|$,

$|K^-| = |K^- - J| + |J^+ \cap K^-| + |J^- \cap K^-|$, et le terme constant vaut :

$$\begin{aligned} l &= 2\,|J^- - K| + 2\,|J^- \cap K^-| + 2\,|K^- - J| + |J^- \cap K^+| + |J^+ \cap K^-| - 1 \\ &= 2\,|L^-| + |J^- \cap K^+| + |J^+ \cap K^-| - 1 \end{aligned}$$

Par conséquent :

- si $J^- \cap K^+ \neq \emptyset$ ou $J^+ \cap K^- \neq \emptyset$, l'inégalité obtenue est redondante.
- sinon $J^- \cap K^+ = J^+ \cap K^- = \emptyset$, et l'inégalité obtenue est une clause[1] ou se transforme en clause par une simple coupe en nombres entiers ($-1 = \left\lfloor -\frac{1}{2} \right\rfloor$) :

$$C'_3 : \sum_{i \in L^-} x_j \leq \sum_{i \in L^+} x_j + |L^-| - 1 \quad < \sum_{i \in L^+} x_j + |L^-| - \frac{1}{2}\ .$$

On peut donc toujours ramener le système d'inégalités $E_F(S, x_0)$ à un système de clauses S' qui lui est équivalent sur les entiers. ■

Définition La clause C'_3 obtenue par élimination entière de x_0 entre les inégalités C_1 et C_2 est appelée *résolvante* des clauses C_1 et C_2.

Comme $E_F(S, x_0)$ est équivalent à S du point de vue de la satisfaisabilité, S est satisfaisable sur les entiers si et seulement si S' est satisfaisable sur les entiers.

On en déduit le théorème de *ground resolution* :

Théorème V.4 *Un système de clauses est insatisfaisable si et seulement si la procédure de résolution produit la clause vide.*

1. Lorsque $J \cap K$ est vide.

La procédure de résolution en logique propositionnelle peut donc être vue comme une variante en nombres entiers de l'élimination de Fourier dans laquelle les inégalités produites sont, à chaque étape, systématiquement resserrées.

La résolution se fonde ainsi sur une règle d'inférence unique qui généralise le *Modus Ponens* :

$$\frac{\bigwedge_{K^-} y_j \Rightarrow x \vee \bigvee_{K^+} z_j \qquad x \wedge \bigwedge_{J^-} y_j \Rightarrow \bigvee_{J^+} z_j}{\bigwedge_{J^- \cup K^-} y_j \Rightarrow \bigvee_{J^+ \cup K^+} z_j}$$

8 De la résolution à l'algorithme de Davis & Putnam

Historiquement, l'introduction du principe de résolution remonte, pour la logique propositionnelle, à l'algorithme de Davis & Putnam. Tout le mérite de Robinson (1965) est de l'avoir généralisé à la logique du premier ordre.

Il y a en fait deux versions de l'algorithme de Davis & Putnam. La première est celle de 1960 qui utilise la résolution comme procédé d'élimination. La similitude avec l'analyse de Fourier y est frappante. La plus connue est la seconde, dite de Davis-Putnam-Loveland. Elle date de 1962. Elle conserve la résolution unitaire mais remplace l'élimination, dans le cas général, par une procédure de séparation et d'évaluation.

8.1 Version originale

Every formula F *in conjunctive normal form has the structure* A ∧ B ∧ R *where* A *is the conjunction of clauses containing a given atomic formula (say, p),* B *is the conjunction of clauses containing the negation of that formula (say, ¬p), and* R *is the conjunction of the remaining clauses.*

Comme pour Fourier, le choix d'une variable partitionne le système en 3. La règle d'élimination de p entre les clauses de A et celles de B est énoncée comme suit. Elle synthétise en une formule toute la combinatoire de la résolution. L'algorithme exige qu'à chaque étape, cette formule soit remise sous forme normale conjonctive.

Théorème V.5 *Supposons* F *mise sous la forme* (A' ∨ p) ∧ (B' ∨ ¬p) ∧ R *où* A' *est obtenue à partir de* A *en supprimant les occurrences de p et* B' *est obtenue à partir de* B *en supprimant les occurrences de ¬p. Alors* F *est insatisfaisable si et seulement si* (A' ∨ B') ∧ R *est insatisfaisable.*

La justification est la suivante. Soient C_k une clause de A' et D_l une clause de B'. La paire de clauses de F ($C_k \vee p$, $D_l \vee \neg p$) a pour résolvante $C_k \vee D_l$.

Par distributivité, le système formé par les résolvantes produites s'écrit :

$$A' \vee B' = \bigwedge_{k,\,l} (C_k \vee D_l) \quad .$$

F est donc satisfaisable si et seulement si le système $(A' \vee B') \wedge R$ résultant de l'élimination de p l'est aussi. ■

La procédure originale de Davis & Putnam consiste en une élimination de Fourier en variables entières, dans laquelle l'ordre d'élimination des variables est partiellement prescrit :

1. En premier, viennent les variables candidates à la résolution unitaire.
2. En second, les variables n'apparaissant que dans des littéraux positifs (ou négatifs), avec lesquelles on ne peut former de paire complémentaire.
3. En dernier, les autres variables — auxquelles s'applique la règle générale.

8.2 Version énumérative

La seconde version de l'algorithme de Davis & Putnam conserve le même ordre de prise en compte des variables mais remplace la règle 3 par une règle dite de *splitting*. Celle-ci décompose l'étude de satisfaisabilité en deux cas : $p = 0$ ou $p = 1$.

Propriété *Avec les notations précédentes,* F *est insatisfaisable si et seulement si les systèmes* $A' \wedge R$ *et* $B' \wedge R$ *sont tous deux insatisfaisables.*

En effet, $(A' \vee B') \wedge R = (A' \wedge R) \vee (B' \wedge R)$. ■

L'algorithme construit un arbre de décision binaire en suivant une stratégie intelligente qui limite l'énumération — grâce au flitrage par la résolution unitaire et l'élimination des littéraux purs. Aujourd'hui toujours, cette technique mixte de séparation et d'élimination reste l'une des plus performantes pour le problème SAT.

9 Autres méthodes énumératives

Nous donnons pour conclure deux autres méthodes de résolution des problèmes SAT. La première exige une reformulation préalable du système de clauses pour mettre en évidence la structure sous-jacente d'un sous-système de Horn. La seconde s'émancipe de la structure même de clause, laissant au programmeur le choix de la décomposition d'une formule en relations élémentaires dont la gestion est déléguée au mécanisme général de réduction d'intervalles.

9.1 Relaxation reposant sur un système de Horn

Pour obtenir une relaxation de type Horn, il suffit de laisser tomber toutes les clauses qui ne sont pas de Horn . La satisfaisabilité du sous-système obtenu, aisément vérifiable, est une condition nécessaire à la satisfaisabilité du système complet de clauses.

Remarque L'étape initiale de filtrage par la résolution unitaire dans l'algorithme de Davis & Putnam peut être considérée comme une relaxation de Horn : elle échoue dès qu'un sous-système Horn-renommable s'avère insatisfaisable. En revanche, l'algorithme ne prévoit pas d'arrêt anticipé à la fin du filtrage — quand le système testé est un système de Horn.

Gallo & Urbani (1989) présentent deux techniques pour exploiter au mieux les «bonnes propriétés» des systèmes de Horn et accélérer la résolution d'un système de clauses, S (lorsque celui-ci n'est pas directement renommable en un système de Horn[1]).

Syntaxiquement, S est transformé en un système équivalent, $H \cup \overline{H}$, où H est un système de Horn qui va servir de support à la relaxation. Dans ce schéma, l'énumération peut se limiter aux variables de $\overline{H}$ (la décomposition est choisie de sorte à en minimiser le nombre).

Avec q variables dans $\overline{H}$ sur n variables au total, la complexité chute d'un facteur 2^{n-q}, passant de $O(2^n)$ *a priori* pour S à $O(2^q)$ pour $\overline{H}$. A chaque instanciation d'une variable de $\overline{H}$, la satisfaisabilité de H doit être re-vérifiée, introduisant un facteur multiplicatif constant, proportionnel à la taille de H.

Considérons une clause de S, $C : x_1 + x_2 + ... + x_k \leq y_1 + y_2 + ... + y_l + (k–1)$. C n'est pas de Horn si $l \geq 2$.

9.1.1 Premier modèle de relaxation

La première méthode, assez intuitive, consiste à transformer les littéraux positifs excédentaires en littéraux négatifs.
Elle renomme l–1 des l variables y_j en y'_j, pour obtenir une clause de Horn $C' : x_1 + x_2 + ... + x_k + y'_2 + ... + y'_l \leq y_1 + (k+l–2)$. Plus exactement, elle remplace chaque clause C par le système formé de C' et d'au plus l–1 équations de couplage : $y'_j = 1 – y_j$ (requises dès lors que les y_j ont des occurrences positives dans d'autres clauses). Ces égalités se décomposent en deux inégalités : $y_j + y'_j \leq 1$ (exclusion) et : $y_j + y'_j \geq 1$ (disjonction).

Les clauses d'exclusion sont des clauses de Horn. Elles viennent s'ajouter à la reformulation des clauses du système pour constituer le système H.
La complexité combinatoire de la résolution provient — on ne saurait s'en étonner — des disjonctions. Ces dernières forment le système complémentaire $\overline{H}$.

Les variables sélectionnées sont les y'_j et les branchements sont binaires.

9.1.2 Second modèle de relaxation

Cette seconde méthode est plus économe en nombre de clauses et de variables introduites. Elle décompose symétriquement chaque clause C en deux inégalités, dont l'une est de Horn : $x_1 + x_2 + ... + x_k \leq z + (k–1)$ et l'autre pas : $z \leq y_1 + y_2 + ... + y_l$.

1. La reconnaissance d'un système Horn-renommable se ramène à un problème 2SAT dont la complexité est quadratique.

Les nouvelles variables, z, servent de passerelle entre deux systèmes monotones, H et $\overline{H}$, de même cardinalité.

Après le choix d'une variable, z, le problème est séparé[1] en autant de sous-problèmes qu'il y a de variables dans la clause : $z \leq y_1 + y_2 + ... + y_l$.

Les résultats expérimentaux obtenus par Gallo & Urbani indiquent une supériorité du second modèle de relaxation sur l'algorithme de Davis & Putnam (qui reste plus performant que la première méthode évoquée). Hooker confirme ces performances mais signale également qu'elles peuvent se dégrader rapidement sur des cas réputés difficiles, avec une croissance explosive de l'arbre de recherche. Il eût été étonnant, avec SAT, que l'on gagne à tous les coups !

9.2 Résolution locale en programmation par contraintes

Comme le relate Simonis (1989), les techniques de propagation et de réduction d'intervalles sont souvent plus appropriées que des techniques complètes comme l'unification booléenne pour la résolution de problèmes appliqués. Assistées d'une heuristique d'énumération en {0,1}, elles offrent une alternative efficace pour propager l'information factuelle.

Les solveurs de domaines finis définissent ainsi des fonctions de base *et*, *ou*, *non*, avec lesquelles le programmeur peut construire des expressions booléennes complexes, sans nécessairement passer par une forme normale.

La table suivante résume les propagations effectuées par les «démons» (ou opérateurs de réduction) associés aux relations ternaires `and` et `or` de CHIP[2]:

$x \wedge y = z$			$x \vee y = z$		
$X = \{0\}$	$\Rightarrow$	$Z = \{0\}$	$X = \{1\}$	$\Rightarrow$	$Z = \{1\}$
$Y = \{0\}$	$\Rightarrow$	$Z = \{0\}$	$Y = \{1\}$	$\Rightarrow$	$Z = \{1\}$
$X = \{1\}$	$\Rightarrow$	$Z = Y$ [a]	$X = \{0\}$	$\Rightarrow$	$Z = Y$
$Y = \{1\}$	$\Rightarrow$	$Z = X$	$Y = \{0\}$	$\Rightarrow$	$Z = X$
$Z = \{1\}$	$\Rightarrow$	$X = Y = \{1\}$	$Z = \{0\}$	$\Rightarrow$	$X = Y = \{0\}$

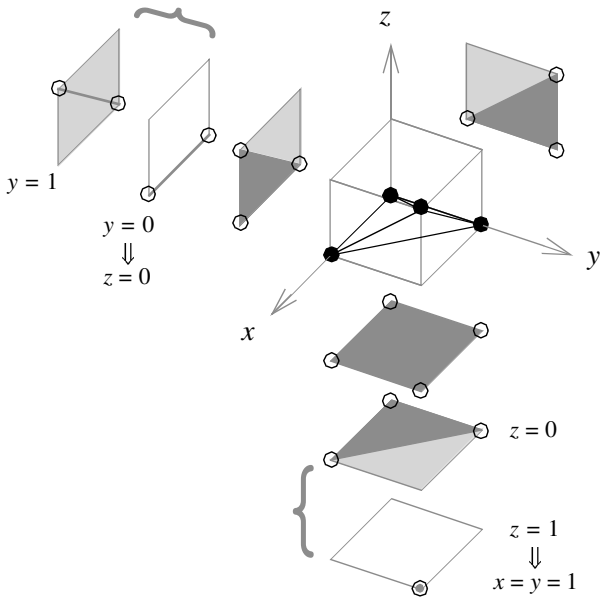
a. Formellement, on déduit l'égalité : $z = y$.

L'implantation de ces réductions est souvent spécifique (Codognet et Diaz 1996), mais une traduction dans l'arithmétique linéaire permet de les simuler.

Les systèmes élémentaires qui définissent ces relations de base sont tous en effet renommables en systèmes de Horn et sont donc, localement, bien résolus. En outre, les projections calculées sont *exactes*, comme l'illustre le schéma de la relation *et*.
L'opérateur de réduction associé à chacun de ces systèmes peut donc faire office d'opérateur de réduction pour chacune de ces contraintes.

1. Par un branchement multiple : $S \cup \{y_1\}$; $S \cup \{\neg y_1, y_2\}$; $S \cup \{\neg y_1, \neg y_2, y_3\}$; etc.
2. En Prolog IV, ces relations sont appelées `band` et `bor`.

Relation	Définition	Traduction	Renommage
$y = \neg x$	$y \Leftrightarrow \neg x$	$y \leq 1 - x$ $1 - x \leq y$ $0 \leq x \leq 1$ $0 \leq y \leq 1$	$y \leq x'$ $x' \leq y$ $0 \leq x' \leq 1$ $0 \leq y \leq 1$
$x \wedge y = z$	$x \wedge y \Leftrightarrow z$	$x + y \leq z + 1$ $z \leq x$ $z \leq y$ $0 \leq x \leq 1$ $0 \leq y \leq 1$ $0 \leq z \leq 1$	
$x \vee y = z$	$x \vee y \Leftrightarrow z$	$z \leq x + y$ $x \leq z$ $y \leq z$ $0 \leq x \leq 1$ $0 \leq y \leq 1$ $0 \leq z \leq 1$	$x' + y' \leq z' + 1$ $z' \leq x'$ $z' \leq y'$ $0 \leq x' \leq 1$ $0 \leq y' \leq 1$ $0 \leq z' \leq 1$



$x \wedge y = z$: solutions et projections

CHAPITRE VI

Cheminements et ordonnancements

Optimisation combinatoire sur les graphes

> «De même que l'Arithmétique étudie les nombres entiers (avec les opérations classiques), que l'Algèbre étudie les opérations en général, que l'Analyse étudie les fonctions, que la Géométrie étudie les formes rigides et la Topologie celles qui ne le sont pas, la Combinatoire étudie, elle, les configurations.»
>
> C. BERGE

Il est bien connu qu'un problème de plus court chemin peut être abordé au travers d'une formulation linéaire en nombres entiers. La résolution est facilitée par la structure du modèle qui assimile la recherche à celle d'un flot de coût minimum sur un réseau. La solution optimale du problème continu donne alors la plus courte distance sur ce graphe, entre une origine et une destination.

Il est moins connu que l'on peut arriver au même résultat en effectuant, à la place d'une relaxation linéaire, une «relaxation par les intervalles», c'est-à-dire, en pratique, en substituant aux pas d'optimisation d'un simplexe les itérations d'un calcul de réduction de domaines.

En basculant du modèle standard (formulation en courants) à son dual (formulation en tensions), nous démontrons que les domaines réduits des variables duales (potentiels au nœuds) renferment dans leurs bornes entières l'information sur les plus courtes distances recherchées. La preuve découle d'une application aux inégalités à deux variables du théorème sur les systèmes 1-monotones. Ce résultat de complétude joue ici pour le dual, un rôle symétrique à celui du théorème d'intégrité pour le primal.

Avec un solveur linéaire comme avec un solveur d'intervalles, les solutions de ce problème d'optimisation combinatoire s'obtiennent sans énumération (et sans avoir à écrire le moindre algorithme).

L'étude des cheminements optimaux sur des graphes munis de «longueurs négatives» soulève des difficultés plus importantes, tant au niveau l'expression (linéaire) du problème que de sa résolution (avec les intervalles).

Dans cette optique, nous procédons à un réexamen du problème, sous son acception la plus générale — qui peut paraître spécieuse du point de vue du primal, mais correspond à des cas très concrets du côté du dual[1].

En notant $\mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}$ le système d'inégalités dual et $\boldsymbol{cx}$ la différence $x_d - x_o$ à maximiser entre l'origine o et la destination d, nous établissons les égalités suivantes :

Problème combinatoire	*Relaxation linéaire*	*Modèle primal*
$\min\{\boldsymbol{by} \mid {}^t\mathbf{A}\boldsymbol{y} = \boldsymbol{c}, \boldsymbol{y} \in \mathbb{N}^m\}$	$= \min\{\boldsymbol{by} \mid {}^t\mathbf{A}\boldsymbol{y} = \boldsymbol{c}, \boldsymbol{y} \geq 0\}$	
	$= \max\{\boldsymbol{cx} \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}\}$	$= \mathrm{Proj}_d \max\{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}, x_o = 0\}$
Modèle dual	*Problème dual*	*Problème résolu avec les intervalles*

Prenant comme référence l'interprétation en tensions lorsque les longueurs b_i sont de signe quelconque, nous retrouvons, par l'analyse des inégalités, certains des algorithmes les plus connus sur les graphes. En l'occurrence, nous faisons dans ce chapitre le lien entre :

- la réduction d'intervalles et l'algorithme de Ford
  (en signalant, en cas d'insatisfaisabilité, leurs problèmes respectifs de convergence lente et de non terminaison)
- l'élimination de Fourier et les algorithmes matriciels (complets et rapides) de calcul des plus courtes distances.

Nous terminons par une généralisation aux problèmes d'ordonnancements simples dont les contraintes de précédences sont des contraintes de tensions déguisées.

1 Modélisation par les graphes

Dans les structures de données informatiques, dans les représentations schématiques, les graphes sont omniprésents. On les utilise pour décrire des connexions physiques, des liaisons chimiques, des dépendances temporelles, etc. Le plan du métro en donne un parfait exemple : un graphe figure la situation relative de points et de lignes, sans considération de leur grandeur — comme ce serait le cas en géométrie.

Formellement, la topologie combinatoire définit les graphes comme des polyèdres complexes à une dimension (les polyèdres simples étant dans ce cas des segments). Un graphe est un assemblage d'*arêtes*, disjointes ou en contact par l'une ou l'autre de leurs extrémités. Ces extrémités, de dimension 0, forment les *sommets* du graphe.

Les arêtes (déformables) deviennent des *arcs* (fléchés) lorsqu'elles sont orientées. Elles définissent une relation d'adjacence entre les sommets. Le graphe est cn l'occurrence celui de cette relation binaire.

1. Nous en verrons un exemple au chapitre suivant.

Il est loisible d'enrichir le formalisme en annotant chaque arc par une couleur, une longueur ou un prix. On parle alors de graphe valué, voire de multi-graphe si plusieurs arcs relient, dans le même sens, deux mêmes sommets.

Définition Un *graphe simple G* est défini par un couple (N, A) dans lequel N désigne un ensemble de sommets (ou *nœuds* du graphe) et A un sous-ensemble de $N \times N$ correspondant aux arcs.

Dans un graphe simple, on a donc, pour tout couple origine-destination, au plus un arc $j \to k$ reliant le sommet j au sommet k.

Définition Un graphe est *sans boucle* s'il est sans arc $j \to j$.

Dans un graphe sans boucle, la relation d'adjacence est irréflexive sur N.

On utilise généralement la représentation graphique pour des structures finies (où les ensembles A et N sont donnés en extension).

Définition Un graphe simple est *fini* quand l'ensemble des ses sommets, N, est fini.

On désignera alors par n le cardinal de N, et m celui de A ($m \leq n^2$).

Définition Un *graphe valué* est un triplet $G = (N, A, v)$ constitué d'un graphe (N, A) et d'une fonction de valuation v définie sur A et à valeurs entières dans $\mathbb{Z}$.

Lorsque $|A| = m$, la fonction v équivaut à la donnée d'un vecteur de $\mathbb{Z}^m$ que nous noterons $\boldsymbol{b}$. Pour tout arc i, $1 \leq i \leq m$, on aura : $v(i) = b_i$.

Dans toute la suite, nous raisonnerons sur la base d'un graphe orienté simple[1], fini et sans boucle[2].

2 Chemins et plus courts chemins sur un graphe

La recherche d'un plus court chemin, d'un sommet à un autre, est un sujet classique de la théorie des graphes. Ce parcours, qui emprunte les arcs et pose en chaque sommet le choix de la direction à prendre pour atteindre le but fixé, sert également de paradigme de résolution de problèmes en Intelligence artificielle (Doran & Michie). Or on sait, depuis le Labyrinthe de Dédale, qu'une telle recherche peut se transformer en piège quasi inextricable.

Pour éviter de se fourvoyer dans sa combinatoire, on résoudra au préalable une question plus générale — mais curieusement plus simple — qui consiste à calculer, pour une valuation donnée, les longueurs des plus courts chemins entre un sommet et tous les autres.

1. Dans le cas d'un multi-graphe, il suffira de retenir un sous-graphe constitué d'arcs de valuation minimale.
2. Plus précisément, sans boucle de valuation négative. Les boucles de valuation positive peuvent être considérées comme des arcs redondants (pour le calcul des longueurs, une boucle de valuation nulle équivaut à une absence de boucle).

2.1 Rappels terminologiques

Définition Un *chemin* est une séquence finie d'arcs $o \to s_1, s_1 \to s_2, \ldots, s_{k-1} \to d$ $(k \geq 0)$ reliant, sur un graphe orienté, un sommet origine o à un sommet destination d.

Il n'existe de chemin entre o et d que si le couple (o, d) appartient à la fermeture réflexo-transitive de la relation $\to$ définie par les arcs du graphe.

Lorsque $k = 0$, on parlera de chemin *vide*.

Définition On dit qu'un sommet s est une *racine* d'un graphe s'il existe un chemin entre s et tous les autres sommets du graphe.

L'existence d'une racine suffit à prouver la connexité d'un graphe.

Définition Un *circuit* est un chemin (non vide) dont l'origine et la destination coïncident.

L'existence de circuits dans un graphe donne naissance à une infinité de chemins distincts.

Exemple Si $a \to b$ et $b \to a$ sont deux arcs du graphe, on peut énumérer les chemins :

$a \to b$
$a \to b \to a$
$a \to b \to a \to b$
$a \to b \to a \to b \to a$
...

Définition Un chemin est *élémentaire* s'il ne comprend pas de circuit, autrement dit, s'il ne recroise aucun de ses propres sommets.

Un graphe fini ne présente qu'un nombre fini de chemins élémentaires.

Définition La *longueur* l d'un chemin c est la somme des valeurs attribuées aux arcs qui le constituent.

La longueur du chemin $o \to s_1, s_1 \to s_2, \ldots, s_{k-1} \to d$ vaut $l = \sum_{i=0}^{k-1} v(s_j \to s_{j+1})$.

Classiquement, ces valeurs représentent des distances et sont donc positives : $\boldsymbol{b} > 0$. Mais on peut aussi rencontrer des valeurs négatives.

Définition Un circuit est *absorbant* s'il est de longueur strictement négative.

La présence d'un circuit absorbant dans un graphe engendre des chemins de longueur négative dont on peut faire croître à volonté la valeur absolue.

Exemple Si $v(a \to b) = 0$ et $v(b \to a) = -1$, le circuit $a \to b \to a$ est absorbant (de longueur -1). Les longueurs des chemins obtenus en suivant ce circuit sont : 0, –1, –1, –2, –2, –3, ...

2.2 Problèmes de plus courts chemins

Soient o et d deux sommets distingués d'un graphe $G = (N, A, b)$. Notons C (resp. C_E) l'ensemble des chemins (resp. chemins élémentaires) de o à d.
On s'intéresse *a priori* aux problèmes d'optimisation suivants :

$$\min\{l(c) \mid c \in C\} \quad \text{(I)}$$

$$\min\{l(c) \mid c \in C_E\} \quad \text{(II)}$$

Pour le problème (I), trois cas peuvent se présenter :

1. il n'existe pas de chemin entre o et d
   $C = \emptyset$: le sommet d n'est pas accessible à partir de o
2. il existe des chemins entre o et d mais pas de plus court chemin
   le problème est non borné : un chemin de o à d rencontre un circuit absorbant
3. il existe un ou plusieurs chemins de longueur minimale l^*
   $l^* = \min\{l(c) \mid c \in C\}$: l^* est la plus courte distance de o à d.

Si l'on s'en tient à des valuations strictement positives (ce qui est le cas le plus courant), 2 ne peut se produire. L'alternative est entre 1 (problème insoluble) et 3 (existence d'un plus court chemin — lequel est alors nécessairement un chemin élémentaire). Le problème (I), et par lui, le problème (II) se résolvent alors de façon polynômiale, avec, entre autres, l'algorithme de Dijkstra (1959) en $O(n^2)$[1].

Plus généralement, on a identité des réponses pour les problèmes (I) et (II) dans les cas 1 et 3, car, quel que soient les signes des longueurs :

- $C = \emptyset$ si et seulement si $C_E = \emptyset$;
- si $\min\{l(c) \mid c \in C\}$ existe, alors $\min\{l(c) \mid c \in C\} = \min\{l(c) \mid c \in C_E\}$.

En revanche, si l'on autorise des valuations négatives, on a, dans le cas 2 : $\inf\{l(c) \mid c \in C\} = -\infty$, alors que $\min\{l(c) \mid c \in C_E\}$ existe et est fini.
Mais on ne sait pas calculer ce dernier plus efficacement qu'en énumérant tous les chemins élémentaires du graphe...

Face à ce risque d'explosion combinatoire, on conçoit qu'on préfère revenir à la définition (I) d'un plus court chemin, si l'on a une garantie d'efficacité à la clé. A cet égard, (I) peut être vu comme une relaxation de (II).
En pratique, on ira jusqu'à relâcher (I) en présence de circuits absorbants dans le graphe. Que $\min\{l(c) \mid c \in C\}$ existe ou non, on s'en tiendra à la réponse «les longueurs des chemins sur le graphe n'ont pas de borne inférieure finie».

En définitive, ce que l'on entend dans la littérature par la «longueur du plus court chemin» est la réponse à un troisième problème (III), justiciable d'une modélisation linéaire (et par conséquent d'une résolution efficace). C'est le modèle que nous allons maintenant formuler. En présence de «longueurs négatives», son interprétation la plus limpide viendra du dual.

1. $O(m + n\log n)$ pour un graphe peu dense ($m << n^2$).

3 Courants minimaux et plus courtes distances

Tout chemin de o à d est constitué d'une suite finie $(a_i)_{i \in I}$ d'éléments de A.
Réciproquement, pour qu'une suite d'éléments de A représente un chemin joignant o à d, elle doit vérifier un certain nombre de conditions.
A peu de choses près, nous allons parvenir à les exprimer en nous servant d'un modèle linéaire dont nous préciserons à ce propos les limites.

Il suffira alors minimiser sous ces contraintes la fonction objectif $\sum_{i \in I} b_i$ pour obtenir la longueur du plus court chemin de o à d.

3.1 Tableaux d'incidence

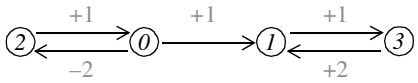
La description de la structure d'un graphe fini $G = (N, A)$ peut être donnée par un tableau à double entrée **T** dont l'origine remonte à Poincaré (1900).

Définition On forme la *matrice d'incidence sommets-arcs* d'un graphe fini sans boucle, (N, A), en portant en ligne les sommets, N, en colonne les arcs, A, et en inscrivant à l'intérieur du tableau :

- -1 à l'intersection d'une ligne j et d'une colonne $j \to k$
- $+1$ à l'intersection d'une ligne j et d'une colonne $k \to j$
- 0 partout ailleurs.

Sa transposée **A** est appelée *matrice d'incidence arcs-sommets.*

Exemple La matrice d'incidence du graphe suivant est un tableau 4×5 :



T	0 → 1	0 → 2	2 → 0	1 → 3	3 → 1
0	–1	–1	+1	0	0
1	+1	0	0	–1	+1
2	0	+1	–1	0	0
3	0	0	0	+1	–1

Les valeurs associées aux 5 arcs forment les composantes d'un vecteur :

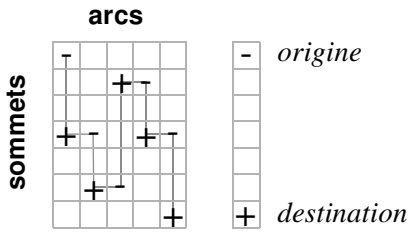
b	+1	-2	+1	+1	+2

Les graphes partiels des chemins peuvent être décrits par des tableaux similaires.

3.2 Formulation en nombres entiers

Prenons un chemin c sur le graphe. En remplaçant chaque arc par sa colonne tirée du tableau d'incidence, nous obtenons un nouveau tableau dont les éléments non nuls, lus verticalement puis horizontalement, forment une suite alternée de moins uns et plus uns.

Additionnons les colonnes de c : les éléments intermédiaires s'annulent.



Le résultat est une colonne creuse, analogue à un arc $o \to d$:

- avec, si o est différent de d :
  -1 pour le sommet origine, $+1$ pour le sommet destination
- identiquement nulle sinon.

On la notera (o, d) pour ne pas la confondre avec un arc du graphe.

Théorème VI.1 *Tout chemin de o à d est solution du système d'équations diophantiennes* $\mathbf{Ty} = \boldsymbol{c}$ *où* $\mathbf{T}$ *est la matrice d'incidence du graphe,* $\boldsymbol{c}$ *la colonne correspondant au couple (o, d) et* $\boldsymbol{y}$ *est un vecteur de* $\mathbb{N}^{|A|}$ *spécifiant une sélection de colonnes de* $\mathbf{T}$ *(avec répétitions éventuelles). Réciproquement, toute solution entière* $\boldsymbol{y} \geq 0$ *du système* $\mathbf{Ty} = \boldsymbol{c}$ *contient un chemin joignant o à d.*

Démonstration

La première partie de la proposition vient d'être démontrée. Reste la réciproque.

Soit $\boldsymbol{y}$ une solution entière positive du système $\mathbf{Ty} = \boldsymbol{c}$. L'équation relative au sommet origine s'écrit comme une différence de deux quantités positives :

$\sum_i y_{j \to o} - \sum_i y_{o \to j} = -1$. Le total étant négatif, on en déduit qu'il existe un sommet k pour lequel $y_{o \to k} \geq 1$.

Retranchons de chaque membre du système $\mathbf{Ty} = \boldsymbol{c}$ la colonne constante correspondant à l'arc $o \to k$ (que l'on mémorisera). Le membre gauche se réécrit $\mathbf{Ty'}$ où $\boldsymbol{y'}$ est le vecteur positif obtenu à partir de $\boldsymbol{y}$ en décrémentant $y_{o \to k}$ de 1. Le membre droit devient la colonne $\boldsymbol{c'}$ du couple (k, d). Autrement dit, l'origine du problème se déplace de o en k.

L'opération peut être répétée sur $\mathbf{Ty'} = \boldsymbol{c'}$ tant que l'origine diffère de la destination d. Comme chaque déplacement fait décroître la norme de la solution courante $\boldsymbol{y'} \geq 0$ (en ajoutant un arc au chemin issu de o), d est forcément atteint en moins de $\| \boldsymbol{y} \|_1$ étapes.

On a donc : $\boldsymbol{y} = \boldsymbol{z} + \boldsymbol{z'}$ avec $\boldsymbol{z} \geq 0$, $\boldsymbol{z'} \geq 0$, $\mathbf{Tz} = \boldsymbol{c}$, $\mathbf{Tz'} = \mathbf{0}$. Les arcs sélectionnés par le vecteur entier $\boldsymbol{z}$ forment — pour une permutation du moins — un chemin de o à d. Les arcs résiduels forment éventuellement un ensemble de circuits[1], connexes ou non au chemin solution. ■

1. Il suffit en effet de retirer un arc $k \to j$ à la solution $\boldsymbol{z'}$ du système homogène $\mathbf{Tz'} = \mathbf{0}$ pour que le problème revienne à joindre k à partir de j.

Ce théorème énonce une condition nécessaire et suffisante de satisfaisabilité : *il existe un chemin (élémentaire) de o à d si et seulement si le système* $\mathbf{T}\boldsymbol{y} = \boldsymbol{c}, \boldsymbol{y} \geq 0$ *admet une solution entière*.

Les solutions $\boldsymbol{y}$ sont définies à une permutation près (ce sont des suites d'arcs «désordonnées»). Mais on a seulement circonscrit le problème, car toutes ne correspondent pas à des chemins de C.

Exemple (suite) Reprenons la matrice d'incidence de l'exemple précédent et appliquons-la au problème du calcul d'un chemin du sommet 1 au sommet 3 sur ce graphe. Le système à résoudre comporte 4 équations pour 5 inconnues entières positives ($y_{01}, y_{02}, y_{20}, y_{13}, y_{31}$) :

$$\begin{aligned} -y_{01} - y_{02} + y_{20} &= 0 && \textit{sommet 0} \\ y_{01} - y_{13} + y_{31} &= -1 && \textit{sommet 1} \\ y_{02} - y_{20} &= 0 && \textit{sommet 2} \\ y_{13} - y_{31} &= +1 && \textit{sommet 3} \end{aligned}$$

(0,0,0,1,0) est évidemment solution, comme toute combinaison de ce vecteur avec une solution (0,*n*,*n*,*m*,*m*) du système homogène associé ($n, m \in \mathbb{N}$). Le système $\mathbf{T}\boldsymbol{y} = \boldsymbol{c}$ a ici une infinité de solutions. Certaines, comme (0,0,0,2,1), sont des chemins de 1 à 3, d'autres, comme (0,5,5,1,0), n'en sont pas mais en comprennent un.
Si l'on remplace le membre droit par (+1,–1, 0, 0), le système devient clairement insoluble : il n'y a pas de chemin allant de 1 à 0.

Venons-en maintenant au problème d'optimisation annoncé :

$$\min\{\boldsymbol{by} \mid \mathbf{T}\boldsymbol{y} = \boldsymbol{c}, \boldsymbol{y} \in \mathbb{N}^{|A|}\} \qquad \text{(III)}$$

Lorsque les valeurs $b_{jk} = v(j \to k)$ associées aux arcs sont des entiers positifs et qu'il existe un chemin de *o* à *d*, l'ensemble des longueurs $\boldsymbol{by}$ est non vide et minoré : il admet donc un minimum entier $l \geq 0$. La minimisation éliminant alors les circuits[1], cette longueur est celle du plus court chemin.
Mais ce n'est plus le cas lorsque les entiers b_{jk} sont de signe quelconque.

Exemple (fin) Le graphe précédent offre un contre-exemple dans le cas de valeurs négatives (mentionnées sur les arcs). En l'occurrence, il existe un plus court chemin, de longueur 1, du sommet 1 au sommet 3 : l'arc $1 \to 3$. Mais la présence du circuit absorbant $0 \to 2 \to 0$ parasite la minimisation, avec des «solutions» (0,*n*,*n*,1,0) dont la longueur, $-n+1$, tend vers $-\infty$. De fait, sur l'ensemble des chemins du graphe, il n'y a pas de plus court chemin.

L'absence de borne inférieure finie donne un moyen de *détecter la présence de circuits absorbants* dans le graphe. Mais en leur présence, la formulation du problème ne permet pas de trouver un plus court chemin s'il en existe.

1. Du moins, ceux de longueur strictement positive (il peut subsister des circuits de longueur nulle).

3.3 Unimodularité et relaxation sur les réels

L'intérêt porté à la structure du système de contraintes se justifie par les propriétés qu'elle confère.

Définition Une matrice $\mathbf{T}$ en $\{0, \pm 1\}$ est *totalement unimodulaire* si le déterminant de toute matrice carrée extraite de $\mathbf{T}$ vaut 0 ou ± 1.

Pour les matrices ayant exactement un +1 et un –1 dans chaque colonne, cette propriété se vérifie aisément par récurrence :

Propriété *Les matrices d'incidence sommets-arcs sont totalement unimodulaires.*

Considérons en effet une sous-matrice carrée d'une matrice d'incidence $\mathbf{T}$. Si chacune de ses colonnes contient à la fois +1 et –1, alors son déterminant est nul ; sinon, le développement de ce déterminant suivant une colonne possédant au plus un coefficient non nul donne 0 ou ± 1. ■

Les systèmes d'inégalités à coefficients entiers dont la matrice est totalement unimodulaire possèdent la propriété remarquable de définir explicitement l'enveloppe convexe de l'ensemble de leurs solutions entières. Le *théorème d'intégrité* suivant montre que ces polyèdres sont à sommets entiers.

Théorème VI.2 *Soit* $\mathbf{T}$ *une matrice* $n \times m$ *totalement unimodulaire et* $\boldsymbol{c}$ *un vecteur de* $\mathbb{Z}^m$. *Alors les sommets du polyèdre* $\{\boldsymbol{y} \in \mathbb{R}^m \mid \mathbf{T}\boldsymbol{y} = \boldsymbol{c}, \boldsymbol{y} \geq 0\}$ *sont à coordonnées entières.*

Démonstration

On peut toujours supposer que la matrice $\mathbf{T}$ a pour rang le nombre n de ses lignes. Si rg $\mathbf{T} = r < n$, une ou plusieurs lignes s'expriment comme combinaisons linéaires des autres. Suivant la valeur des coefficients du membre droit $\boldsymbol{c}$, les contraintes correspondantes sont soit redondantes (auquel cas, on peut en supprimer les lignes), soit incompatibles avec les autres (auquel cas, le système n'a pas de solution).

Soit $\boldsymbol{s}$ un sommet du polyèdre $\{\boldsymbol{y} \mid \mathbf{T}\boldsymbol{y} = \boldsymbol{c}, \boldsymbol{y} \geq 0\}$. Si $\boldsymbol{s}$ existe, $\boldsymbol{s}$ est l'unique intersection d'une famille finie d'hyperplans, formée des égalités $\mathbf{T}\boldsymbol{y} = \boldsymbol{c}$, auxquelles se rajoutent $m - r$ égalités $y_j = 0$ issues des hyperplans qui bornent le cône positif de $\mathbb{R}^m$ (et dont l'ajout revient à supprimer autant de colonnes de $\mathbf{T}$). Ainsi $\boldsymbol{s}$ est nécessairement solution d'un système $\mathbf{T'}\boldsymbol{y'} = \boldsymbol{c'}$ de r équations à r inconnues, où $\mathbf{T'}$ est une sous-matrice carrée inversible extraite (ou base) de $\mathbf{T}$. Donc $\boldsymbol{s} = (\mathbf{T'})^{-1}\boldsymbol{c'}$ avec $\det(\mathbf{T'}) = \pm 1$. Comme, par les formules de Cramer, $(\mathbf{T'})^{-1}$ reste entier, il s'ensuit que si $\boldsymbol{c}$ est à coordonnées entières, $\boldsymbol{s}$ l'est également. ■

Un théorème plus général de Hoffman & Kruskal (1956) établit que, pour une matrice entière $\mathbf{T}$, la condition «$\mathbf{T}$ est totalement unimodulaire» est nécessaire et suffisante pour que le polyèdre $\{\boldsymbol{y} \mid \mathbf{T}\boldsymbol{y} \leq \boldsymbol{c}, \boldsymbol{y} \geq 0\}$ soit à sommets entiers, et ce pour tout vecteur entier $\boldsymbol{c}$.

Proposition *Si le graphe est simplement connexe, sa matrice d'incidence est de rang $|N|-1$.*

Si l'on fait la somme des lignes du système $\mathbf{T}\boldsymbol{y} = \boldsymbol{c}$, on obtient $0 = 0$: le système est au plus de rang $|N|-1$. Si le graphe est simplement connexe, quel que soit le vecteur $\boldsymbol{c}$ correspondant à un couple (o, d) tel que $o \neq d$, le système $\mathbf{T}\boldsymbol{y}_1 - \mathbf{T}\boldsymbol{y}_2 = \boldsymbol{c}$ a une solution (à tout arc $j \to k$ on a rajouté son symétrique $k \to j$). Eliminons la ligne de o et faisons varier d : $(\mathbf{T}, -\mathbf{T})$, dont le rang est égal à $\mathbf{T}$, engendre une base de $\mathbb{R}^{|N|-1}$. Donc $\mathbf{T}$ est de rang $|N|-1$. ■

Corollaire *La totale unimodularité de $\boldsymbol{T}$ nous permet de tirer deux conséquences :*

1. *le problème de satisfaisabilité sur les entiers est équivalent à un problème de satisfaisabilité sur les réels*
2. *le problème d'optimisation sur les entiers est — pour des valuations entières — équivalent à un problème d'optimisation sur les réels.*

On peut donc relâcher les contraintes d'intégrité de ces deux problèmes linéaires pour confier leur résolution, par exemple, à un algorithme du Simplexe spécialisé.

Appliqué à la modélisation (III), cela donne :

Problème de décision	Problème d'optimisation
existence d'un chemin de o à d	*calcul de la plus courte distance*
$\mathrm{Q} = \{\boldsymbol{y} \in \mathbb{Z}^m \mid \mathbf{T}\boldsymbol{y} = \boldsymbol{c}, \boldsymbol{y} \geq 0\} \neq \varnothing$ $\Leftrightarrow$ $\mathrm{R} = \{\boldsymbol{y} \in \mathbb{R}^m \mid \mathbf{T}\boldsymbol{y} = \boldsymbol{c}, \boldsymbol{y} \geq 0\} \neq \varnothing$	$\min\{\boldsymbol{b}\boldsymbol{y} \mid \boldsymbol{y} \in \mathrm{Q}\}$ $=$ $\min\{\boldsymbol{b}\boldsymbol{y} \mid \boldsymbol{y} \in \mathrm{R}\}$

Le théorème d'intégrité nous garantit que s'il existe une solution (optimale), alors il existe une solution (optimale) entière : l'élargissement aux réels n'est pas seulement une approximation correcte, elle est aussi complète.

3.4 Interprétation par les courants

Le théorème d'intégrité nous fait passer d'un problème combinatoire discret à un problème continu : celui d'un courant dans le graphe. En effet, lorsque le membre droit est nul, les équations du système $\mathbf{T}\boldsymbol{y} = \boldsymbol{c}$ traduisent la conservation d'un flux $\boldsymbol{y} \in \mathbb{R}^m$ transitant par un nœud j :

$$\sum_k y_{k \to j} - \sum_k y_{j \to k} = 0$$

Tout ce qui entre en ce nœud en ressort. C'est la première loi de Kirchhoff dans les réseaux.

Quand le bilan n'est pas nul, le membre droit c_j quantifie l'apport ou le retrait de matière ou de fluide effectué en ce nœud.

Par analogie, on peut dire que le problème du plus court chemin est résolu en injectant dans le réseau une unité de flux en o et en tenant de la recueillir en d.

Les solutions qui créent *ex nihilo* des quantités de flux infinies dans des circuits n'ont pas de sens physique, ce qui justifie qu'elles soient rejetées au niveau de la minimisation.

Remarque La reconstitution des chemins de longueur minimale reste un problème combinatoire qui s'effectue de manière non-déterministe, à partir des solutions du système d'équations diophantiennes : $\mathbf{T}\boldsymbol{y} = \boldsymbol{c}$, $\boldsymbol{by} = l^*$ ($\boldsymbol{y} \in \mathbb{N}^m$).

4 Tensions maximales et plus courtes distances

4.1 Un calcul analogique

Lorsque le graphe est symétrique et que ses valuations sont positives, une manière amusante de résoudre un problème de plus court chemin est d'en réaliser un modèle «physique» en corde non élastique. On noue des segments de longueur proportionnelle à celle des arêtes. Puis on tend au maximum le réseau ainsi formé, en le tirant d'un côté par le nœud origine et de l'autre par le nœud destination. Si l'origine et la destination sont reliées, l'opération met en tension une corde qui révèle le plus court chemin recherché.

Cet artifice, signalé par Minty (1957), traduit plus profondément une dualité entre courants et tensions. A un chemin minimal sur un graphe correspond une tension maximale, et vice versa.

4.2 Formulation en tensions

Définition Une *potentiel* sur un graphe est une fonction de N dans $\mathbb{R}$.

Soit $\mathbf{A} = {}^{t}\mathbf{T}$. $\mathbf{A}$ est un matrice entière $m \times n$.

Le programme linéaire dans lequel $\boldsymbol{y} \in \mathbb{R}^{|A|}$ désigne un courant :

$$\min\{\boldsymbol{by} \mid \mathbf{T}\boldsymbol{y} = \boldsymbol{c}, \boldsymbol{y} \geq 0\} \qquad \text{(III.1)}$$

est le dual [1] du programme linéaire dans lequel $\boldsymbol{x} \in \mathbb{R}^{|N|}$ désigne un potentiel :

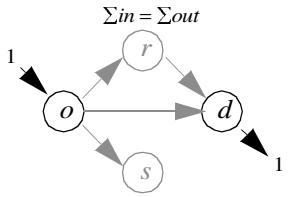
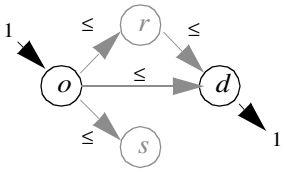
$$\max\{\boldsymbol{cx} \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}\} \qquad \text{(III.2)}$$

$\boldsymbol{cx} = x_d - x_o$ représente la tension (ou différence de potentiel) entre o et d.

De même que les lignes du primal (III.1) sont linéairement dépendantes, les colonnes $\boldsymbol{x}$ du dual (III.2) sont définies à une constante près.
Supprimer l'équation au noeud o dans le modèle primal revient à imposer un potentiel 0 en o et à calculer en chaque nœud la différence de potentiel avec o.

1. $l^* = \max\{\boldsymbol{cx} \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}\} = -\min\{-\boldsymbol{cx} \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}\} = -\min\{\boldsymbol{c}(-\boldsymbol{x}) \mid \mathbf{A}(-\boldsymbol{x}) \geq -\boldsymbol{b}\} = -z^*$
$= -\max\{-\boldsymbol{by} \mid \mathbf{T}\boldsymbol{y} = \boldsymbol{c}, \boldsymbol{y} \geq 0\} = \min\{\boldsymbol{by} \mid \mathbf{T}\boldsymbol{y} = \boldsymbol{c}, \boldsymbol{y} \geq 0\}$

Exemple Comparaison des formulations primale et duale sur un graphe :

Courants		**Tensions**
une variable par arc		*une variable par nœud*
une contrainte par nœud		*une contrainte par arc*
		
$\{y_{rd} + y_{od} = 1,\ y_{or} = y_{rd},\ y_{os} = 0\}$		$\{x_r \le b_{or},\ x_d \le b_{od},\ x_s \le b_{os},\ x_d - x_r \le b_{rd}\}$
$b_{od}.y_{od} + b_{or}y_{or} + b_{os}.y_{os} + b_{rd}.y_{rd}$	$\ge$	x_d

Le lien entre les solutions du primal et celles du dual s'effectue par les coûts. Si $\boldsymbol{x}$ est solution de $\mathbf{A}\boldsymbol{x} \le \boldsymbol{b}$ et $\boldsymbol{y}$ solution de $\mathbf{T}\boldsymbol{y} = \boldsymbol{c}$, $\boldsymbol{y} \ge 0$, alors $\boldsymbol{cx} \le \boldsymbol{by}$:

1. s'il n'y a *pas de tension compatible*, $\sup\{x_d \mid \mathbf{A}\boldsymbol{x} \le \boldsymbol{b},\ x_o = 0\} = -\infty$, le problème primal est non borné : nous allons alors montrer qu'*il existe des circuits absorbants* ;
2. si le problème dual est non borné, $\sup\{x_d \mid \mathbf{A}\boldsymbol{x} \le \boldsymbol{b},\ x_o = 0\} = +\infty$, il n'y a *pas de courant compatible*, donc *pas de chemin de o* à *d* ;
3. à l'optimum, $\max\{x_d \mid \mathbf{A}\boldsymbol{x} \le \boldsymbol{b},\ x_o = 0\} = \min\{\boldsymbol{by} \mid \mathbf{T}\boldsymbol{y} = \boldsymbol{c},\ \boldsymbol{y} \ge 0\}$, la *tension maximale* en *d* donne la *longueur d'un chemin minimal o - d*.

4.3 Conditions de satisfaisabilité

Notons S le système de n variables et $m+2$ inégalités : $\mathbf{A}\boldsymbol{x} \le \boldsymbol{b}$, $x_0 = 0$.

Propriété S *est satisfaisable si et seulement si le système* $\mathbf{A}\boldsymbol{x} \le \boldsymbol{b}$ *est satisfaisable.*

Les potentiels sont en effet définis à une constante près. Si $\boldsymbol{x}$ est solution de $\mathbf{A}\boldsymbol{x} \le \boldsymbol{b}$, alors pour toute constante k, sa translation $\boldsymbol{x} + k\boldsymbol{1}$ est solution. Si $\mathbf{A}\boldsymbol{x} \le \boldsymbol{b}$ est satisfaisable, on peut donc choisir k de sorte à annuler x_0. Réciproquement, toute solution de S est solution du sous-système $\mathbf{A}\boldsymbol{x} \le \boldsymbol{b}$. ■

Propriété S *est satisfaisable si et seulement si le graphe des inégalités de* S *ne contient pas de circuit absorbant.*

$\mathbf{A}\boldsymbol{x} \le \boldsymbol{b}$ est insatisfaisable si et seulement si $0 \le -1$ est une inégalité conséquence du système. D'après le lemme de Farkas, ceci revient à dire qu'il existe $\boldsymbol{y} \ge 0$ vérifiant $\mathbf{T}\boldsymbol{y} = 0$ et $\boldsymbol{by} \le -1$ (avec $\mathbf{T} = {}^t\mathbf{A}$). Le vecteur $\boldsymbol{y}$ (qui appartient au noyau de **T)** se décompose en une somme[1] de circuits dont la longueur totale est strictement négative. Il existe donc au moins un circuit[2] de longueur strictement négative. ■

1. Il s'agit ici d'une combinaison linéaire à coefficients +1 (valeur absolue des coefficients du système).

On a pour un tel circuit :

$$\begin{aligned} x_{j_2} - x_{j_1} &\le b_{j_1 j_2} \\ x_{j_3} - x_{j_2} &\le b_{j_2 j_3} \\ x_{j_4} - x_{j_3} &\le b_{j_3 j_4} \\ \dots &\le \dots \\ x_{j_1} - x_{j_k} &\le b_{j_k j_1} \\ \hline 0 = \quad x_{j_1} - x_{j_1} &\le b_{j_1 j_2} + b_{j_2 j_3} + \dots + b_{j_k j_1} \le -1 \end{aligned}$$

Remarque La satisfaisabilité du système dual est acquise dès lors que les valuations sont positives (puisqu'avec $\boldsymbol{b} \ge 0$, $\boldsymbol{x} = 0$ est solution).

4.4 Monotonie et optimalité

Toutes les inégalités du problème dual (on parlera de contraintes de tensions) font intervenir deux variables avec des coefficients opposés +1 et –1.
Le système $\mathbf{A}\boldsymbol{x} \le \boldsymbol{b}$ est donc *1-monotone*. Il le demeure si l'on élimine le potentiel x_o avec le rajout de l'équation $x_o = 0$.

Théorème VI.3 *Lorsque o est racine du graphe et qu'il n'y a pas de circuit absorbant, les potentiels solutions admettent un plus grand élément* $\boldsymbol{x}^*$ *dont chaque composante,* x_j^**, est entière si* $\boldsymbol{b}$ *est entier, et a pour valeur la longueur du plus court chemin de o à j.*

Démonstration En effet, que *o* soit racine du graphe signifie qu'il existe un chemin entre le sommet origine et tout autre sommet (le cas 2 est exclu). Donc, quelle que soit la destination *d* sur le graphe, la longueur d'un chemin de *o* à *d* (et d'un chemin élémentaire en particulier) est un majorant de $\{x_d \mid \mathbf{A}\boldsymbol{x} \le \boldsymbol{b}, x_o = 0\}$. En l'absence de circuit absorbant, l'ensemble $P = \{\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} \le \boldsymbol{b}, x_o = 0\}$ des potentiels compatibles n'est pas vide et a toutes ses projections majorées par $|N|.\| \boldsymbol{b} \|_\infty$ [1]. Il résulte du théorème **III.4** que max P existe et appartient à $\overline{\mathbb{Z}}^{|N|}$.

L'interprétation de l'optimum en termes de plus court chemin vient du cas 3, la projection de max P sur l'axe x_j étant égale à $\max\{x_j \mid \mathbf{A}\boldsymbol{x} \le \boldsymbol{b}, x_o = 0\}$. ■

Cette propriété de structure explique pourquoi il n'est guère plus difficile de résoudre le problème du plus court (ou plus long) chemin entre deux sommets que de résoudre ce même problème entre un sommet et tous les autres.

2. On peut imposer à ce circuit d'être *élémentaire*, c'est-à-dire, d'avoir tous ses sommets distincts (à l'exclusion du dernier qui est identique au premier).
Un graphe fini contient un nombre fini de circuits élémentaires et tout circuit absorbant contient un circuit élémentaire absorbant.

1. Tout chemin élémentaire sur le graphe a au plus $|N|-1$ arcs.

Le tableau suivant résume les questions de satisfaisabilité et d'optimalité posées par le dual.

Problème de décision	Problème d'optimisation
existence d'une tension compatible	*calcul des plus forts potentiels*
$P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}, x_o = 0\} \neq \varnothing$	max P

La formulation du problème en tensions exprimée ci-dessus revient à une maximisation simultanée des valeurs des potentiels aux nœuds :

$$\max \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}, x_o = 0\} \qquad \text{(III.3)}$$

En ce sens, elle est plus générale que la formulation en courants. Toute solution optimale de (III.3) est une solution optimale particulière de (III.2) :

$$\max\{\boldsymbol{c}\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}\} = x_d^* = \text{proj}_d \max \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}, x_o = 0\}$$

Exemple Dans le triangle du plan (x, y) situé sous la diagonale principale du carré unité, deux sommets satisfont au critère de maximisation des abscisses : (1, 0) et le maximum global (1, 1). La solution de base calculée par le simplexe correspond à l'un des deux. Avec Prolog III, par exemple, on obtient :

```
> maximize(x) particular_value(y,z), {0<= x <= 1, 0<= y <= 1, y <= x};
    {x = 1, z = 0, -y + 1 >= 0, y >= 0}

> maximize(x) maximize(y), {0 <= x <= 1, 0 <= y <= 1, y <= x} ;
    {x = 1, y = 1}
```

Propriété *Le polyèdre des tensions compatibles possède une structure de treillis.*

L'ensemble de ses solutions du système d'inégalités monotones $\mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}$ forme un demi-treillis supérieur. Comme il est égal à l'ensemble des solutions du système $(-\mathbf{A})\boldsymbol{x} \geq -\boldsymbol{b}$ qui est lui-même monotone, il forme également un demi-treillis inférieur. ■

De même que l'on a :

$$\begin{aligned} \text{proj}_d \max \{\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}, x_o = 0\} &= \max \{x_d - x_o \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}\} \\ &= \textit{plus courte distance de } o \textit{ à } d \end{aligned}$$

on peut donc donner une interprétation en termes de plus courts chemins au minimum global de l'ensemble des solutions (lorsqu'il existe) :

$$\begin{aligned} \text{proj}_d \min \{\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}, x_o = 0\} &= \min \{x_d - x_o \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}\} \\ &= -\max \{-(x_d - x_o) \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}\} \\ &= -\max \{x_o - x_d \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}\} \\ &= -\textit{ plus courte distance de } d \textit{ à } o \end{aligned}$$

Remarque Dans le cas où o n'est pas racine du graphe, le calcul du sup dans $\mathbb{R} \cup \{+\infty\}$ peut se substituer, pour certaines composantes, au calcul du max. En effet :

- tous les nœuds situés sur une arborescence issue de o ont une valeur finie ;
- les autres, étant inaccessibles, sont à une distance infinie de o.

4.5 Systèmes à variables entières

Du fait de la totale unimodularité de **A**, le théorème d'intégrité s'applique tout aussi bien à la formulation duale qu'à la formulation primale. Nous le redémontrons ici par des moyens plus faibles, qui ne font appel qu'au caractère monotone du système de contraintes de tensions.

Propriété *Les systèmes de contraintes de tensions sont stables par élimination de Fourier.*

Soit ($C_1 : x_k - x_j \leq b_{jk}$, $C_2 : x_j - x_l \leq b_{lj}$) une paire de contraintes complémentaires vis à vis d'une variable x_j ($j \neq k, j \neq l$). L'inégalité résultant de l'élimination de x_j exprime encore une différence de potentiels :

$C_1 + C_2 : x_k - x_l \leq b_{jk} + b_{lj} = b'_{lk}$ avec b'_{lk} entier. ■

En particulier, les projections sur les axes du polyèdre des solutions sont des intervalles dont les bornes sont entières si elles sont finies.

Propriété *Un système de contraintes de tensions est satisfaisable sur les entiers si et seulement si il est satisfaisable sur les réels*

Découle immédiatement du théorème **III.3**. ■

Propriété *Les sommets du polyèdre des solutions d'un système de contraintes de tensions sont à coordonnées entières.*

Soit $P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}, x_o = 0\}$ le polyèdre des solutions d'un système de contraintes de tensions. Tout sommet $\boldsymbol{s}$ de P peut être défini comme l'unique intersection d'un nombre fini d'hyperplans supports : $\mathbf{A'}\boldsymbol{x} = \boldsymbol{-b'}$ (où $\mathbf{A'}$ et $\boldsymbol{b'}$ sont obtenus en éliminant des lignes). On a donc :

$$\{\boldsymbol{s}\} = \{\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}, x_o = 0, \boldsymbol{-\mathbf{A'}x} \leq \boldsymbol{-b'}\} = \max \{\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}, x_o = 0, \boldsymbol{-\mathbf{A'}x} \leq \boldsymbol{-b'}\}$$

Le résultat découle du caractère 1-monotone du système obtenu. ■

5 Identification des plus courts chemins

Une fois que l'on dispose de l'information sur plus courtes distances entre la racine o et chacun des sommets du graphe, il est relativement aisé de reconstituer les chemins optimaux de o à d :

- le potentiel maximal en d, x_d^*, donne la longueur d'un plus plus court chemin de o à d.
- les arcs d'un tel chemin sont les $y_{j \to k} \geq 1$ d'une solution entière optimale du modèle primal.

Dans le modèle dual, ces arcs correspondent aux inégalités saturées :

$$x_k^* - x_j^* = b_{jk}\,.$$

Cette équation, déduite du théorème des écarts complémentaires, traduit en l'espèce le «principe d'optimalité». Si un plus court chemin de o à k (de longueur x_k^*) passe par j avant d'arriver à k, alors

- d'une part, le chemin suivi de o à j est optimal pour arriver à j (il a pour longueur x_j^*)
- d'autre part, la longueur de l'arc $j \rightarrow k$ est celle du plus court chemin de j à k.

Pour retrouver un tel chemin, il suffit de progresser à reculons à partir de d, en sélectionnant un arc qui pointe vers cette destination et réalise la plus courte distance entre ses extrémités. Son origine devient alors la nouvelle destination, et ainsi de suite jusqu'à rejoindre o.

Le processus est non-déterministe — à l'inverse du calcul d'optimisation. Mais le résultat de ce calcul préalable contraint le problème et permet l'énumération.

Le programme Prolog IV suivant, `pcc/3`, en donne une spécification logique à l'aide d'une règle d'arrêt et d'une règle récursive. Pour la clarté des notations, les variables `A` et `Z` représentent l'origine et la destination (supposées connues) sur le graphe, `Y` étant un sommet intermédiaire. `[]` désigne la liste vide. Le constructeur de liste `[E|L]` rajoute un élément `E` en tête de la liste `L`.

Le prédicat `dist/2` fait référence à la table des plus courtes distances calculées depuis l'origine. Le prédicat `arc/3` décrit les arcs et leur valuation (qu'on supposera ici strictement positive[1]).

```
pcc(A, A, []).                  % Il y a un chemin min. vide de A à A.

pcc(A, Z, [(Y,Z)|C]) :-        % Il y a un chemin min. non vide de A à Z
                                % commençant par C et se terminant par (Y,Z) si :
      dist(Z, Xz),              % Z étant à une distance min. Xz de A,
      arc(Y, Z, Byz),           % (Y,Z) étant un arc de longueur Byz,
      dist(Y, Xy),              % la distance min. de A à Y
      Xz = Xy + Byz,            % vaut Xz – Byz,
      pcc(A, Y, C).             % et C est un chemin min. de A à Y.
```

La machine Prolog produit, par *backtracking*, tous les chemins élémentaires (inversés), `C`, solutions de la requête `pcc(A,Z,C)`.

1. Autrement, il peut y avoir un nombre infini de chemins optimaux.
Pour garantir la terminaison de l'algorithme et ne pas tomber dans des circuits de longueur nulle, il convient alors de limiter l'énumération à des chemins élémentaires (en marquant les sommets visités).

6 Algorithmes de résolution

6.1 Remarques historiques

Dans sa formulation linéaire (III.1), le problème du plus court chemin apparaît comme un cas très particulier de «problème de transport», sur un réseau ferré par exemple, entre plusieurs origines et plusieurs destinations :

$$\min\{\boldsymbol{by} \mid \mathbf{T}\boldsymbol{y} = \boldsymbol{c}, \boldsymbol{y} \geq 0\}$$

avec $c_j < 0$ aux points de production (généralisant o) et $c_j > 0$ aux points de consommation (généralisant d).

Ce problème fut étudié de façon pionnière par Kantorovitch, dès 1939. Il en donna une méthode de résolution reposant essentiellement sur ce que l'on appelle aujourd'hui une analyse primale-duale. En voici, avec nos notations, le théorème :

Pour qu'un plan admissible $\boldsymbol{y} \in \mathbb{R}^{|A|}$ *(satisfaisant aux conditions* $\mathbf{T}\boldsymbol{y} = \boldsymbol{c}$, $\boldsymbol{y} \geq 0$*) soit optimal, il faut et il suffit qu'il existe des potentiels* $\boldsymbol{x} \in \mathbb{R}^{|N|}$ *tels que :*

a. $x_k - x_j \leq b_{jk}$ (pour tout arc $j \to k$ du graphe) ;

b. $x_k - x_j = b_{jk}$ si $y_{j \to k} \neq 0$.

Dans le cas d'un plus court chemin, $\boldsymbol{c}$ correspond à la colonne (o, d).

Les égalités **(b)** fournissent un certificat d'optimalité à la solution du primal.

On a vu qu'elle servaient à identifier un plus court chemin quand toutes les distances minimales étaient connues. Lorsque tel n'est pas le cas, elles conduisent aux équations (non-linéaires) de Bellman :

$$x_o = 0$$
$$x_j = \min_{\{k \mid k \to j \in A\}} \{x_k + b_{kj}\} \text{ pour } j \neq o.$$

Ces équations permettent de déterminer les x_j par récurrence (en suivant un ordre topologique sur les sommets) lorsque le graphe est sans circuit[1].

6.2 Réduction d'intervalles et algorithme de Ford

La formulation duale (III.3) se prête bien à une résolution par les intervalles. Appliqué au polyèdre

$$\mathrm{P} = \{\boldsymbol{x} \in \mathbb{R}^{|N|} \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}, x_o = 0\}$$

le théorème **IV.4** énonce que si P admet un maximum (et si la discrétisation de l'espace est suffisante), alors le maximum de P a pour coordonnées les maxima (entiers) des domaines réduits des variables.

1. Leur transcription en programmation par contrainte peut s'affranchir de cette restriction.

Autrement dit, en numérotant les variables à partir de o :

$$\max \mathrm{P} = \boldsymbol{x}^* = (0, \max X_1, \ldots, \max X_{|N|-1})$$

La collecte de cette information se fait en temps constant, sans énumération ni effort d'optimisation supplémentaire, par accès aux bornes supérieures des domaines : $x_j^* = \max X_j$.

L'existence d'un maximum pour P suppose :

1. que le graphe ne possède pas de circuit absorbant
2. que o est racine.

Remarque Les x_j étant alors majorés par la longueur d'un chemin $c = (a_i)_{i \in \mathrm{I}}$ de o à j :

$$x_j = x_j - x_o = \sum_{k \to l \in \mathrm{I}} (x_l - x_k) \leq \sum_{k \to l \in \mathrm{I}} b_{kl} = l(c)$$

tous les domaines des variables accessibles à partir de $x_o = 0$ sont, de proche en proche, automatiquement réduits. On n'a donc pas besoin d'expliciter les bornes par excès : $x_j \leq |N|.\| \boldsymbol{b} \|_\infty$ pour que le théorème **IV.4** s'applique.

Remarque Si o n'est pas racine du graphe, le résultat tient encore pour les sommets accessibles à partir de o. Les autres sont à une distance de $+\infty$.

Nous pouvons condenser ces résultats dans la proposition suivante :

Théorème VI.4 *Soit* A *la matrice d'incidence d'un graphe dont 0 est un sommet. Si le système* $\mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}$, $x_o = 0$ *est satisfaisable (et la discrétisation de l'espace suffisante), alors, à l'issue du calcul de point fixe, les intervalles réduits* X_j *des variables donnent les projections exactes sur les axes — leurs bornes indiquant les plus courtes distances à partir de et vers l'origine 0 :*

$$\sup X_j = \textit{plus courte distance de 0 à } j$$
$$-\inf X_j = \textit{plus courte distance de } j \textit{ à } 0$$

D'un point de vue algorithmique, la matrice d'incidence arcs-sommets **A** n'ayant que deux coefficients non nuls par ligne, les formules de réduction de domaines prennent ici une forme particulièrement simple.

Reprenons les formules **(1.1)** du chapitre IV. En notant I_j et I_k les intervalles courants des variables x_j et x_k et D_{kj} la relation $x_j - x_k \leq b_{kj}$, on a :

$$\mathrm{apx}(\mathrm{proj}_j(D_{kj} \cap I_j \times I_k)) = I_j \cap J_j = I_j'$$
$$\text{avec : } \sup(J_j) = \lceil \sup(I_k) + b_{kj} \rceil$$

En supposant que l'on travaille dans l'espace discrétisé $(\mathbb{R}, \Im_M)$, on peut, si max P existe, prendre M suffisamment grand pour s'assurer que tous les calculs sur les bornes supérieures restent exacts. Dans ce cas, les bornes étant entières, la formule du sup se simplifie en :

$$\sup(J_j) = \sup(I_k) + b_{kj}$$

En d'autres termes, la contrainte D_{kj} produit la réduction $I_j \times I_k \rightarrow I_j' \times I_k$ avec

$$\sup(I_j') = \min\{\sup(I_j)\ ,\ \sup(I_k) + b_{kj}\}$$

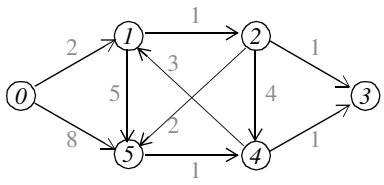
Propriété *Cette diminution progressive de la borne supérieure du domaine des variables x_j correspond à l'application de l'algorithme de Ford[1] (1956):*

1. A tout sommet $j \neq o$, on associe une marque S_j initialisée à $+\infty$ ($S_o = 0$).
2. Tant qu'il existe un arc $k \rightarrow j$ vérifiant $S_j > S_k + b_{kj}$, on écrit $S_j := S_k + b_{kj}$
3. Lorsqu'une telle diminution n'est plus possible, la marque S_j est égale à la longueur d'un plus court chemin de o à j.

Lorsque le graphe ne contient pas de circuit absorbant, l'algorithme de Ford s'arrête au bout d'un nombre fini d'itérations.
De même que pour la réduction d'intervalles, sa complexité est en $O(nm)$, avec $n = |N|$ et $m = |A|$.

S'il n'y a pas de chemin allant de o à j, à la fin de l'algorithme, on a $S_j = +\infty$.

Exemple Recherche des plus courts chemins du sommet 0 à tous les autres sommets sur un graphe avec circuits, d'ordre $|N| = 6$, de taille $|A| = 10$, ne présentant que des valuations positives :




```
tensions([X0,X1,X2,X3,X4,X5]) :-
          X1.-.X0 le 2,
          X5.-.X0 le 8,
          X5.-.X1 le 5,
          X2.-.X1 le 1,
          X5.-.X2 le 2,
          X4.-.X5 le 1,
          X1.-.X4 le 3,
          X4.-.X2 le 4,
          X3.-.X2 le 1,
          X3.-.X4 le 1.
```

Les maxima correspondant aux distances minimales recherchées s'obtiennent immédiatement en instanciant x_0 à 0[2].

```
>>  tensions(Xs), Xs = [0|_].
    Xs ~ [0,le(2),le(3),le(4),le(6),le(5)].        % x5 ≤ 5
```

Le plus court chemin de 0 à 5 est ainsi de longueur 5 : $\sup(X_5) = 5$.
Si l'on prend maintenant 5 pour origine sur le graphe :

```
>>  tensions([X0,_,_,_,_,0]), glb(X0, B).
    B = -5,
    X0 ~ ge(-5).                                   % -5 ≤ x0
```

le fait que x_0 ne soit pas majoré indique qu'il n'y a pas de chemin de 5 à 0.

1. Cf. ROSEAUX, tome 1, p.111. Gondran & Minoux citent également Berge (1958).
2. Il suffit de poser $x_0 \leq 0$ pour le calcul du maximum global.

La borne inférieure[1] redonne alors l'inverse de la plus courte distance de 0 à 5. La réduction d'intervalles s'avère en revanche quasi inopérante sur la formulation en courants.

```
courants([Y01,Y05,Y15,Y12,Y25,Y54,Y41,Y24,Y23,Y43], L) :-
                    .-. Y01 .-. Y05 = -1,              % origine 0
        Y01 .-. Y15 .-. Y12 .+. Y41 =  0,
        Y12 .-. Y25 .-. Y24 .-. Y23 =  0,
                        Y23 .+. Y43 =  0,
        Y54 .-. Y41 .+. Y24 .-. Y43 =  0,
        Y05 .+. Y15 .+. Y25 .-. Y54 = +1,              % destination 5

    Y01 ge 0, Y05 ge 0, Y15 ge 0, Y12 ge 0, Y25 ge 0,
    Y54 ge 0, Y41 ge 0, Y24 ge 0, Y23 ge 0, Y43 ge 0,

    L = 2.*.Y01 .+. 8.*.Y05 .+. 5.*.Y15 .+. Y12 .+. 2.*.Y25
        .+. Y54 .+. 3.*.Y41 .+. 4.*.Y24 .+. Y23 .+. Y43.
```

On en tire simplement que la longueur l d'un chemin de 0 à 5 est positive :

```
>>  courants(Ys, L).
    Ys ~ [cc(0,1),cc(0,1),ge(0),ge(0),ge(0),ge(0),ge(0),ge(0),0,0],
    L ~ ge(0).
```

Mais si l'on impose la valeur $l = 5$, calculée avec la formulation duale, on déduit qu'il n'existe qu'un seul chemin optimal de 0 à 5 : $0 \rightarrow 1 \rightarrow 2 \rightarrow 5$.

```
>>  courants(Ys, 5), tous_entiers(Ys).     % en posant int(Yi) pour tout i ∈ A
    Ys = [1,0,0,1,1,0,0,0,0,0].
```

On obtiendrait un résultat identique avec un programme linéaire minimisant l [2].

Quand le graphe présente des circuits absorbants, l'algorithme de Ford ne s'arrête plus. Et la réduction d'intervalles qui, par construction, est assurée de converger, ne le fait que très lentement.

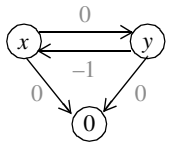
Pour les sommets atteignables à partir d'un circuit absorbant, les bornes supérieures des domaines des variables régressent alors, à petits pas, vers $-\infty$. La finitude de l'approximation et la présence de la fonction partie entière font que ces bornes se stabilisent aux alentours de la valeur $-M$. Les domaines réduits retournés sont alors de la forme $]-\infty, -M+\varepsilon[$ (et inversement, pour les bornes inférieures, $]M-\varepsilon, +\infty[$). L'insatisfaisabilité du problème dual n'est donc pas détectée[3]. De plus, ce résultat est obtenu au bout d'un nombre d'itérations considérable, proche du pire des cas où tous les points de la droite discrète sont passés en revue ($2Mnm$ si $n = |N|$ et $m = |A|$; $8556380159nm$ pour Prolog IV).

Ce phénomène de *convergence lente* (qui n'est pas à craindre avec $\boldsymbol{b} \geq 0$) peut se produire sur des très petits exemples et se traduire par des temps de réponse prohibitifs.

1. La primitive Prolog IV pour le calcul de l'inf. (resp. du sup.) est `glb/2` (resp. `lub/2`).
2. Avec les intervalles, la minimisation nécessiterait une exploration arborescente, procédant par énumération (sur les entiers) ou dichotomie (sur les réels).
3. Il faudrait pour cela rajouter des contraintes exprimant que les longueurs x_j des chemins élémentaires solutions sont minorées par $-|N|.\|\boldsymbol{b}\|_\infty$.

Exemple Si l'on donne à l'arc $1 \to 5$ la valeur -5 dans l'exemple précédent, on créé un circuit absorbant $1 \to 5$, $5 \to 4$, $4 \to 1$ de longueur -1. On obtient alors, en s'armant de patience :

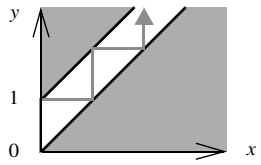
```
>> tensions_bis(Xs), Xs = [0|_].
Xs ~ [0, le(-1.6777212e7), le(-1.6777211e7), le(-1.6777214e7),
                           le(-1.6777216e7), le(-1.6777215e7)].
```

Exemple Prenons le graphe 

. Les contraintes de tensions associées :

$y \geq x + 1$, $y \leq x$, $x \geq 0$, $y \geq 0$ forment un système clairement insatisfaisable. Mais les réductions d'intervalles qui s'effectuent par pas de 1, finissent par s'arrêter à la limite de la grille de discrétisation.

de $x \geq 0$ et de $y \geq x + 1$, on infère $y \geq 1$
de $y \geq 1$ et de $x \geq y$, on infère $x \geq 1$
...
de $x \geq k$ et de $y \geq x + 1$, on infère $y \geq k + 1$
de $y \geq k + 1$ et de $x \geq y$, on infère $x \geq k + 1$
...

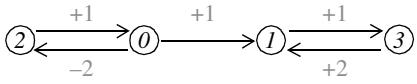


D'où, au bout de tous ces calculs, la réponse positive :

```
X ~ ge(1.6777216e7), Y ~ ge(1.6777216e7). .
```

signifiant «il y a peut-être des solutions au-delà» — alors que les droites sont parallèles et ne peuvent se croiser.

Exemple Avec le graphe du début du chapitre :




```
tensions_ter([X0,X1,X2,X3]) :-
                X1.-.X0 le  1,
                X2.-.X0 le -2,
                X0.-.X2 le  1,
                X3.-.X1 le  1,
                X1.-.X3 le  2.
```

en prenant 1 pour origine, on obtient[1], au bout d'un temps cpu de 623216 ms:

```
>>  tensions_ter([X0,X1,X2,X3]), X1 = 0.
    X1 = 0, X3 ~ cc(-2,1),
    X2 ~ ge(1.6777215e7), X0 ~ ge(1.6777216e7).
```

En soumettant le même jeu d'inégalités au Simplexe, on obtiendrait immédiatement un diagnostic d'insatisfaisabilité.

En résumé, l'application des techniques de réduction d'intervalles au calcul des plus courts chemins réalise «gratuitement» l'optimisation simultanée des potentiels x_j, comme conséquence d'une recherche de satisfaisabilité.
Mais la complétude de cette recherche n'est garantie que si les x_j sont bornés.

Lorsque les les b_i sont de signe quelconque, des phénomènes de convergence lente (ou d'unification bloquante) sont à redouter.

1. Alors que le système est globalement insoluble, on obtient la plus courte distance de 1 à 3...

Le lecteur intéressé trouvera dans les travaux de Lhomme des idées pour détecter et éradiquer à la volée certains phénomènes cycliques à l'origine de cas de convergence lente dans les langages de programmation par contraintes.

D'autres approches récentes (Bockmayer & Pisaruk, Régin & Rueher) se proposent d'intégrer sous forme de «contraintes globales» des algorithmes spécialisés de calcul de flots ou de plus courts chemins. Venant se substituer au mécanisme général de réduction d'intervalles, ils peuvent — dans un contexte bien précis — en pallier l'incomplétude et l'éventuelle inefficacité.

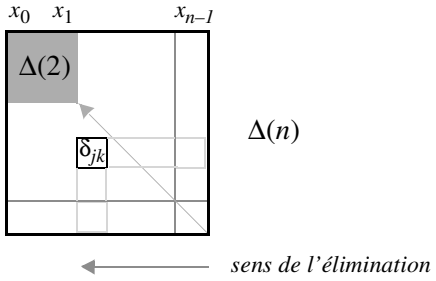
6.3 Elimination de Fourier et algorithmes matriciels

La spécialisation de l'élimination de Fourier aux systèmes linéaires (III.3) va répondre au problème, en donnant une procédure de résolution non seulement complète mais également polynômiale.

La raison pour laquelle l'élimination de Fourier est polynômiale en fonction du nombre de variables (ou de sommets du graphe), $n = |N|$, est qu'on ne peut exprimer, sans redondance, plus de n^2 contraintes majorant les différences de potentiels entre deux nœuds. Il suffit de retenir, lors de chaque élimination, le minimum des majorants des différences $x_j - x_k$ pour échapper au risque d'explosion du nombre d'inégalités ($1 \le j \le n$, $1 \le k \le n$).
Comme il y a n–1 variables à éliminer et que l'on part d'au plus n^2 inégalités, on obtient un algorithme en $O(n^3)$ [1].

Une manière concise de représenter ces inégalités $(x_k - x_j \le \delta_{jk})_{j \ne k}$ est de disposer leur membre droit δ_{jk} dans un tableau carré de taille n, à l'intersection de la ligne j et de la colonne k, en omettant cette constante si l'inégalité n'existe pas ou, de façon équivalente, en inscrivant $+\infty$ dans la case (j, k). Une diagonale invariablement nulle complète ce tableau.

Si l'on élimine les variables du système $\mathbf{A}\boldsymbol{x} \le \boldsymbol{b}$, $x_0 = 0$ dans l'ordre des indices décroissants (de sorte à ne pas toucher à la variable factice x_0), l'algorithme de Fourier produit des tableaux de taille décroissante : $\Delta(n)$, $\Delta(n-1)$, …, $\Delta(2)$.



1. Cette spécialisation de l'algorithme de Fourier peut être rapprochée du test de satisfaisabilité de Pratt (1977), de complexité cubique (en $n + m$ d'après Shostak), qui est fondé sur la détection de circuits de longueur négative.
L'application de l'algorithme de Ford-Bellman, qui autorise également les longueurs négatives, donnerait $O(n.m)$, soit $O(n^3)$ pour des graphes denses.

$\Delta(n) = (\delta_{jk}{}^0)_{j,k \le n-1}$, qui correspond au système S de départ, est initialisé avec les valeurs associées aux arcs du graphe des contraintes :

$$\delta_{jk}{}^0 = \begin{cases} b_{jk} \text{ si } j \to k \text{ est un arc du graphe} \\ +\infty \text{ sinon} \end{cases} \qquad \text{si } j \neq k$$

$$\delta_{jj}{}^0 = 0$$

$\Delta(n-p) = (\delta_{jk}{}^p)_{j,k \le n-p-1}$, qui correspond à $E_F(S, (x_{n-1}, \ldots, x_{n-p}))$, est calculé à partir de $\Delta(n-p+1)$ par élimination de la variable x_{n-p} :

$$\delta_{jk}{}^p = \inf\{\delta_{jk}{}^{p-1}, \delta_{j,n-p}{}^{p-1} + \delta_{n-p,k}{}^{p-1}\} \qquad \text{si } j \neq k$$

$$\delta_{jj}{}^p = 0 \qquad \text{si } \delta_{j,n-p}{}^{p-1} + \delta_{n-p,j}{}^{p-1} \ge 0$$

L'apparition d'une valeur négative et non nulle sur la diagonale d'un tableau est automatiquement synonyme d'insatisfaisabilité.

Remarque Si $\delta_{j,n-p}{}^{p-1} + \delta_{n-p,j}{}^{p-1} = 0$, l'élimination de x_{n-p} produit l'inégalité $0 \le 0$, signe que les inégalités entre x_{n-p} et x_j sont en fait des égalités : $x_{n-p} - x_j = \delta_{j,n-p}{}^{p-1}$.

Au bout de $n-2$ éliminations, si l'on n'a pas produit d'élément diagonal < 0, on obtient le tableau suivant :

$\Delta(1)$	x_0	x_1
x_0	0	$\delta_{01}{}^{n-2}$
x_1	$\delta_{10}{}^{n-2}$	0

Alors S est satisfaisable si et seulement si : $\delta_{01}{}^{n-2} + \delta_{10}{}^{n-2} \ge 0$, auquel cas $\delta_{01}{}^{n-2}$ représente la plus courte distance du sommet origine 0 au sommet 1.

Connaissant la plus courte distance de 0 à 1, on peut affecter sa valeur à x_1, puis calculer, en remontant, la plus courte distance de 0 à 2, et ainsi de suite jusqu'à n, en «maximisant» à tour de rôle les projections des solutions de S (puisque l'on sait que ces maxima sont compatibles) :

$$\begin{aligned} x_0{}^* &= 0 \\ x_1{}^* &= \delta_{01}{}^{n-2} \\ x_2{}^* &= \inf\{\delta_{02}{}^{n-3}, x_1{}^* + \delta_{12}{}^{n-3}\} \\ &\ldots \\ x_{n-p}{}^* &= \inf_{j \le n-p-1}\{x_j{}^* + \delta_{j,\,n-p}{}^{p-1}\} \\ &\ldots \\ x_{n-1}{}^* &= \inf_{j \le n-2}\{x_j{}^* + \delta_{j,\,n-1}{}^0\} \end{aligned}$$

On récupère ainsi, en $O(n^2)$ opérations supplémentaires, les plus courtes distances du sommet 0 à tous les autres, soit $\boldsymbol{x}^* = \sup\{\boldsymbol{x} \in S\}$.

Remonter la «solution» $\boldsymbol{x}^*$ de S revient à remplir la première ligne du tableau avec les longueurs des plus courts chemins de 0 à $j \neq 0$. Une légère modification de l'algorithme (dont nous verrons qu'elle n'affecte pas sa complexité) permet de remplir tout le tableau avec les longueurs des plus courts chemins entre tous les sommets. Cette méthode matricielle a pour effet de regrouper les calculs des n problèmes :

$$\max\{\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}, x_j = 0\}, \text{ pour } j \text{ variant de } 0 \text{ à } n-1.$$

Le tableau de départ, $\Delta(n)$, est défini comme précédemment. La taille des tableaux restant égale à n, les tableaux intermédiaires $\Delta(n-p) = (\delta_{jk}^{\,p})_{j,k \leq n}$ s'obtiennent par des formules similaires, à ceci près que l'on met maintenant à jour toutes les cases du tableau, au lieu de se limiter aux indices inférieurs à $n-p-1$:

$$\delta_{jk}^{\,p} = \inf\{\delta_{jk}^{\,p-1}, \delta_{j,n-p}^{\,p-1} + \delta_{n-p,k}^{\,p-1}\} \quad \text{si } j \neq k$$

$$\delta_{jj}^{\,p} = 0 \quad \text{si } \delta_{j,n-p}^{\,p-1} + \delta_{n-p,j}^{\,p-1} \geq 0$$

$\delta_{jk}^{\,p}$ représente la longueur du plus court chemin de j à k n'ayant pour sommets intermédiaires que $\{n-1, n-2, ..., n-p\}$. Au choix de l'ordre près, on retrouve les équations de l'*algorithme de Floyd* (1962):

$$\delta_{jk}^{\,p} = \inf\{\delta_{jk}^{\,p-1}, \delta_{j,p}^{\,p-1} + \delta_{p,k}^{\,p-1}\}$$

L'apparition d'une valeur $\delta_{jj}^{\,p} < 0$ signale l'insatisfaisabilité du système.

Le calcul du tableau final $\Delta = \Delta(1) = (\delta_{jk})_{j,k \leq n}$ nécessite une triple boucle (sur p, j et k). D'où une complexité en $O(n^3)$.

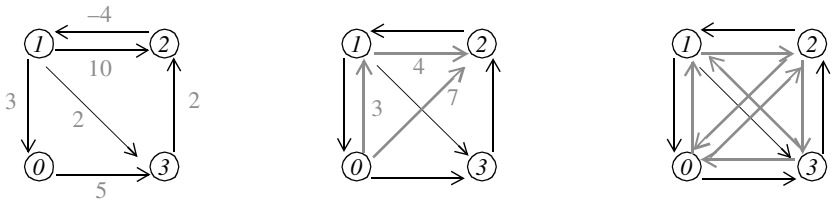
Réalisant la clôture réflexo-transitive du graphe G, la matrice Δ donne une représentation *canonique* du polyèdre des solutions de S.

Cette représentation matricielle des bornes des différences $x_k - x_j$ est appelée *Difference Bounds Matrix* par Dill (DBM). Ses avantages sont les suivants :

- pour toute paire de sommets (j, k), δ_{jk} est la plus courte distance de j à k
- la projection de P parallèlement à un axe x_j a pour représentation canonique la matrice carrée extraite de Δ dans laquelle on a supprimé la ligne et la colonne d'indice j
- une inégalité $x_k - x_j \leq b$ est redondante par rapport à S dès lors que $\delta_{jk} \leq b$
- le système $S \cup \{x_k - x_j \leq b\}$ est satisfaisable si et seulement si $0 \leq b + \delta_{kj}$

Exemple Soit S = {$-x_1 \leq 3$, $x_2 - x_1 \leq 10$, $x_1 - x_2 \leq -4$, $x_3 \leq 5$, $x_3 - x_1 \leq 2$, $x_2 - x_3 \leq 2$}
La figure suivante donne les graphes issus : (*a*) de S, (*b*) de l'élimination de Fourier $E_F(S, (x_3,x_2,x_1))$, (*c*) du calcul de la représentation canonique Δ de S.

Δ	**0**	**1**	**2**	**3**
0	0	3	7	5
1	3	0	4	2
2	–1	–4	0	–2
3	1	–2	2	0



(*a*) (*b*) (*c*)

L'élimination de Fourier de x_3 produit les inégalités : $x_2 \leq 7$, $x_2 - x_1 \leq 4$ (resserrée). Celle de x_2 donne : $x_0 - x_1 \leq 3$ (et déduit l'égalité : $x_2 - x_1 = 4$). L'élimination de x_1 conclut à la satisfaisabilité de S — ce que confirme la diagonale nulle de la forme résolue Δ, qui, par comparaison, apparaît redondante.

Δ montre en définitive que l'on a[1] :
$x_3 - x_1 = 2$, $x_3 - x_2 = -2$, $x_2 - x_1 = 4$, $-3 \leq x_1 \leq 3$, $1 \leq x_2 \leq 7$, $-1 \leq x_3 \leq 5$.

1. Les équations implicites détectées, lors de le l'élimination de Fourier, entre les variables x_j et x_k, se retrouvent dans le tableau matriciel sous la forme : $\delta_{jk} = -\delta_{kj}$.

7 Problèmes d'ordonnancement

Dans un problème de pur ordonnancement, les inconnues x_j sont les dates de démarrage des différentes tâches constitutives d'un projet et les contraintes proviennent des dépendances entre ces tâches.

Définition On dit qu'une tâche en *précède* une autre lorsque cette dernière ne peut démarrer avant la fin de la première.

Si j précède k, on a : $x_k \geq x_j + b_j$ où b_j représente la durée de la tâche j.

Un *projet* d'ordonnancement est donc défini par la *durée* de chaque tâche et les *précédences* entre ces tâches.

Des *fenêtres de temps* peuvent également borner les dates de démarrage des différentes tâches, en particulier, celles marquant le début (x_0) et la fin (x_n) du projet. L'objectif généralement admis est d'achever le projet au plus tôt, en minimisant sa durée totale.

Formellement, un tel problème d'ordonnancement se résume à un programme linéaire :

$$\min\{x_n \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b},\ x_0 = 0\}$$

dans lequel $\mathbf{A}$ désigne une matrice d'incidence arcs-sommets.

Graphiquement, la représentation *potentiels-tâches* de Roy (1960) fait correspondre à chaque tâche j, un sommet et à chaque contrainte de précédence entre j et k, un arc valué par la durée de j. Notons que ce graphe doit être sans circuit (cela signifierait sinon qu'une tâche précède elle-même).

Par rapport au problème précédent, on voit que le sens des inégalités est inversé, de même que l'objectif qui devient une minimisation. Le problème dual correspond maintenant au calcul d'un *plus long chemin* dans le graphe de précédence. Un tel chemin est appelé *chemin critique*.

Les mêmes algorithmes s'appliquent, en remplaçant «plus court» par «plus long», circuit de longueur strictement négative par circuit de longueur strictement positive, max par min et $+\infty$ par $-\infty$.

En particulier, la résolution par les intervalles est *complète* si le domaine des variables est *borné* (l'horizon est fini, que le temps soit continu ou discret). Il est à noter qu'elle le demeure lorsque les durées ne sont plus données que par des encadrements. Les contraintes de précédence s'écrivent alors :

$$x_k \geq x_j + z_j \text{ avec } \underline{b}_j < z_j < \overline{b}_j.$$

Ces inégalités à 3 (et non plus 2) variables restent en effet monotones. *A posteriori*, c'est probablement l'une des raisons du succès des méthodes à base de programmation par contrainte dans la résolution de problèmes plus complexes d'ordonnancement. Car elles peuvent ainsi filtrer efficacement des permutations de tâches incompatibles.

8 En résumé

Dans ce chapitre, nous avons étudié la résolution de systèmes monotones particuliers : les systèmes de contraintes de tension (ou de précédence) dont les inégalités linéaires, à une ou deux variables (réelles ou entières) et à coefficients entiers, prennent les formes très simples :

$$x \leq \text{constante}, -x \leq \text{constante}, x - y \leq \text{constante}.$$

En introduisant au besoin une nouvelle variable $x_0 = 0$, les contraintes unaires se réécrivent de manière équivalente comme des contraintes de différence ($x \leq b$ devenant $x - x_0 \leq b$). Le système, $\mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}$, s'interprète alors comme le dual d'un problème de plus court chemin sur un graphe dont $\mathbf{A}$ est la matrice d'incidence arcs-sommets, $\boldsymbol{b}$ une valuation et l'origine 0, un des sommets.

Tester la *satisfaisabilité* de ce système revient vérifier l'*absence de circuits absorbants* sur le graphe associé.
A cet effet, la spécialisation de la procédure d'élimination de Fourier fournit un algorithme de décision polynômial.

Calculer les *plus courtes distances* à partir du sommet origine permet d'en exhiber une *solution particulière*, à savoir le maximum global :

$$\max\{\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}, x_0 = 0\}$$

Nous avons montré que, lorsque le système était satisfaible, la résolution par les intervalles pouvait fort bien s'acquitter de ce calcul d'optimisation (problème OPT) et donner les projections exactes des solutions (à la manière de l'algorithme de Ford). Mais qu'elle était en général impuissante à en détecter l'insatisfaisabilité (problème SAT) lorsque les valuations b_i étaient de signe quelconque (ces situations pouvant de plus donner lieu à des temps de calcul rédhibitoires).

Fort heureusement, ces problèmes ne sont pas à craindre pour deux importantes classes d'applications[1] :

- les problèmes de plus courts chemins (où la longueur des arcs est positive)
- les problèmes d'ordonnancement simple à horizon fini.

Dans le continu, les propriétés évoquées se généralisent, sans nécessiter de reformulation, aux problèmes (continus) de flots entre une origine et plusieurs destinations.

1. L'éventuelle «lenteur» de la résolution par les intervalles destine typiquement ces formulations à des applications où celle-ci peut être compensée par une plus grande facilité à intégrer d'autres contraintes.

III

CHAPITRE VII

Vérification de propriétés d'automates synchronisés

Analyse de l'accessibilité dans les Réseaux de Petri

> «Il se trouve que (pour une classe importante de disciplines formelles) le concept de vérité ne coïncide jamais avec celui de démonstrabilité. Car toutes les propositions démontrables sont vraies mais il existe des propositions vraies qui ne sont pas démontrables.»
>
> A. TARSKI

Avec ce chapitre applicatif, nous abordons les méthodes formelles pour la vérification des systèmes et programmes. Nous nous intéressons à l'automatisation de la preuve de conformité d'une réalisation vis à vis d'exigences données, et plus particulièrement, d'exigences de sûreté.

Nous rappelons, en préambule, d'où vient le besoin d'outils de vérification. Nous définissons ensuite les propriétés de sûreté et donnons leur traduction en termes d'accessibilité sur un graphe d'états qui capture la sémantique du système étudié. Enfin, nous introduisons des modèles dont le nombre d'états peut être infini : les réseaux de Petri.

L'analyse de ces modèles est fondée sur un calcul des états accessibles dont on ne sait garantir la terminaison. Une définition intensionnelle peut en être donnée par un programme logique, où la présence de contraintes numériques s'impose naturellement. Reconnaissant en ces systèmes des contraintes linéaires de tensions, nous concluons d'emblée, dans ce contexte, à l'équivalence entre une résolution sur les entiers et une résolution sur les réels. Ce point nous permet de formuler entièrement le problème de l'accessibilité en Prolog III.

Opérationnellement, la stratégie en profondeur d'abord de Prolog ne garantit pas l'exhaustivité de la recherche (et donc la complétude de la procédure de preuve). Il nous faut modifier l'interprète et passer, classiquement, à un mécanisme d'évaluation ascendante. Nous en détaillons l'incidence, d'abord au niveau de la gestion des règles, puis au niveau de la gestion des contraintes — où s'immisce un test d'inclusion.

Après validation de la solution linéaire, nous nous posons la question de la faisabilité d'une solution avec les intervalles. Munis de résultats partiels en complétude, nous montrons comment une petite adjonction au test de satisfaisabilité suffit à garantir l'exactitude des calculs, même lorsque les ensembles que les inégalités délimitent ne sont pas bornés. Nous en tirons une implantation originale en Prolog IV.

Ces développements nous permettent de reproduire expérimentalement, avec une efficacité satisfaisante, les résultats de Bérard & Fribourg (1999) pour l'analyse des réseaux de Petri. Poursuivant (avec notre prototype linéaire) l'application de cette méthode, nous attestons de la puissance des techniques proposées pour réduire la combinatoire du calcul, mais en sortant du cadre monotone. La relaxation sur les réels ne donnant plus qu'un dégrossissage des états accessibles, nous devons restaurer la preuve par une phase finale d'énumération. Nous y parvenons sans difficulté sur un dernier exemple, grâce à la mémorisation d'un arbre d'accessibilité.

1 Philosophie du Model Checking

La conception des systèmes informatiques industriels (matériels ou logiciels, centralisés ou répartis, ouverts ou enfouis) peut présenter des difficultés algorithmiques subtiles. Lorsque plusieurs processus sont susceptibles d'évoluer «en parallèle», les comportements résultant de leurs interactions deviennent difficilement prédictibles : aux mêmes données initiales peuvent correspondre plusieurs exécutions potentielles. Il faut alors s'assurer que toutes ces évolutions sont conformes aux spécifications.

Pour ces programmes où le contrôle est primordial, les techniques dites de *Model Checking* viennent outiller une démarche de vérification qui a sa place dès la phase de conception préliminaire. Mieux vaut en effet s'apercevoir d'une erreur au stade de l'élaboration des plans — en amont du développement, que d'avoir à reprendre entièrement un ouvrage pour cause de vice caché.

Valider *a priori* une conception, c'est porter un diagnostic de correction partielle en examinant une solution à l'aune des exigences fonctionnelles auxquelles elle est censée répondre. Cette analyse s'effectue sur la base de deux documents :

- d'un part, le «plan» de l'automate (machine d'états, système de transitions ou structure de Kripke), décrivant formellement l'ensemble des comportements dynamiques du système proposé ;
- d'autre part, le «cahier des charges» de conception, énumérant les propriétés attendues de cet automate (lesquelles sont très souvent formulées en logique temporelle).

La donnée de l'automate équivaut à celle d'un *graphe* (fini ou infini) ayant pour noeuds les *états globaux* du système et pour arcs les différentes *actions* conduisant à un changement d'état. Les comportements de l'automate s'interprètent simplement comme des cheminements sur ce graphe.

La solution conçue satisfait aux propriétés énoncées lorsqu'au regard des spécifications, aucun cheminement n'apparaît illégal.

La vérification de modèles (ou *Model Checking*) correspond à la résolution mécanique de ce problème de décision. Si la réponse est *oui*, celle-ci certifie que l'automate est un *modèle* (au sens logique) de la propriété. Si la réponse est *non*, elle sert à mettre en évidence une suite d'actions y contrevenant.

2 Automates et propriétés de sûreté

Dans sa plus simple expression, un automate synchronisé est une machine à états discrets, définie par la donnée d'un ensemble (discret) d'états globaux E et d'une relation de transition T de E dans E.

Définition Le graphe orienté $G = (E, T)$ est appelé *graphe d'états* de l'automate.

Définition La fermeture réflexo-transitive de la relation de transition T définit la *relation d'accessibilité* entre états.

Un état final est accessible à partir d'un état initial s'il existe un chemin (fini) sur G entre ces deux états.

Pour caractériser un ensemble d'états, c'est-à-dire, un sous-ensemble F de E, (par exemple, l'ensemble des configurations initiales), on aura recours à une *formule d'état* ϕ. Celle-ci se résumera le plus souvent à un système d'inégalités linéaires, ou à une disjonction de tels systèmes.

Les *propriétés de sûreté* sont des propriétés fondamentales pour le bon fonctionnement des automates synchronisés. Il peut s'agir :

- de prévenir les situations de blocage (*deadlock*) dans les systèmes «réactifs», en fonctionnement permanent
- d'interdire l'accès simultané à une ressource non partageable — comme un registre mémoire en écriture (c'est le fameux problème de l'exclusion mutuelle)
- de cantonner les états globaux du système aux seuls états sûrs.

Ces propriétés d'invariance stipulent qu'à partir d'une configuration initiale ϕ_0 autorisée, des états «interdits» ne doivent jamais être atteints, ou encore, que tous les états accessibles doivent vérifier une propriété ϕ, où ϕ est une formule d'état.

En logique temporelle linéaire, ceci revient à prouver que : $\phi_0 \Rightarrow \Box\, \phi$
ou, de façon équivalente, que : $\phi_0 \Rightarrow \neg\, (\Diamond\, \neg \phi\,)$.

La vérification d'une propriété de sûreté se ramène en pratique au calcul itératif de l'ensemble des états accessibles, lequel ne doit pas contenir d'état interdit. C'est ce que résume la formule :

$$\text{Post}^*(\mathit{Source}) \cap \mathit{Cible} = \varnothing$$
où $\mathit{Source} = \text{Sol}(\phi_0)$ et $\mathit{Cible} = \text{Sol}(\neg\phi)$.

3 Réseaux de Petri

Les réseaux de Petri (1962) servent à modéliser des systèmes parallèles où les processus peuvent être relativement nombreux, avec des exigences de synchronisation strictes. Ils sont principalement utilisés pour la spécification et la vérification des automatismes, en contrôle-commande, ou pour les protocoles, en télécommunications.

3.1 Définition des places et transitions

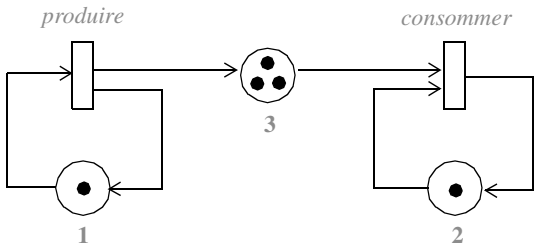
Dans le formalisme des réseaux de Petri, un système est composé d'un nombre fini de *places* (ou sites), symbolisées par des cercles. Ces places sont occupées par des *jetons*. Certaines places servent au flot de données, d'autres au flot de contrôle. Dans le premier cas, les jetons y sont interprétés comme des ressources (produites ou consommées) et leur nombre (positif ou nul) n'a pas de raison d'être borné.

Un état global du système est un *marquage*, c'est-à-dire une distribution de jetons aux différentes places.

Les *transitions*, symbolisées par des barres, correspondent à des actions modifiant l'état du système. Ces actions (génériques) sont en nombre fini. Le réseau indique les conditions pour lesquelles une action est possible ainsi que les conséquences de cette action. Chaque transition relie un ensemble de places en entrée à un ensemble de places en sortie. La possibilité d'effectuer une action dépend ainsi de conditions locales à certains sites.

Pour qu'une transition soit *franchissable*, chacune des places situées en entrée doit contenir au moins un jeton — ou plus si un *poids* (entier positif) est expressément spécifié. Le franchissement d'une transition a pour effet de décrémenter le nombre de jetons dans les places en entrée et de l'incrémenter dans les places en sortie. Suite à cette opération, le nombre total de jetons peut varier.

Exemple Le réseau suivant à trois places et deux transitions représente un producteur et un consommateur. Le producteur (1) et le consommateur (2) disposent chacun d'un jeton de contrôle. Le site (3), représenté ici avec trois jetons, sert à stocker la production en attente de consommation. L'action *produire* ajoute un jeton au site (3). L'action *consommer* lui en retire un. Elle n'est possible que si un jeton est déjà présent en (3).



Dans les réseaux avec *arcs inhibiteurs*, la règle de franchissement est modifiée pour interdire une transition lorsqu'une place tierce contient un ou plusieurs jetons — ce qui revient à conditionner la transition par un test à zéro.

L'arc inhibiteur qui part de la place p pour contrôler la transition est symbolisé par une pointe circulaire ——○ . La transition est infranchissable dès que le nombre de jetons en place p est, si l'on généralise cette notion, supérieur au poids mentionné sur l'arc inhibiteur. Si ce poids vaut 1, il ne doit y avoir aucun jeton en place p pour pouvoir franchir la transition.

Les arcs inhibiteurs confèrent aux réseaux de Pétri la puissance des machines de Turing.

Formellement :

Définition Un *réseau de Petri* est un quadruplet $R = (P, T, \alpha, \beta)$ où P est un ensemble fini de places, T un ensemble fini de transitions, et α et β deux fonctions :

- α, définie sur $P \times T$ et à valeurs dans $\mathbb{Z}$, indique le poids $|\alpha(p, t)|$ de l'arc reliant la place d'entrée p à la transition t [1].
  Une valeur négative (typiquement –1) indique que l'arc est inhibiteur.
- β, définie sur $T \times P$ et à valeurs dans $\mathbb{N}$, indique le poids $\beta(t, p)$ de l'arc reliant la transition t à la place de sortie p.

Par défaut, le poids d'un arc vaut 1 lorsqu'il est non nul.

Soit $\boldsymbol{x}$ le vecteur de $\mathbb{N}^{|P|}$ représentant l'état du système à un instant donné. Pour que, partant de ce marquage, une transition t soit franchissable, $\boldsymbol{x}$ doit vérifier un ensemble de conditions concernant les places p situées en entrée :

$$\forall p,\ 1 \le p \le |P| \qquad \alpha(p, t) \neq 0 \quad \Rightarrow \quad \begin{array}{l} x_p \ge \alpha(p, t) \ \text{ si } \alpha(p, t) > 0, \\ x_p < |\alpha(p, t)| \ \text{ si } \alpha(p, t) < 0 \end{array} \qquad (1)$$

Exemple Si $\alpha(p, t) = -2$, l'arc de p vers t est un arc inhibiteur généralisé et la pré-condition associée est $x_p < 2$, soit $x_p \le 1$.

Le franchissement de la transition t a pour effet de transformer $\boldsymbol{x}$ en un nouveau marquage $\boldsymbol{x'} \in \mathbb{N}^{|P|}$:

$$\forall p,\ 1 \le p \le |P| \quad \begin{array}{ll} x_p' = x_p - \alpha(p, t) + \beta(t, p) & \text{si } \alpha(p, t) > 0 \\ x_p' = x_p + \beta(t, p) & \text{sinon} \end{array} \qquad (2)$$

Exemple Pour la transition produire, $\alpha(1, t) = \beta(t, 1) = 1$, donc $x_1' = x_1$.

Définition Un *réseau de Petri marqué* est un couple $(R, \boldsymbol{x}_0)$ où $\boldsymbol{x}_0$ est un marquage du réseau de Petri R, appelé *marquage initial*.

---

1. Avec la convention qu'un poids nul équivaut à une absence d'arc.

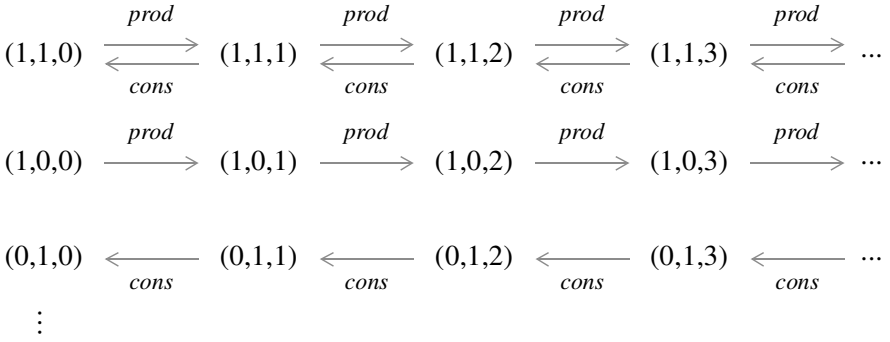
Remarque En utilisant la formulation avec des inégalités larges ($x_p \leq |\alpha(p, t)| - 1$ lorsque $\alpha(p, t) < 0$), les pré-conditions (1) conduisent à des encadrements des variables d'états x_i par des intervalles fermés de $\mathbb{R}^+$.
Les équations (2) montrent que $\boldsymbol{x'} = \boldsymbol{x} + \boldsymbol{c}_t$ où $\boldsymbol{c}_t$ est un vecteur constant de $\mathbb{Z}^{|P|}$ caractérisant l'effet sur chaque place du franchissement de la transition t.
Un système formé uniquement de contraintes (1) et (2) est donc 1-monotone.
Avec deux variables par égalité, c'est un cas particulier de problème de tensions.

Par récurrence, on prouve que :

Propriété *Tout état accessible à partir de $\boldsymbol{x}_0$ est nécessairement de la forme $\boldsymbol{x'} = \boldsymbol{x}_0 + \boldsymbol{c}$ où $\boldsymbol{c}$ est une combinaison linéaire à coefficients entiers positifs des contributions des différentes transitions : $\boldsymbol{c} = \sum_{t \in T} \lambda_t \boldsymbol{c}_t$ avec $\lambda_t \in \mathbb{N}$.*

Un réseau de Petri a une structure finie de graphe mais le graphe des états qui lui correspond est en général infini.

Exemple En développant le graphe du système producteur-consommateur, on obtient :



En partant d'une configuration initiale avec un jeton de contrôle en 1 et 2, le graphe des marquages accessibles se restreint au sous-graphe (infini) construit sur (1,1,n), avec $n \geq 0$ (première composante connexe figurée ci-dessus).

3.2 Programme à compteurs équivalent

Un réseau de Petri exprime sous forme déclarative (et non déterministe) les relations devant exister entre données, avant et après tout changement d'état. Ses places peuvent être assimilées à des compteurs d'entiers que l'automate manipule à l'aide d'un jeu d'instructions arithmétiques simples (incrément, décrément de variable, comparaison à une constante).

Il n'est donc pas étonnant que l'on puisse établir directement une correspondance sémantique entre un réseau de Petri R et un programme logique p — avec contraintes arithmétiques linéaires — représentant un automate à compteurs. Dans cette traduction, chaque compteur devient une dimension numérique et chaque transition une règle récursive, de la forme *p(successeur) si p(prédécesseur)* :

$$p(\boldsymbol{x'}) \rightarrow p(\boldsymbol{x}) \ , \ \{\mathbf{A}(\boldsymbol{x}, \boldsymbol{x'}) \geq \boldsymbol{b}, (\boldsymbol{x}, \boldsymbol{x'}) \geq 0\} \ ;$$

Ces règles se lisent de la façon suivante[1] : un marquage $\boldsymbol{x'}$ est accessible s'il est le résultat d'une transition effectuée à partir d'un marquage $\boldsymbol{x}$ lui-même accessible. Nous adoptons ici des notations à la Prolog III (Colmerauer, 1990), en mettant entre accolades les systèmes de contraintes, pour bien montrer que chaque règle ne contient que deux littéraux.

Stricto sensu, nous aurions dû écrire : $(\boldsymbol{x}, \boldsymbol{x'}) \in \mathbb{N}^{2|P|}$. Les contraintes portent en effet sur des marquages entiers. Mais les contraintes (1) et (2) s'avérant des contraintes de tensions, *chaque transition préserve l'intégrité des variables* (du fait de la totale unimodularité des matrices **A)**.

La relaxation $(\boldsymbol{x}, \boldsymbol{x'}) \geq 0$ sera donc justifiée si l'on est assuré de partir de marquages entiers (ou d'aboutir à des marquages entiers). On pourra ainsi résoudre plus efficacement ces contraintes en les interprétant sur les réels.

Dans toute la suite, nous ferons l'hypothèse que les régions sources (ou cibles) sont décrites par des conjonctions ou disjonctions de contraintes de tensions[2] *de la forme x Op k ou y – x Op k , avec k constante entière et Op* $\in \{\leq, \geq, =\}$.

Exemple La transition *produire* dans le système producteur-consommateur se traduit par :

```
p(<x1,x2,x'3>) -> p(<x1,x2,x3>) , {x'3 = x3+1, x1>=1, x2>=0, x3>=0} ;
```

On reconnaît entre accolades un système de contraintes de précédences ou de tensions. Implicitement, il faut lui ajouter les équations de tête : `x'1 = x1, x'2 = x2.`

Les contraintes attachées à une règle regroupent donc, sans les discerner, les conditions de franchissement de la transition et les équations de changement d'état. Comme ces contraintes ne présentent pas de directionnalité, on peut tout aussi bien calculer le prédécesseur d'un état à partir de son successeur que le successeur à partir de son prédécesseur. Cette symétrie peut être mise à profit pour faire fonctionner le moteur de règles en chaînage avant (simulation) comme en chaînage arrière (rétro-simulation).

Au total, il y a autant de règles pour p qu'il y a de transitions dans R. On garde ainsi une représentation finie d'un automate potentiellement infini.

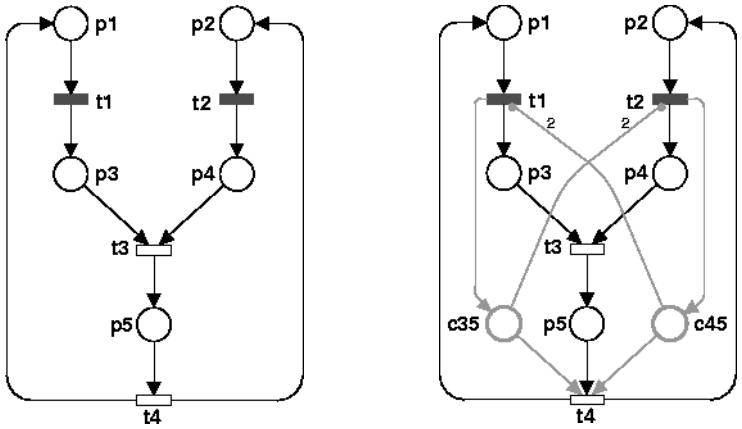
S'il l'on ajoute à ce programme récursif le cas de base :

$$p(\boldsymbol{x}_0) \rightarrow ;$$

exprimant qu'un marquage initial $\boldsymbol{x}_0$ est un état accessible du réseau[3], alors p définit une *relation de fermeture* qui est celle de l'accessibilité[4]sur $(R, \boldsymbol{x}_0)$.

1. Le sens de la flèche Prolog III est inversé par rapport à la transition suivant l'arc $\boldsymbol{x} \rightarrow \boldsymbol{x'}$.
2. De façon courante, les inégalités à une variable suffiront à les décrire. La généralisation sera utile pour les requêtes paramétrées.
3. Ou plus généralement, si l'on ajoute un ensemble de règles $p(\boldsymbol{x}_0) \rightarrow \{\mathbf{A}\boldsymbol{x}_0 \geq \boldsymbol{b}, \boldsymbol{x}_0 \geq 0\}$ définissant les marquages initiaux.
4. En anglais, *reachability* (cf. nom donné à la relation p dans le code de l'exemple).

Exemple La figure suivante (www.lsv.ens-cachan.fr/~phs/MARS) montre, à gauche, un réseau de Petri où le nombe de jetons en place 5 n'est pas contrôlé. A droite, le réseau après correction : deux places de contrôle et deux arcs inhibiteurs sont rajoutés. Initialement, seules les places 1 et 2 disposent de jetons. On doit garantir qu'il n'y a jamais plus de deux jetons en place 5.



Il est facile d'exhiber un contre-exemple pour le réseau de gauche : la suite d'actions $t_1t_2t_3t_1t_2t_3t_1t_2t_3$ conduit ainsi du marquage initial (3,3,0,0,0) au marquage (0,0,0,0,3).

Le réseau contrôlé R se traduit par la base de règles :

```
reachable(<x1-1, x2, x3+1, x4, x5, x35+1, x45)      ->        % t1
    reachable(<x1, x2, x3, x4, x5, x35, x45>) ,
    {x1 >=1, x2 >= 0, x3 >= 0, x4 >= 0, x5 >= 0,
     x35 >= 0, 0 <= x45 <= 1} ;

reachable(<x1, x2-1, x3, x4+1, x5, x35, x45+1>)     ->        % t2
    reachable(<x1, x2, x3, x4, x5, x35, x45>) ,
    {x1 >= 0, x2 >= 1, x3 >= 0, x4 >= 0, x5 >= 0,
     0 <= x35 <= 1, x45 >= 0} ;

reachable(<x1, x2, x3-1, x4-1, x5+1, x35, x45>)     ->        % t3
    reachable(<x1, x2, x3, x4, x5, x35, x45>) ,
    {x1 >= 0, x2 >= 0, x3 >= 1, x4 >= 1, x5 >= 0,
     x35 >= 0, x45 >= 0} ;

reachable(<x1+1, x2+1, x3, x4, x5-1, x35-1, x45-1>) ->        % t4
    reachable(<x1, x2, x3, x4, x5, x35, x45>) ,
    {x1 >= 0, x2 >= 0, x3 >= 0, x4 >= 0, x5 >= 1,
     x35 >= 1, x45 >= 1} ;
```

Les marquages initiaux $\boldsymbol{x}_0$ se traduisent par la base de faits :

```
reachable(<x1, x2, 0, 0, 0, 0, 0>)                  ->        ,
    {x1 >= 0, x2 >= 0} ;
```

Prouver la propriété d'invariance $x_5 \leq 2$ revient à réfuter l'accessibilité sur le réseau de marquages où le nombre de jetons en place 5 est supérieur à 3. La requête correspondante est :

```
->  reachable(<x1, x2, x3, x4, x5, x35, x45>) ,
    {x1 >= 0, x2 >= 0, x3 >= 0, x4 >= 0, x5 >= 3, x35 >= 0, x45 >= 0};
```

Nous démontrerons qu'elle est impossible à satisfaire sur $(R, \boldsymbol{x}_0)$, quel que soit l'état initial $\boldsymbol{x}_0$.

4 Preuves d'accessibilité

4.1 Limitations théoriques

Le problème de l'accessibilité dans un réseau de Petri consiste à caractériser tous les marquages atteignables à partir d'un marquage initial, en vue de la vérification d'une propriéte d'invariance.

D'un point de vue sémantique, l'accessibilité dans un réseau $(R, \boldsymbol{x}_0)$ s'analyse à travers le prédicat p qui traduit cette relation : $p(\boldsymbol{x})$ est vrai si et seulement si $\boldsymbol{x}$ est un marquage accessible.

Mais d'un point de vue algorithmique, déterminer si un marquage est accessible à partir d'un marquage initial est un problème *indécidable* pour les réseaux de Petri avec arcs inhibiteurs[1].

En pratique, cela ne veut pas dire que des preuves automatiques ne pourront être faites, mais que nous ne disposerons au mieux que de *semi-algorithmes*, sans garantie de terminaison. Le programme p ne pourra donc y déroger.

4.2 Incomplétude de la stratégie opérationnelle de Prolog

Même lorsque l'ensemble des marquages est fini (et donc l'accessibilité décidable), un programme Prolog aussi simple que p peut être sujet à des problèmes de terminaison.

Théoriquement, pour savoir si une région cible est accessible, il suffit de lancer la requête :

$$\rightarrow p(\boldsymbol{x}_c)\ ,\ \mathrm{S}_c\ ;$$

où $\boldsymbol{x}_c$ est un vecteur de variables numériques dont on s'intéresse à connaître les valeurs solutions et S_c un système de contraintes auquel $\boldsymbol{x}_c$ doit satisfaire pour appartenir à l'ensemble des marquages cibles.

Posons $\boldsymbol{x}_c = \boldsymbol{x}_n$ et $\mathrm{S} = \mathrm{S}_n$. Le triplet $\langle \boldsymbol{x}_n, p(\boldsymbol{x}_n), \mathrm{S}_n \rangle$ représente l'état de départ de la machine. Etant donnée la forme très simple du programme p, le mécanisme de son interprétation par une machine abstraite — non-déterministe — se résume à trois formules :

$$(1) \qquad \langle \boldsymbol{x}_n, p(\boldsymbol{x}_m), \mathrm{S}_m \rangle$$

$$(2) \qquad p(\boldsymbol{x}_m) \rightarrow p(\boldsymbol{x}_{m\text{-}1})\ ,\ \mathrm{S}_{m,m\text{-}1}$$

$$(3) \qquad \langle \boldsymbol{x}_n, p(\boldsymbol{x}_{m\text{-}1}), \mathrm{S}_m \cup \mathrm{S}_{m,m\text{-}1} \rangle$$

(1) représente l'état de la machine à un instant donné

(2) représente le choix d'une règle du programme p pour le changement d'état (on a renommé ici les variables de tête)

(3) représente le nouvel état de la machine après application de la règle (2).

1. L'accessibilité entre deux états donnés est décidable dans les réseaux sans arcs inhibiteurs (Mayr 1984), mais la complexité du problème est inconnue.

Le passage dans ce nouvel état n'est possible que si l'union des systèmes de contraintes est un système satisfaisable.

Seule la sélection en (2) et l'application en (3) d'une règle de base :

$$p(\boldsymbol{x}_0) \rightarrow \mathrm{S}_0$$

permet de mettre un terme à ces réécritures. La machine fournit alors comme réponse un système de contraintes simplifié, équivalent pour $\boldsymbol{x}_n$ à l'ensemble des inégalités accumulées le long du chemin menant à $\boldsymbol{x}_0$, à savoir :

$$\mathrm{S}_n \cup \mathrm{S}_{n,n\text{-}1} \cup \mathrm{S}_{n\text{-}1,n\text{-}2} \cup \ldots \cup \mathrm{S}_{1,0} \cup \mathrm{S}_0.$$

Cette simplification finale correspond à une élimination de Fourier par projection sur $\boldsymbol{x}_n$ (ensemble des variables de la question). Elle peut être effectuée à chaque inférence.

Un tel résultat apporte la preuve qu'une partie de la région cible est accessible à partir de la région source — pour la preuve du contraire, il faut que toutes les tentatives de démonstration échouent.

Malheureusement, une machine Prolog qui ne mémorise pas les états par où elle passe peut aisément s'enfermer dans un circuit et y boucler indéfiniment.

Exemple Si on lance telle quelle la requête `reachable/1` dans l'exemple précédent, Prolog III s'enferre en appliquant répétitivement la règle de transition n°2.

Il faut donc, d'abord, rendre la stratégie de calcul équitable, afin que la recherche soit exhaustive (et la preuve complète). En effet, une recherche standard en profondeur d'abord ne garantit pas que tous les états accessibles soient trouvés, contrairement à une *recherche en largeur d'abord*.

Il convient ensuite d'éviter certaines situations de non terminaison, notamment, quand le graphe des états accessibles est fini. S'il contient des circuits, un mécanisme de *tabulation* peut empêcher le développement d'un arbre de recherche infini.

Recherche en largeur d'abord et tabulation se combinent classiquement dans le cadre d'une évaluation ascendante. Techniquement, pour le programme *p*, il n'y aura guère de différence entre un mode *bottom-up* et un mode *top-down*.

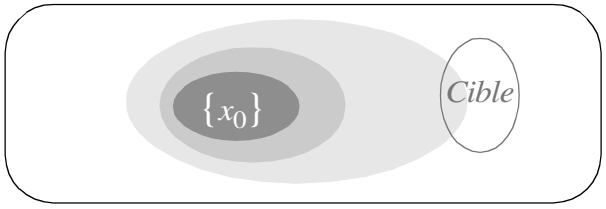
4.3 Calcul itératif des états accessibles

Une façon mécanique de procéder aux calculs est suggérée par la définition du plus petit point fixe de l'opérateur de conséquence immédiate associé au programme *p* (Van Emden & Kowalski, 1976).

Définition L'ensemble des *conséquences immédiates* d'un état $\boldsymbol{x}$ correspond à l'ensemble de ses *successeurs* Post($\{\boldsymbol{x}\}$) via la relation de transition *T*.

En itérant l'application de cet opérateur, on obtient l'ensemble des conséquences logiques de *p*, soit, tout naturellement, celui des états accessibles.

On calcule tout d'abord les successeurs immédiats E_1 du marquage initial $\boldsymbol{x}_0$. Puis les successeurs E_2 de ces successeurs. Et ainsi de suite, jusqu'à ce que l'ensemble $\bigcup_{i=0} E_i$, où $E_0 = \{\boldsymbol{x}_0\}$, ne grossisse plus. S'il en est ainsi, un point fixe est atteint (pour $n \geq 0$) : c'est l'ensemble d'états Post*(E_0) *recherché.*



Pour des automates d'états finis, ce calcul itératif fonde les techniques de *Model Checking*. L'espace de recherche étant alors fini, il se termine toujours.

Pour des automates d'états infinis comme les automates à compteurs et les réseaux de Petri, la pratique montre que ce calcul peut se stabiliser, même quand la région de départ est elle-même infinie (Bérard & Fribourg, 1999).

Exemple Pour le réseau producteur-consommateur, le calcul de Post*($\{(1,1,0)\}$) ne s'arrête jamais, alors que le calcul de Post*($\{(1, 1, n) \mid n \geq 0\}$) atteint son point fixe dès la première itération.

Le calcul itératif des états accessibles ne nécessite donc pas la construction du graphe des états accessibles. Il se suffit d'une représentation symbolique finie de l'ensemble (potentiellement infini) de ces états.

Notre propos est maintenant d'aborder la question de l'implantation effective de ce calcul en programmation logique avec contraintes.

Dans un premier temps, nous allons décrire comment interpréter les règles du programme *p* pour effectuer un calcul classique, de type simulation, sur un réseau où le nombre de marquages accessibles est fini. Puis nous passerons à l'aspect essentiel que constitue, pour les réseaux non bornés, le calcul symbolique des marquages accessibles et la manipulation d'ensembles d'états à travers les contraintes de tensions.

5 Gestion des règles

Modifier la stratégie de recherche en Prolog s'effectue par le biais d'une méta-interprétation des règles.

5.1 Moteur d'interprétation

Pour mémoriser les états visités avec une technique de tabulation simple, on constitue une base de faits dynamique, `fact/2`, que l'on initialise avec la description du marquage initial $\boldsymbol{x}_0$ (correspondant au précédent cas de base) :

```
fact(0, reachable(X0)) -> {X0 = ...} ;
```

Suivant le principe de l'évaluation ascendante, on calcule à chaque itération *i* l'ensemble des successeurs des marquages accessibles en moins de *i*–1 coups. Chaque itération correspond ainsi à une double boucle, sur la base de faits et sur la base de règles :

- sélectionner un fait : $p(\boldsymbol{x})$, S ;
- sélectionner une règle : $p(\boldsymbol{y}) \rightarrow p(\boldsymbol{z})$, T ;
- déduire, par *Modus Ponens*, un nouveau fait : $p(\boldsymbol{y})$, $\{\boldsymbol{x} = \boldsymbol{z}\} \cup S \cup T$.

De ces déductions, on ne retiendra — en les assertant dans la base — que les marquages qui ne sont pas déjà connus en base. Notons que ces «effets de bord» utilisés par le méta-interprète ne compromettent en aucune manière la nature déclarative du programme source *p* (rebaptisé `reachable/1` dans le code). Leur intérêt sera de simplifier par rapport à $\boldsymbol{y}$ le système de contraintes $\{\boldsymbol{x} = \boldsymbol{z}\} \cup S \cup T$.

Vu la structure particulière des clauses à interpréter (avec deux littéraux, un positif, un négatif), il s'avère inutile de passer en revue toute la base de faits (évaluation *naïve*). Pour produire de nouveaux marquages, il suffit de partir des faits les plus récemments déduits. L'indexation de la base `fact/2` par la date d'itération *i* permet une économie substantielle en recherche.

```
forward_iteration_step(I) ->
    fact(I-1, F)            % fait déduit à l'étape précédente (i–1)
    rule(F', [F])           % instance de clause dont le littéral négatif s'unifie avec F
    conditional_assert(fact(I, F'))
    fail ;                  % force l'énumération exhaustive des faits et règles

forward_iteration_step(_) -> ;


conditional_assert(fact(_, F)) ->
    fact(_, F) !            % ignorer un fait déjà connu
    fail ;

conditional_assert(fact(I, F)) ->
    assert(fact(I, F), []) ;
```

On réitère le processus jusqu'à ne plus trouver de fait nouveau à asserter. Lorsque tel est le cas, on a atteint le plus petit point fixe du programme *p* dont la base de faits engendrée donne alors une expression sous forme énumérée.

Remarque Du fait de la symétrie des règles de transition, il est très simple de passer d'une évaluation ascendante — qui progresse à partir de la région source — pour le calcul des successeurs Post*(*Source*), à une évaluation descendante — qui régresse à partir de la région cible et calcule l'ensemble des états pouvant y mener, pour le calcul des prédécesseurs Pre*(*Cible*). Il suffit d'échanger dans la règle le rôle de l'antécédent et du conséquent :

```
backward_iteration_step(I) ->
    fact(I-1, F)
    rule(F, [F'])
    conditional_assert(fact(I, F'))
    fail ;

backward_iteration_step(_) -> ;
```

5.2 Compilation

Le surcoût de la méta-interprétation dans le calcul du point fixe peut être réduit en remplaçant les appels au prédicat prédéfini `rule/2` [1] par des appels de prédicats définis.

Nous transformerons donc, au titre de pré-traitement, la base de règles définissant le prédicat unaire `reachable` en une base de faits équivalente définissant le prédicat binaire `transition`. L'interêt comparé de ces deux formulations est très bien discuté dans l'ouvrage de Kowalski (1979).

Exemple La transition *produire* dans le système producteur-consommateur se traduit maintenant par :

```
transition(<x1,x2,x3>, <x1,x2,x'3>) -> ,
    {x'3 = x3+1, x1>=1, x2>=0, x3>=0} ;
```

La déclaration du marquage initial qui initialise la base de fait devient :

```
fact(0, X0) -> {description du marquage initial} ;
```

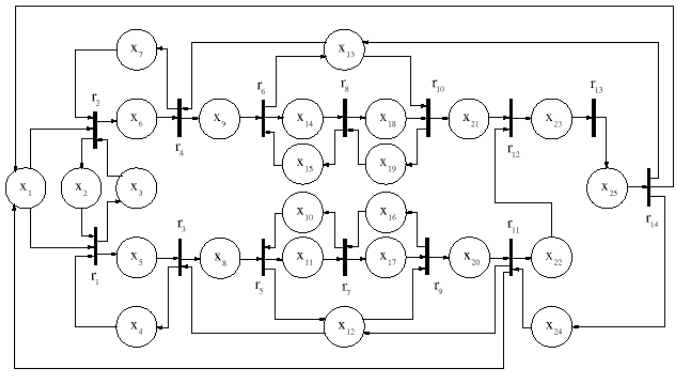
et la double boucle correspondant à l'itération *i* se simplifie en :

```
forward_iteration_step(I) ->
    fact(I-1, M)
    transition(M, M')
    conditional_assert(fact(I, M'))
    fail ;

forward_iteration_step(I) -> ;
```

C'est à peu près tout ce dont on a besoin pour démontrer la présence ou l'absence de blocage dans l'exemple suivant, lorsqu'on part d'un marquage initial connu.

Exemple Ce réseau de Petri, à 25 places et 14 transitions, modélise un atelier flexible. avec quatre machines, deux robots, deux sites tampons de capacité 3 (x_{10} et x_{15}) et une cellule d'assemblage.



1. `rule/2` correspond en Prolog III à la méta-primitive `clause/2` de la norme ISO. L'unification avec la tête et le corps d'une clause remonte (sans qu'il soit besoin de le dire) le système de contraintes attaché. L'assert en Prolog III opère de façon symétrique.

Les marquages initiaux sont définis par $x_1 \geq 1$ et $x_2 = x_4 = x_7 = x_{12} = x_{13} = x_{16} = x_{19} = x_{24} = 1$, $x_{10} = x_{15} = 3$, $x_3 = x_5 = x_6 = x_8 = x_9 = x_{11} = x_{14} = x_{17} = x_{18} = x_{20} = x_{21} = x_{22} = x_{23} = x_{25} = 0$. Pour un marquage initial satisfaisant à cette formule, il se trouve que l'ensemble des marquages accessibles est fini : le réseau est borné. L'algorithme atteint donc son point fixe.

Un blocage du réseau intervient lorsqu'aucune transition ne peut être franchie, c'est-à-dire, lorsque, pour chaque transition, au moins une place en entrée est sans jeton. Ces situations sont résumées par la formule d'état : $x_2 = x_5 = x_7 = x_9 = x_{10} = x_{12} = x_{13} = x_{16} = x_{19} = x_{20} = x_{22} = x_{23} = x_{25} = 0$. C'est ce qu'indique la clause :

```
deadlock(<_,0,_,_,0,_,0,_,0,0,_,0,0,_,_,0,_,_,0,0,_,0,0,_,0>) -> ;
```

Une fois la base de faits engendrée, on peut l'interroger pour vérifier s'il existe des marquages accessibles correspondant à de telles situations. La conjonction suivante fait office de requête :

```
deadlock(X) fact(_, X) ;
```

Si la recherche Prolog réussit, elle donne, en substitution réponse, un exemple de marquage à la fois accessible et responsable de blocage. Si elle échoue, la réponse négative de l'interprète est une preuve d'absence de situation de blocage pour le réseau marqué.

Le tableau suivant récapitule les résultats obtenus pour le réseau en faisant varier de 1 à 12 le nombre de jetons initialement placés en 1.

Valeur de x1	Nombre d'iterations	Taille de la base de faits	Absence de blocage
1	14	14	oui
2	19	74	oui
3	23	204	oui
4	27	464	oui
5	30	784	oui
6	33	1168	oui
7	36	1528	oui
8	39	1858	oui
9	40	2034	non
10	41	2130	non
11	41	2142	non
12	41	2144	non

On observe une absence de situation de blocage pour $x_1 \leq 8$, mais pas au delà. Le compte de jetons pour la place 1 n'étant pas borné, si l'on veut une réponse valable pour toute valeur initiale de x_1, il faut raisonner symboliquement, avec toute la puissance d'expression et de résolution des contraintes linéaires.

6 Gestion des contraintes

Partant d'un marquage initial donné, l'application d'une séquence de transitions aboutit à une simulation classique de l'évolution du réseau. Or il arrive très fréquemment qu'une même séquence de transitions s'applique à d'autres états initiaux : tous ceux vérifiant les mêmes conditions pour le franchissement de la première transition et dont les successeurs, à chaque transition franchie, vérifient également les conditions de franchissement des transitions suivantes. En l'absence d'arcs inhibiteurs notamment, si $\boldsymbol{x}$ est un marquage accessible à partir de $\boldsymbol{x}_0$, alors la même séquence de transitions permet, à partir d'un marquage $\boldsymbol{y}_0 \geq \boldsymbol{x}_0$, d'atteindre un marquage $\boldsymbol{y} \geq \boldsymbol{x}$.

On peut donc factoriser ces suivis dans une «simulation symbolique» qui suit un ensemble d'états (défini par des contraintes) plutôt qu'un état unique (défini par des valeurs).

6.1 Calcul symbolique des états accessibles

Informellement, il suffit de considérer l'état courant comme un fait :

$$p(\boldsymbol{x}) \text{ , S}$$

où le système de contraintes linéaires attaché à $\boldsymbol{x}$ n'est plus qu'une *spécification partielle* du n-uplet $\boldsymbol{x}$. On manipule alors, dans l'abstraction, un ensemble, Sol(S), qui n'est plus un singleton mais un polyèdre à l'intérieur non vide.

Franchir une transition à partir de l'état courant $\boldsymbol{x}$ revient à appliquer (en sens inverse du chaînage arrière) une règle de la forme :

$$p(\boldsymbol{x'}) \rightarrow p(\boldsymbol{x}) \text{ , T}$$

Lorsque la transition est effective, l'état successeur décrit un nouvel ensemble, Sol(S'), défini par le fait :

$$p(\boldsymbol{x'}) \text{ , S'}$$

où S' est le résultat de l'élimination de $\boldsymbol{x}$ du système de contraintes S $\cup$ T.

Les faits mémorisés dans le calcul itératif des états successeurs correspondent ainsi aux définitions de Sol(S_0), ..., Sol(S), Sol(S'), etc. La base représente l'union (finie) de ces ensembles (potentiellement infinis) d'états accessibles.

La seule retouche à apporter à l'algorithme d'évaluation ascendante a trait à l'enregistrement d'un nouveau fait. Vérifier qu'il n'est pas déjà répertorié revient à tester si l'ensemble des états qu'il décrit n'est pas inclus dans l'union des ensembles constituant la base. Pour permettre en pratique la terminaison de l'algorithme, le minimum à vérifier sera que ce nouvel ensemble ne soit inclus dans aucun des ensembles précédemment calculés, autrement dit, que le nouveau fait ne soit pas impliqué par un fait plus ancien.

Cette simplification du test d'inclusion (qui n'affecte pas le caractère de preuve des réponses positives ou négatives) peut avoir pour effet de retarder, indéfiniment parfois, la détection du point fixe Post*(Sol(S_0)).

Mais la procédure de décision pour un test d'implication global (et non local) serait par trop combinatoire[1] — sans pour autant garantir la terminaison de l'algorithme.

La modification du code est donc la suivante :

```
conditional_assert(fact(_, F)) ->
    fact(_, E)
    contains(E, F) !      % teste si F ⊂ E (remplace le test d'égalité entre les valeurs E et F)
    fail ;

conditional_assert(fact(I, F)) ->
    assert(fact(I, F), []) ;
```

A présent défini, le schéma général de calcul fait ressortir trois opérations fondamentales sur les inégalités linéaires et les polyèdres qu'elles définissent :

1. le *test de non vacuité* dans $\mathbb{N}^{2|P|}$ de l'ensemble Sol(S ∪ T)
   — utilisé lors de la sélection d'une règle
2. la *projection* d'un tel ensemble sur le sous-espace $\mathbb{N}^{|P|}$
   — normalisation effectuée au moment de l'enregistrement d'un fait
3. le *test d'inclusion* entre sous-ensembles de $\mathbb{N}^{|P|}$
   — servant à la terminaison.

Leur implantation à l'aide des outils généralistes de la programmation par contraintes est détaillée et justifiée dans ce qui suit.

6.2 Programmation en Prolog III

Tout langage du type de Prolog III dispose d'un solveur de contraintes linéaires complet qui effectue naturellement les deux premières opérations.

En effet, si S désigne un système de contraintes de tensions, les systèmes T apportés par une règle de transition étant toujours formés de contraintes de tensions, le système S ∪ T reste un système de contraintes de tensions. Le théorème d'intégrité nous assure dans ce cas que la non vacuité de l'ensemble de ses solutions sur $\mathbb{N}^{2|P|}$ équivaut à la non vacuité de l'ensemble de ses solutions sur $\mathbb{R}^{+2|P|}$ — ce que garantit le test de satisfaisabilité de S ∪ T.

D'autre part, pour ces systèmes de contraintes, l'élimination de Fourier de variables entières est licite. L'élimination de $\boldsymbol{x}$ dans S ∪ T est également automatique à partir du moment où le fait asserté ne porte que sur $\boldsymbol{x'}$. Elle aboutit[2] à nouveau à un système de contraintes de tensions qui définit la projection sur $\boldsymbol{x'}$: $\text{Proj}(\text{Sol}(S \cup T) \cap \mathbb{N}^{2|P|}, \boldsymbol{x}) = \text{Proj}(\text{Sol}(S \cup T), \boldsymbol{x}) \cap \mathbb{N}^{|P|}$.

1. Ce problème est en fait NP-complet.
2. La simplification des systèmes de contraintes effectuée par Prolog III n'est pas aussi poussée puisqu'une partie des variables «indésirables» n'est pas éliminée (c'est en particulier le cas des variables d'écart, utilisées dans la forme résolue du simplexe). Cependant, le système simplifié décrit toujours le même polyèdre (qui est ici l'enveloppe convexe des solutions entières en $\boldsymbol{x'}$).

Enfin, tous les faits initiaux étant par hypothèse définis par des contraintes de tensions, ce trait reste une caractéristique des faits enregistrés dans la base. Il en résulte, comme prévu, que l'omission des contraintes d'intégrité n'affecte pas la preuve d'accessiblité : *l'accessibilité d'un polyèdre entraîne celle de tous ses points entiers.*

Le test d'inclusion `contains/2` est en définitive la seule opération qui ne soit pas native en Prolog III. Mais il peut être méta-programmé [1]:

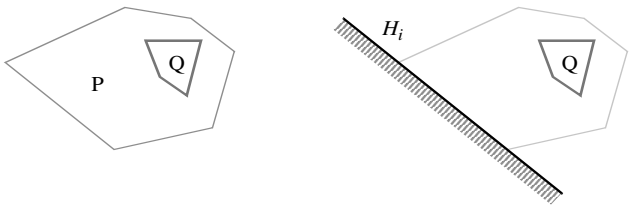
- Tester l'inclusion de deux polyèdres en nombres entiers équivaut à tester l'inclusion de leurs enveloppes polyédriques.
- Prouver qu'un polyèdre Q = Sol(T) est inclus dans un autre polyèdre P = Sol(S) revient à prouver que chaque inégalité de S est conséquence de T.
- Or, d'après la version affine du lemme de Farkas (théorème II.4), une inégalité $\boldsymbol{cx} \geq d$ est conséquence d'un système $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ si et seulement si le système $\boldsymbol{y}\mathbf{A} = \boldsymbol{c}$, $\boldsymbol{y} \geq 0$, $\boldsymbol{yb} \geq d$ est satisfaisable.

Le test d'inclusion se ramène ainsi à la résolution d'un système linéaire en Prolog III.

Remarque Une méthode géométrique plus directe pour le test d'inclusion est la suivante. Supposons, pour simplifier, que le polyèdre P de $\mathbb{R}^{+|P|}$ soit de dimension[2] «pleine» $|P|$. P se décrit alors comme une intersection finie de demi-espaces $(D_i)_{i \in I}$. Dans ces conditions, tester si Q est inclus dans P équivaut à vérifier que pour tout $i \in$ I, $Q \subset D_i$, ou de façon équivalente, que $Q \cap \overline{D}_i = \varnothing$.

Chaque demi-espace étant défini par une inégalité, $D_i = \{\boldsymbol{x} \in \mathbb{R}^{+|P|} \mid \boldsymbol{a}_i\boldsymbol{x} \leq b_i\}$, le complémentaire de D_i est le demi-espace $\overline{D}_i = \{\boldsymbol{x} \in \mathbb{R}^{+|P|} \mid \boldsymbol{a}_i\boldsymbol{x} > b_i\}$ dont la définition s'obtient en inversant le signe d'inégalité et en le rendant strict[3].

Cette petite manipulation syntaxique exige de passer par la syntaxe abstraite de Prolog III pour obtenir une expression sous forme d'arbre de la contrainte définissant D_i. Une fois l'expression de l'inégalité complémentaire obtenue (celle-ci peut être compilée), il suffit de revenir au niveau de l'interprétation concrète pour tester la satisfaisabilité des conditions $T \cup \{\boldsymbol{a}_i\boldsymbol{x} > b_i\}$.



1. Ce test est prédéfini dans la bibliothèque CLPQ fournie avec Sicstus Prolog — elle-même méta-programmée (Holzbaur).
2. Cette dimension est définie par le rang du système d'équations linéaires (Gauss), présent, aux côtés du système d'inéquations (Simplexe), dans la forme normale de Prolog III.
3. On peut en fait conserver une inégalité large, puisqu'on ne s'intéresse ici qu'aux solutions entières. Il suffit de poser : $\overline{E}_i = \{\boldsymbol{x} \in \mathbb{R}^{+|P|} \mid \boldsymbol{a}_i\boldsymbol{x} \geq b_i + 1\}$.

En comparant Q à chacun des hyperplans supports de P : $H_i = \{\boldsymbol{x} \mid \boldsymbol{a}_i\boldsymbol{x} = b_i\}$, on réduit à nouveau le test d'inclusion de Q dans P à une série de tests d'insatisfaisabilité sur des systèmes de contraintes linéaires.

Exemple Retournons à l'atelier flexible. Nous pouvons maintenant lancer directement le calcul des états accessibles à partir de l'ensemble E_0 des marquages définis par la formule d'état : $x_1 \geq 1$, $x_2 = x_4 = x_7 = x_{12} = x_{13} = x_{16} = x_{19} = x_{24} = 1$, $x_{10} = x_{15} = 3$, $x_3 = x_5 = x_6 = x_8 = x_9 = x_{11} = x_{14} = x_{17} = x_{18} = x_{20} = x_{21} = x_{22} = x_{23} = x_{25} = 0$. L'initialisation de la base de faits se borne à l'enregistrement :

```
fact(0, <x1,1,0,1,0,0,1,0,0,3,0,1,1,0,3,1,0,0,1,0,0,0,0,1,0>) ->
    {x1 >= 0} ;
```

Le point fixe Post*(E_0) étant atteint, la base de faits engendrée renferme tous les états accessibles. Son interrogation donne :

```
deadlock(X) fact(_, X) ;

{X = <z,0,1,1,0,1,0,1,0,0,3,0,0,1,2,0,1,1,0,0,1,0,0,1,0>, z >= 0}
```

La réponse à cette requête montre qu'il existe des (entiers) $z \geq 0$ pour lesquels le marquage correspondant $\boldsymbol{x}$ est un état de blocage accessible à partir d'un marquage initial (ce que l'on savait déjà pour les valeurs 9, 10, 11, 12 de x_1).

Pour obtenir une réponse plus précise, nous devons poser une requête paramétrée. Pour cela, on ajoute une 26ième composante au vecteur d'état $\boldsymbol{x}$ qui reste invariante à travers les changements d'états . Cette composante nous sert ici de pointeur vers la valeur initiale de x_1.

On relance alors la recherche à partir de l'ensemble d'états initiaux :

```
fact(0, <x1,1,0,1,0,0,1,0,0,3,0,1,1,0,3,1,0,0,1,0,0,0,0,1,0,x1>) ->
    {x1 >= 1} ;
```

Puis, on repose la même requête, avec pour argument supplémentaire x_1 :

```
deadlock(X) fact(X.<x1>) ;

{X = <z,0,1,1,0,1,0,1,0,0,3,0,0,1,2,0,1,1,0,0,1,0,0,1,0>, x1 = z + 9,
     z >= 0}
```

On déduit de la réponse obtenue que pour toute valeur entière de x_1 supérieure ou égale à 9, une situation de blocage est atteinte (pour un marquage dans lequel z vaut précisément $x_1 - 9$). Cette réponse étant unique, elle est la preuve de la propriété suivante :

$x_1 \leq 8$ *est une condition nécessaire et suffisante d'absence de blocage.*

On résume ainsi en une évaluation ascendante toutes les simulations possibles sur les valeurs initiales de x_1.

Ces résultats corroborent ceux obtenus par Bérard & Fribourg avec l'outil de vérification symbolique HyTech pour automates hybrides (Henzinger *et al.*)[1].

Outil de vérification	Nombre [a] [b] d'iterations	Taille [c] de la base de faits
HyTech	49	560
Prototype Prolog III	41	2144

a. On obtient, au bout du même nombre d'itérations, le même nombre de faits que lors de la simulation avec $x_1 = 12$. Le remplacement du test d'égalité par un test d'inclusion se paye par un temps de calcul vingt fois plus long en Prolog III.

b. Le plus petit nombre d'itérations obtenu avec le prototype Prolog III provient vraisemblablement d'une différence de stratégie de recherche.

c. L'épuration par HyTech des faits redondants explique en grande partie l'écart de taille entre les bases de faits engendrées (l'algorithme implanté vérifie qu'aucun fait n'est impliqué par un fait plus ancien, mais un fait plus récent peut très bien rendre redondant un fait plus ancien).

6.3 Programmation en Prolog IV

Dans un langage comme Prolog IV, nous disposons du choix entre plusieurs techniques de résolution de contraintes linéaires. Nous faisons ici mention d'une modélisation originale faisant appel au solveur d'intervalles.

Les difficultés viendront du fait que ce dernier, contrairement au solveur linéaire[2], est incomplet.

Tout ce qui a été dit à propos de la relaxation des calculs sur les réels reste ici valable : sous réserve que les inégalités spécifiant les marquages initiaux (ou les marquages cibles) entrent dans la classe des contraintes de tensions, l'analyse de l'accessibilité peut être effectuée sans se soucier des contraintes d'intégrité[3].

6.3.1 Test de non vacuité

Observons en premier lieu que les inconnues étant des quantités positives, tous les systèmes de contraintes rencontrés sont de la forme $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \geq 0$, où $\mathbf{A}$ est une matrice d'incidence de graphe. Ces systèmes sont monotones : s'ils sont solubles, l'ensemble P de leurs solutions admet un plus petit élément $\boldsymbol{x}^*$ (à coordonnées entières). Le théorème IV.4 nous assure qu'il est bien calculé si l'ensemble P est borné. C'est là le seul problème pour le test de non vacuité.

1. www.eecs.berkeley.edu
2. Le solveur linéaire de Prolog IV est, dans sa conception, similaire à celui de Prolog III. Mais son implantation actuelle pèche par une quasi absence de procédure de simplification et un manque de primitives de méta-programmation (assert de règles avec contraintes linéaires ; conversion de contraintes en termes) pour envisager un portage du prototype Prolog III.
3. Il n'y a donc pas lieu d'utiliser dans les règles la contrainte `int/1` de Prolog IV.

Pour justifier la complétude de ce test dans l'approximation des intervalles, nous ferons l'hypothèse que, non pas P, mais le plus petit élément $\boldsymbol{x}^*$ de P est borné : $\boldsymbol{x}^* \in \mathrm{P} \cap \{\boldsymbol{x} \mid \boldsymbol{x} \leq \boldsymbol{d}\}$, avec $\| \boldsymbol{d} \| \ll M$ (limite de précision de la grille). On a, algébriquement, $\boldsymbol{x}^* = \boldsymbol{x}_0 + \sum_{t \in T} \lambda_t \boldsymbol{c}_t$. En pratique :

- le nombre d'itérations qui fait croître les entiers λ_t est limité par la combinatoire de la preuve
- les coefficients des transitions $\boldsymbol{c}_t$ sont tous des entiers très petits
- $\boldsymbol{x}_0$ est un élement de la région de départ dont le minimum est proche de 0.

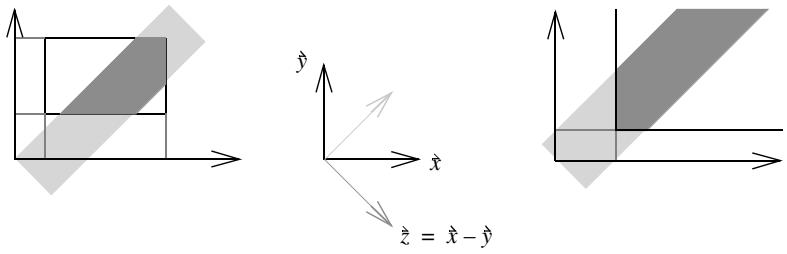
Dans ces conditions, si $\boldsymbol{X}$ désigne le pavé des domaines réduits calculé dans l'espace discrétisé $(\mathrm{R}, \Im_M)$, on a bien : $\boldsymbol{x}^* = \min \boldsymbol{X}$ (dans les cas violant l'hypothèse, on aurait pu avoir pour $\boldsymbol{X}$ un infimum, mais pas de minimum).

Si l'on ajoute donc au filtrage effectué par les intervalles (qui nous assure que le pavé $\boldsymbol{X}$ calculé est non vide) la vérification que tous les intervalles X_i de $\boldsymbol{X}$ sont fermés inférieurement[1] [2], on obtient une procédure de décision complète pour les systèmes de contraintes de tensions envisagés[3].

6.3.2 Opération de projection

Le second problème lié à l'usage des intervalles en Prolog IV tient à l'absence de procédure de simplification — ou à une simplification trop radicale au moment de l'assert ou de l'affichage en sortie des résultats, puisque seules les contraintes de type (c'est-à-dire, les contraintes d'appartenance aux intervalles constituant le pavé des solutions) sont conservées. Mais les polyèdres solution des systèmes de contraintes de tensions ont une géométrie suffisamment simple et régulière pour remédier à ce problème.

Considérons dans le plan (x, y) un polygone défini par des contraintes de tensions : $a \leq x \leq b$, $c \leq y \leq d$, $e \leq x - y \leq g$. Celui-ci peut être décrit comme l'intersection d'une «boîte» $X \times Y$, parallèle aux axes, et d'une «bande» Z, parallèle à la diagonale principale de ce plan.



1. La récupération des bornes inférieures des domaines des variables peut s'effectuer en Prolog IV par l'intermédiaire du prédicat prédéfini `glb/2` (*greatest lower bound*).
2. La distinction entre ouverts et fermés n'étant pas toujours effective dans l'implantation, on peut aussi vérifier que le minimum de chaque variable est un entier.
3. Si l'hypothèse sur le plus petit élément tient, et si, symétriquement lorsqu'il existe, le plus grand élément du polyèdre des tensions ne s'approche jamais de la limite de précision M de la grille, alors le calcul du pavé des domaines réduits est toujours exact.

Introduisons la dimension corrélée $z = x - y$. Par le truchement de cette variable redondante, une contrainte de différence de potentiels entre x et y se résume à encadrement sur z. On peut ainsi décrire un polygone de $\mathbb{R}^2$ par un pavé de $\mathbb{R}^3$.

Dans la modélisation qui en résulte pour les réseaux de Pétri, les variables principales, x et y, sont positives ; les variables de différence, z, ne sont pas signées.

Si l'espace est de dimension n, il faut $n(n-1)/2$ dimensions redondantes pour «normaliser» le système de contraintes. On collecte par ce biais tous les encadrements sur les variables principales (dont on obtient les projections exactes) et sur leurs différences (dont on ne retient que les encadrements les plus serrés). Il en résulte une simplification partielle des systèmes, suffisante pour les opérations de projection.

Définition Soient S un système de contraintes de tensions sur $\mathbb{R}^n$. Le *système normal* associé à S est le système d'inégalités de $\mathbb{R}^{n(n+1)/2}$ obtenu en posant, pour tout couple $i < j$, $z_{ij} = x_j - x_i$ (c'est à dire $x_i + z_{ij} \leq x_j$ et $x_j \leq x_i + z_{ij}$) et en remplaçant dans S chaque contrainte à deux variables $x_j - x_i \leq b_{ij}$ par $z_{ij} \leq b_{ij}$ si $i < j$ et par $-z_{ji} \leq b_{ij}$ si $j < i$.

Du fait des égalités à trois variables définissant les variables redondantes $\boldsymbol{z}$, le système normal n'est plus monotone. Mais vis à vis de la réduction opérée par les intervalles, il se comporte comme si l'on n'avait pas ôté de contraintes au système initial, mais seulement rajouté des contraintes d'égalité et d'inégalité sur les différences.

Les formules de réduction de domaines pour $z = y - x$ se résument en effet à :

$x + z \leq y$	$y \leq x + z$
$\inf(X) + \inf(Z) \leq \inf(Y)$	$\sup(Y) \leq \sup(X) + \sup(Z)$
$\sup(X) \leq \sup(Y) - \inf(Z)$	$\inf(Y) - \sup(Z) \leq \inf(X)$

Si l'on avait initialement $x_j - x_i \leq b_{ij}$, on a nécessairement $\sup(Z_{ij}) \leq b_{ij}$.
On infère donc de $x_j \leq x_j + z$: $\inf(X_j) - b_{ij} \leq \inf(X_i)$ et $\sup(X_j) \leq \sup(X_i) + b_{ij}$ — ce que l'on aurait obtenu par les formules de réduction de cette inégalité.

C'est en ce sens que la formulation en $(\boldsymbol{x}, \boldsymbol{z})$ est véritablement «redondante».

Remarque Contrairement à une représentation de type *Difference Bounds Matrices* (DBM) qui borne les différences entre variables par les plus courtes (ou plus longues) distances entre les sommets correspondants sur le graphe des contraintes, cette représentation normale n'est pas canonique. Mais elle synthétise correctement l'ensemble des contraintes.

Exemple Posons $z_{12} = x_2 - x_1$, $z_{13} = x_3 - x_1$ et $z_{23} = x_3 - x_2$. La résolution par les intervalles du système d'inégalités $-x_1 \leq 0$, $x_3 \leq 8$, $z_{12} \leq 10$, $-z_{12} \leq -4$, $z_{13} \leq 2$, $-z_{23} \leq 2$ rend le pavé : $x_1 \in [0,6]$, $x_2 \in [4,10]$, $x_3 \in [2,8]$, $z_{12} \in [4,10]$, $z_{13} \in [-4,2]$, $z_{23} \in [-2,4]$, alors que formellement $z_{12} = 4$, $z_{13} = 2$, $z_{23} = -2$.

Pour l'aspect projection à proprement parler, de $\mathbb{R}^{+2|P|}$ dans $\mathbb{R}^{+|P|}$, qui correspond à l'élimination de $\boldsymbol{x}$ après le franchissement de la transition menant à $\boldsymbol{x'}$, il nous faut distinguer :

1. le système initial S et les préconditions, T_1, portant sur les variables de $\boldsymbol{x}$
2. le système d'équations de mise à jour des marquages, T_2, liant $\boldsymbol{x}$ à $\boldsymbol{x'}$.

Les premières conditions s'ajoutent aux contraintes définissant l'état courant et font s'intersecter deux polyèdres de $\mathbb{R}^{|P|}$. Les secondes incrémentent le système résultant avec un système toujours satisfaisable puisqu'il introduit autant de nouvelles variables que d'équations.

Avec les variables de différence, les transitions deviennent des relations de $\mathbb{R}^{n(n+1)/2}$ vers $\mathbb{R}^{n(n+1)/2}$. Le système T_2 s'écrit :

$$x'_i = x_i + c_i \qquad \text{pour } i = 1, ..., |P|$$
$$z'_{ij} = z_{ij} + e_{ij} \qquad \text{pour } 1 \le i < j \le |P|$$

avec c_i et e_{ij} constantes entières.

Pour justifier la mise à l'écart, lors de la projection, des variables x_i (resp. z_{ij}), il nous suffit de montrer que les intervalles effectuent dans ce cas une élimination de Gauss :

$$x_i = x'_i - c_i$$
$$z_{ij} = z'_{ij} - e_{ij}$$

et réexpriment, toutes les contraintes sur $\boldsymbol{x}$ (resp. $\boldsymbol{z}$) en fonction de $\boldsymbol{x'}$ (resp. $\boldsymbol{z'}$).

Supposons par exemple $x_i \le b_i$. Alors : $\sup(X_i) \le b_i$ et, par les formules de réduction tirées de $x'_i \le x_i + c_i$, on a : $\sup(X'_i) \le \sup(X_i) + c_i$. On obtient donc $x'_i \le b_i + c_i$ comme si l'on avait effectué une élimination formelle.
Il en va de même pour les différences z_{ij}.

De ce fait, il n'y a pas d'approximation : les bornes des domaines des variables (principales et de différence) renferment toutes les informations utiles. Les encadrements sur ($\boldsymbol{x'}$, $\boldsymbol{z'}$), joints aux égalités définissant $\boldsymbol{z'}$, constituent donc un système normal définissant la projection sur $\boldsymbol{x'}$ du polyèdre des solutions de $S \cup T_1 \cup T_2$.

Les variables d'état et de différence étant assertées avec leurs contraintes de domaine en Prolog IV, l'enregistrement d'un nouveau fait dans la base réalise automatiquement la simplification souhaitée pour l'opération de projection, sans aucune perte d'information.

6.3.3 Test d'inclusion

Le test d'inclusion entre deux polyèdres de $\mathbb{R}^n$ définis par des contraintes de tension va se résumer, dans $\mathbb{R}^{n(n+1)/2}$ à la vérification de l'inclusion de pavés (dont les cotes affichées sont, pour les z_{ij}, approximatives).

Ce test est relativement simple à méta-programmer (les bornes d'un domaine pouvant être récupérées par le biais du prédicat prédéfini `ebounds/3` de Prolog IV). Mais il dépend crucialement du caractère implicitement entier des variables (positives) manipulées.

En effet, si l'on prend le cas le plus simple — à une dimension — de l'inclusion d'un intervalle X dans $[c, +\infty[$ avec c entier, on aura, de manière générale (X désignant par le domaine de la variable x) : $x \in X$ est conséquence de $x \geq c$ si et seulement si $x \in X$ et $x < c$ sont incompatibles.

Or, nous n'avons pas de résultat garantissant la complétude de la résolution d'un tel système. En revanche, si x est une variable entière, la contrainte exprimant la négation de $x \geq c$ est équivalente à $x \leq c - 1$, qui est cette fois une contrainte de tensions. Auquel cas, cette inégalité forme avec les inégalités exprimant l'appartenance de x au domaine X un système de contraintes de tensions que l'on sait résoudre de façon complète sur $\mathbb{R}^+$ avec les intervalles (en testant, sous les mêmes hypothèses, l'existence du minimum).

Exemple Pour vérifier en Prolog IV que la contrainte $z_{12} \leq 4$ est une conséquence du système de l'exemple précédent, on ne devra pas compter sur la seule primitive `ble/3` qui rend un booléen vrai lorsque $z_{12} > 4$ est faux. Ou alors, il faudra la combiner avec la contrainte `int`(z_{12}) pour le test de consécution.

```
systemeS(X1, X2, X3, Z12, Z13, Z23) :-
    Z12 = X2.-.X1, Z13 = X3.-.X1, Z23 = X3.-.X2,
    .-.X1 le 0,
    X3 le 8,
    Z12 le 10, .-.Z12 le -4,
    Z13 le 2,
    .-.Z23 le 2.

>> systemeS(_,_,_,Z12,_,_), gt(Z12, 4).

Z12 ~ oo(4,10).
```

En revanche, le test réussit dans son équivalent pour variables entières :

```
>> systemeS(_,_,_,Z12,_,_), ge(Z12, 5).

false.
```

Le test d'inclusion se ramène, encore une fois, à une série de tests de satisfaisabilité de systèmes de contraintes de tensions à variables positives, pour lesquels nous disposons d'une procédure de décision.

Remarque Un des intérêts de la formulation avec variables de différence est de prévenir, dans les cas de base, l'apparition de phénomènes de convergence lente lors de la routine du test d'inclusion. On est alors amené à calculer l'intersection de demi-espaces dont les hyperplans supports sont parallèles. Si, dans un plan (x, y), les contraintes sur $x - y$ s'avèrent antagonistes, alors le domaine de la variable $z = x - y$ qui sert à les exprimer devient vide et l'insatisfaisabilité est immédiatement détectée.

6.3.4 Application

Nous pouvons maintenant récapituler :

- une double application du théorème sur les systèmes 1-monotones nous a permis de démontrer, sous des hypothèses vraisemblables, la correction et la complétude des tests de non vacuité et d'inclusion réalisés sur des ensembles d'entiers positifs délimités par des contraintes de tensions
- une normalisation des contraintes de tension nous a de plus permis de simplifier les opérations de projection, les réduisant à la définition d'un pavé de $\mathbb{N}^{n(n+1)/2}$, soit à $n^2 + n$ bornes avec $n = |P|$.

Exemple Vérification du contrôleur de réseau MARS défini en début de chapitre.

On rappelle qu'on ne doit pas atteindre un marquage cible dans lequel il y a strictement plus de 2 jetons en place 5 à partir d'un marquage source où seules les places 1 et 2 disposent de jetons. Prouver cette propriété peut s'effectuer en chaînage arrière, en calculant Pre*(*Cible*) ∩ *Source*.

Les spécifications sont les suivantes (pour plus de lisibilité, nous les donnons en Prolog III).

Pour le réseau non contrôlé :

- définition des marquages initiaux (ensemble *Source*) :

```
init(<x1, x2, 0, 0, 0>) -> {x1 >= 0, x2 >= 0} ;
```

- initialisation de la base avec la définition des états interdits (*Cible*) :

```
fact(0, <x1, x2, x3, x4, x5>) ->
    {x1 >= 0, x2 >= 0, x3 >= 0, x4 >= 0, x5 >= 3} ;
```

Interrogation de la base de faits engendrée :

```
> init(X) fact(_,X) ;

    {X = <x1,x2,0,0,0>,
     u = x1-3, v = x2-3, u >= 0, v >= 0,     % x1 >= 3 , x2 >= 3
     w = x1-1, w >= 0, x1 >= 0, x2 >= 0}     % inégalités redondantes
```

Tous les marquages initiaux ayant plus de 3 jetons en place 1 et en place 2 peuvent donc conduire, dans le réseau non contrôlé, à un état interdit.

Prolog IV donne une réponse équivalente :

```
X ~ [ge(3),ge(3),0,0,0]
```

Plus précisément, en posant $z_{ij} = x_i - x_j$ et $x_0 = 0$, on obtient les encadrements :

z	1	2	3	4	5
0	[3,+∞[	[3,+∞[	0	0	0
1		]–∞,+∞[	[3,+∞[	[3,+∞[	[3,+∞[
2			[3,+∞[	[3,+∞[	[3,+∞[
3				0	0
4					0

Ce tableau correspond à la matrice DBM des plus courts chemins[1] de *i* à *j* :

	0	**1**	**2**	**3**	**4**	**5**
0	0	+∞	+∞	0	0	0
1	-3	0	+∞	+∞	+∞	+∞
2	-3	+∞	0	+∞	+∞	+∞
3	0	-3	-3	0	0	0
4	0	-3	-3	0	0	0
5	0	-3	-3	0	0	0

En reprenant la même démarche pour le réseau contrôlé avec arcs inhibiteurs :

```
init(<x1, x2, 0, 0, 0, 0, 0>) -> {x1 >= 0, x2 >= 0} ;

fact(0, <x1, x2, x3, x4, x5, x35, x45>) ->
    {x1 >= 0, x2 >= 0, x3 >= 0, x4 >= 0, x5 >= 3, x35 >= 0, x45 >= 0} ;
```

après calcul de Pre* :

```
> init(X) fact(_,X) ;
>
```

L'absence de solution (en Prolog III comme en Prolog IV) est cette fois la preuve de la *non accessibilité des états interdits* à partir des états initiaux.

Le contenu de la base de faits (après simplification) est résumé dans le tableau MARS qui réexprime, sous forme d'une disjonction de conjonctions, la sémantique de la relation d'accessibilité, définie à l'origine sous forme clausale, comme une conjonction de disjonctions.

Les statistiques sur le calcul de Pre* sont les suivantes :

	Nombre d'itérations	**Taille de la base de faits**	**Temps machine en secondes (estimation suivant le solveur)**	
Réseau non contrôlé	10	30	linéaire[a]	15 | 11
			intervalles[b]	7
Réseau contrôlé	11	47	linéaire	29 | 32
			intervalles	32

a. A gauche, mesure donnée par Prolog III (v1.4) incluant les temps d'accès disque (pour les écritures temporaires nécessaires à la conversion par in_term des contraintes en termes).
A droite, temps cpu obtenu après portage du prototype en Sicstus Prolog (v3.7), avec la bibliothèque CLPQ (résolution des contraintes linéaires en précision infinie).

b. Temps cpu donné par Prolog IV (v1.3) — sans indexation.
Bien qu'effectuées sur un même machine, ces mesures de temps présentent une marge d'incertitude pouvant atteindre 20 %.

1. Le sens des inégalités étant inverti (≥ au lieu de ≤), il faut prendre l'opposé des valeurs des variables pour retrouver l'interprétation des plus courts chemins.

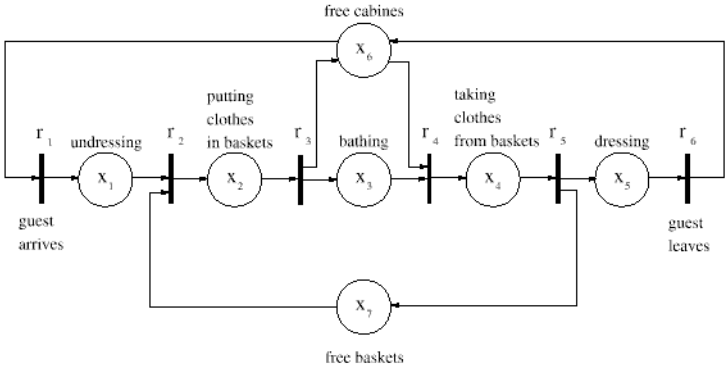
TABLEAU MARS — **Calcul des prédécesseurs - réseau contrôlé**

Iteration	Fait	Définition Prolog III / CLPQ avec <x1,x2,x3,x4,x5,x35,x45> = <A,B,C,D,E,F,G>	Définition Prolog IV
	47	{ G=<3 , E>=0 , B>=0 , G>=2 , F>=2 , D>=5 , C>=2 , A>=1}	[ge(1),ge(0),ge(2),ge(5),ge(0),ge(2),cc(2,3)]
10	46	{ F=<3 , E>=0 , A>=0 , G>=2 , F>=2 , D>=2 , C>=5 , B>=1}	[ge(0),ge(1),ge(5),ge(2),ge(0),cc(2,3),ge(2)]
	45	{ F=<3 , D>=1 , A>=0 , G>=2 , F>=2 , E>=1 , C>=4 , B>=1}	[ge(0),ge(1),ge(4),ge(1),ge(1),cc(2,3),ge(2)]
9	44	{ G=<3 , C>=1 , B>=0 , G>=2 , F>=2 , E>=1 , D>=4 , A>=1}	[ge(1),ge(0),ge(1),ge(4),ge(1),ge(2),cc(2,3)]
	43	{ G=<3 , C>=0 , B>=0 , G>=2 , F>=2 , E>=2 , D>=3 , A>=1}	[ge(1),ge(0),ge(0),ge(3),ge(2),ge(2),cc(2,3)]
	42	{ G=<2 , G>=1 , F>=1 , E>=0 , C>=1 , B>=0 , D>=4 , A>=2}	[ge(2),ge(0),ge(1),ge(4),ge(0),ge(1),cc(1,2)]
	41	{ F=<3 , D>=0 , A>=0 , G>=2 , F>=2 , E>=2 , C>=3 , B>=1}	[ge(0),ge(1),ge(3),ge(0),ge(2),cc(2,3),ge(2)]
8	40	{ F=<2 , G>=1 , F>=1 , E>=0 , D>=1 , A>=0 , C>=4 , B>=2}	[ge(0),ge(2),ge(4),ge(1),ge(0),cc(1,2),ge(1)]
	39	{ F=<2 , G>=1 , F>=1 , E>=0 , A>=0 , D>=2 , C>=4 , B>=1}	[ge(0),ge(1),ge(4),ge(2),ge(0),cc(1,2),ge(1)]
	38	{ F=<2 , G>=1 , F>=1 , E>=1 , D>=0 , A>=0 , C>=3 , B>=2}	[ge(0),ge(2),ge(3),ge(0),ge(1),cc(1,2),ge(1)]
	37	{ F = 0 , G=<1 , G>=0 , E>=0 , D>=0 , A>=1 , C>=2 , B>=3}	[ge(1),ge(3),ge(2),ge(0),ge(0),0,cc(0,1)]
	36	{ G=<2 , G>=1 , F>=1 , E>=0 , B>=0 , D>=4 , C>=2 , A>=1}	[ge(1),ge(0),ge(2),ge(4),ge(0),ge(1),cc(1,2)]
	35	{ G=<2 , G>=1 , F>=1 , E>=1 , C>=0 , B>=0 , D>=3 , A>=2}	[ge(2),ge(0),ge(0),ge(3),ge(1),ge(1),cc(1,2)]
	34	{ G = 0 , F=<1 , F>=0 , E>=0 , C>=0 , B>=1 , D>=2 , A>=3}	[ge(3),ge(1),ge(0),ge(2),ge(0),cc(0,1),0]
7	33	{ F = 0 , G = 0 , E>=0 , C>=0 , B>=1 , D>=2 , A>=3}	[ge(3),ge(1),ge(0),ge(2),ge(0),0,0]
	32	{ G = 0 , F=<1 , F>=0 , E>=0 , C>=1 , B>=1 , D>=2 , A>=2}	[ge(2),ge(1),ge(1),ge(2),ge(0),cc(0,1),0]
	31	{ G=<1 , G>=0 , F>=0 , E>=0 , C>=0 , B>=0 , D>=3 , A>=3}	[ge(3),ge(0),ge(0),ge(3),ge(0),ge(0),cc(0,1)]
	30	{ G=<2 , G>=1 , F>=1 , C>=1 , B>=0 , E>=1 , D>=3 , A>=1}	[ge(1),ge(0),ge(1),ge(3),ge(1),ge(1),cc(1,2)]
	29	{ F=<1 , G>=0 , F>=0 , E>=0 , D>=0 , A>=0 , C>=3 , B>=3}	[ge(0),ge(3),ge(3),ge(0),ge(0),cc(0,1),ge(0)]
	28	{ F = 0 , G=<1 , G>=0 , E>=0 , D>=1 , A>=1 , C>=2 , B>=2}	[ge(1),ge(2),ge(2),ge(1),ge(0),0,cc(0,1)]
	27	{ F=<2 , G>=1 , F>=1 , D>=1 , A>=0 , E>=1 , C>=3 , B>=1}	[ge(0),ge(1),ge(3),ge(1),ge(1),cc(1,2),ge(1)]
	26	{ F = 0 , G = 0 , E>=0 , A>=1 , D>=1 , C>=2 , B>=2}	[ge(1),ge(2),ge(2),ge(1),ge(0),0,0]
6	25	{ F = 0 , G = 0 , E>=0 , B>=1 , D>=2 , C>=1 , A>=2}	[ge(2),ge(1),ge(1),ge(2),ge(0),0,0]
	24	{ G = 0 , F=<1 , F>=0 , E>=0 , B>=1 , A>=1 , D>=2 , C>=2}	[ge(1),ge(1),ge(2),ge(2),ge(0),cc(0,1),0]
	23	{ F = 0 , G=<1 , G>=0 , E>=0 , B>=1 , A>=1 , D>=2 , C>=2}	[ge(1),ge(1),ge(2),ge(2),ge(0),0,cc(0,1)]
	22	{ F=<2 , G>=1 , F>=1 , D>=0 , A>=0 , E>=2 , C>=2 , B>=1}	[ge(0),ge(1),ge(2),ge(0),ge(2),cc(1,2),ge(1)]
	21	{ F=<1 , G>=0 , F>=0 , E>=0 , D>=1 , A>=0 , C>=3 , B>=2}	[ge(0),ge(2),ge(3),ge(1),ge(0),cc(0,1),ge(0)]
	20	{ F = 0 , G=<1 , G>=0 , D>=0 , A>=1 , E>=1 , C>=1 , B>=2}	[ge(1),ge(2),ge(1),ge(0),ge(1),0,cc(0,1)]
	19	{ G=<2 , G>=1 , F>=1 , C>=0 , B>=0 , E>=2 , D>=2 , A>=1}	[ge(1),ge(0),ge(0),ge(2),ge(2),ge(1),cc(1,2)]
	18	{ G=<1 , G>=0 , F>=0 , E>=0 , C>=1 , B>=0 , D>=3 , A>=2}	[ge(2),ge(0),ge(1),ge(3),ge(0),ge(0),cc(0,1)]
5	17	{ G = 0 , F=<1 , F>=0 , C>=0 , B>=1 , E>=1 , D>=1 , A>=2}	[ge(2),ge(1),ge(0),ge(1),ge(1),cc(0,1),0]
	16	{ G=<1 , G>=0 , F>=0 , C>=0 , B>=0 , E>=1 , D>=2 , A>=2}	[ge(2),ge(0),ge(0),ge(2),ge(1),ge(0),cc(0,1)]
	15	{ F=<1 , G>=0 , F>=0 , D>=0 , A>=0 , E>=1 , C>=2 , B>=2}	[ge(0),ge(2),ge(2),ge(0),ge(1),cc(0,1),ge(0)]
	14	{ F=<1 , G>=0 , F>=0 , E>=0 , B>=1 , A>=0 , D>=2 , C>=3}	[ge(0),ge(1),ge(3),ge(2),ge(0),cc(0,1),ge(0)]
	13	{ G=<1 , G>=0 , F>=0 , E>=0 , B>=0 , A>=1 , D>=3 , C>=2}	[ge(1),ge(0),ge(2),ge(3),ge(0),ge(0),cc(0,1)]
	12	{ F = 0 , G=<1 , G>=0 , D>=1 , C>=1 , B>=1 , A>=1 , E>=1}	[ge(1),ge(1),ge(1),ge(1),ge(1),0,cc(0,1)]
4	11	{ G = 0 , F=<1 , F>=0 , D>=1 , C>=1 , B>=1 , A>=1 , E>=1}	[ge(1),ge(1),ge(1),ge(1),ge(1),cc(0,1),0]
	10	{ G = 0 , F=<1 , F>=0 , D>=0 , C>=0 , B>=1 , A>=1 , E>=2}	[ge(1),ge(1),ge(0),ge(0),ge(2),cc(0,1),0]
	9	{ F = 0 , G=<1 , G>=0 , D>=0 , C>=0 , B>=1 , A>=1 , E>=2}	[ge(1),ge(1),ge(0),ge(0),ge(2),0,cc(0,1)]
	8	{ G>=0 , F>=0 , E>=0 , B>=0 , A>=0 , D>=3 , C>=3}	[ge(0),ge(0),ge(3),ge(3),ge(0),ge(0),ge(0)]
	7	{ F=<1 , G>=0 , F>=0 , B>=1 , A>=0 , E>=1 , D>=1 , C>=2}	[ge(0),ge(1),ge(2),ge(1),ge(1),cc(0,1),ge(0)]
3	6	{ G=<1 , G>=0 , F>=0 , B>=0 , A>=1 , E>=1 , D>=2 , C>=1}	[ge(1),ge(0),ge(1),ge(2),ge(1),ge(0),cc(0,1)]
	5	{ G>=0 , F>=0 , B>=0 , A>=0 , E>=1 , D>=2 , C>=2}	[ge(0),ge(0),ge(2),ge(2),ge(1),ge(0),ge(0)]
	4	{ F=<1 , G>=0 , F>=0 , D>=0 , C>=1 , B>=1 , A>=0 , E>=2}	[ge(0),ge(1),ge(1),ge(0),ge(2),cc(0,1),ge(0)]
2	3	{ G=<1 , G>=0 , F>=0 , D>=1 , C>=0 , B>=0 , A>=1 , E>=2}	[ge(1),ge(0),ge(0),ge(1),ge(2),ge(0),cc(0,1)]
1	2	{ G>=0 , F>=0 , D>=1 , C>=1 , B>=0 , A>=0 , E>=2}	[ge(0),ge(0),ge(1),ge(1),ge(2),ge(0),ge(0)]
0	1	{ G>=0 , F>=0 , E>=3 , D>=0 , C>=0 , B>=0 , A>=0}	[ge(0),ge(0),ge(0),ge(0),ge(3),ge(0),ge(0)]

7 Extensions de la méthode

Parce que la procédure de calcul du point fixe ne parvient pas à se stabiliser ou converge trop lentement, il arrive que la génération s'enlise. L'«exemple d'école» suivant est à cet égard plus difficile qu'il n'y paraît.

Exemple Ce réseau de Petri à 7 places et 6 transitions (cité dans Arnold *et al.*) modélise le fonctionnement d'une piscine un peu particulière. Un nageur doit d'abord trouver une cabine (stock en 6) pour se changer (transition 1), puis un panier libre (stock en 7) pour ranger ses vêtements (transition 2). Il libère la cabine et dépose son panier à la consigne pour aller nager (transition 3). Lorsqu'il revient du bain, il reprend une cabine (transition 4), récupère ses vêtements, rend son panier (transition 5), enfin se rhabille et libère la cabine en sortant (transition 6). S'il y a c cabines et p paniers ($c, p \geq 1$), le marquage initial est <0,0,0,0,0, c, p>.



Les situations de blocage du réseau sont décrites par la formule disjonctive : $x_2 = x_4 = x_5 = x_6 = 0 \wedge (x_1 = 0 \vee x_7 = 0)$. Il n'y a plus de cabine libre (transitions 1 et 4 infranchissables) ou de panier libre (2 infranchissable) et ces ressources ne peuvent être libérées (3, 5, 6 infranchissables). Soit :

```
fact(0, <x1, 0, x3, 0, 0, 0, 0>) -> {x1 >= 0, x3 >= 0} ;
fact(0, <0, 0, x3, 0, 0, 0, x7>) -> {x3 >= 0, x7 >= 0} ;
```

Mais en partant de ces faits, la procédure de calcul patauge.

La méthode de Bérard & Fribourg propose de remédier à ce type situation en rajoutant des fusions et en passant aux méta-transitions.

Définition Une *fusion f* correspond à l'application en une seule étape d'une séquence de transitions $t_1 t_2 ... t_k$:

$$\boldsymbol{x} \overset{f}{\rightarrow} \boldsymbol{x'} \quad \Leftrightarrow \quad \exists\ \boldsymbol{x}_1 \ldots \boldsymbol{x}_{k-1} \quad \boldsymbol{x} \overset{t_1}{\rightarrow} \boldsymbol{x}_1 \overset{t_2}{\rightarrow} \ldots \rightarrow \boldsymbol{x}_{k-1} \overset{t_k}{\rightarrow} \boldsymbol{x'}$$

Les fusions peuvent accéler la convergence en favorisant la production de conséquences logiques de plus grand intérêt. L'ajout de fusions n'affecte ni la structure des règles ni le calcul de l'ensemble des états accessibles.

Définition Une *méta-transition* t^+ correspond à l'application d'une transition *t*, itérée un nombre indéfini de fois :

$$x \xrightarrow{t^+} x' \quad \Leftrightarrow \quad \exists n \in \mathbb{N}^* \quad x \xrightarrow{t^n} x'$$

On traduit ici avec des contraintes linéaires l'équivalent de l'étoile de Kleene des expressions régulières.

Exemple La méta-transition généralisant l'action *produire* dans le système producteur-consommateur enlève *n* de jetons de la place 1 et en dépose *n* en place 3. Elle s'écrit :

```
p(<x1,x2,x'3>) -> p(<x1,x2,x3>) ,
    {x'3 = x3 + n, x1 >= 1, x2 >= 0, x3 >= 0, n >= 1} ;
```

Appliquée au marquage initial (1,1,0), elle induit l'accessibilité à l'ensemble des marquages (1,1,*n*) vérifiant $n \geq 1$[1].

La variable *n* (implicitement entière) représente un nombre de jetons inconnu. Son élimination (licite) donne ici simplement :

```
p(<x1,x2,x'3>) -> p(<x1,x2,x3>) ,
    {x'3 >= x3 + 1, x1 >= 1, x2 >= 0, x3 >= 0} ;
    % l'égalité x'3 = x3 + 1 de la transition d'origine devient une inégalité.
```

De façon générale, l'introduction des méta-transitions change la structure syntaxique du système de contraintes, le faisant sortir du cadre très strict des contraintes de tensions et de celui (un peu moins strict) des contraintes 1-monotones. La relaxation sur les réels n'est plus exacte (et l'utilisation des intervalles pour la résolution linéaire ne se justifie plus).

Exemple La version itérée de la transition 4 du réseau «piscine et paniers» s'écrit :

```
reachable{<x1-n,x2+n,x4,x5,x6,x7-n>, <x1,x2,x3,x4,x5,x6,x7>) ->
    {x1 >= n, x7 >= n, n >=1,
     x2 >= 0, x3 >= 0, x4 >= 0, x5 >= 0, x6>= 0} ;
```

Cette règle est équivalente à la suivante[2] :

```
reachable{<x'1,x'2,x4,x5,x6,x'7>, <x1,x2,x3,x4,x5,x6,x7>) ->
    {x'1 + x7  = x1 + x'7,
     x'1 + x'2 = x1 + x2,
     x1  >= x'1 + 1,
     x'1 >= 0, x2 >= 0,  x'7 >= 0,
     x3  >= 0, x4 >= 0, x5 >= 0, x6>= 0} ;
```

En l'absence de certitude, tout ce que l'on peut affirmer est que le calcul de Post* sur les réels aboutit à une *approximation par excès*, $\overline{\text{Post}}^*(\textit{Source})$, de l'ensemble des états accessibles. Cette approximation (plus facile à calculer) peut servir à la démonstration d'invariants (plus faibles).

1. Ce qui n'est pas sans évoquer un «compactage» à la manière de Karp et Miller (1969).
2. Fortunément, la permanence des coefficients ±1 dans les inégalités (après élimination de la variable existentielle *n*) laisse à penser qu'au cours du calcul, d'autres éliminations de variables d'états seront encore licites.

Si le calcul se termine,

- $\overline{\text{Post}}^*(Source) \cap Cible \neq \emptyset$ ne constitue plus une preuve que la région cible est accessible, mais seulement une *présomption d'accessiblité*. Montrer qu'il existe des solutions entières exige alors de retracer un chemin d'accès, sans méta-transitions.
- $\overline{\text{Post}}^*(Source) \cap Cible = \emptyset$ garde le statut de *preuve d'inaccessibilité* (les états atteignables étant forcément dans l'union des régions calculées).

Dans bien des cas, l'effet d'une séquence de transitions sur un réseau de Petri ne dépend pas de l'ordre des transitions. Un choix judicieux de fusions à utiliser comme méta-transitions est alors suggéré par un processus de décomposition du réseau. Partant du langage des transitions $(t_1 + t_2 + ... + t_m)^*$ on peut, dans certains cas, aboutir à un langage «plat» $f_1^* f_2^* ... f_n^*$, fait de fusions de ces transitions ($f_j = t_{i_1} t_{i_2} \ldots t_{i_k}$) et définissant de manière équivalente l'accessibilité d'un état (Fribourg & Olsén).

En termes d'accélération de la convergence, pour le petit exemple de la piscine, ce «cocktail» fusions + méta-transitions prend nettement l'avantage sur la technique d'abstraction par élargissement (*widening*) reprise par Delzanno & Podelski et également implantée en programmation logique avec contraintes[1].

Exemple Aux 6 transitions d'origine de l'exemple de la piscine ($r_1, r_2, r_3, r_4, r_5, r_6$), on rajoute 5 fusions obtenues avec le programme Prolog d'Olsén[2] (r_2r_3, r_4r_5, $r_1r_2r_3$, $r_4r_5r_6$, $r_1r_2r_3r_4r_5$). En lançant le calcul des prédécesseurs des états de blocage par méta-transitions, on obtient très rapidement un point fixe.

Outil de vérification	Technique d'accéleration	Nombre d'iterations	Taille de la base[a] de faits	Temps machine (en sec.)
HyTech	fusions + méta	22	10	
Prolog III / CLPQ[b]	fusions + méta	6	17	22
DP Checker / CLPR[c]	widening		*ne s'arrête pas* [d]	
	fusions + méta	10	145	116

a. On peut vérifier (après élimination des faits redondants et «triangulation» des polyèdres non comparables) que les bases de faits engendrées sont équivalentes .

b. On donne ici, à titre de comparaison, les temps obtenus avec Sicstus Prolog, après portage du prototype en CLPQ.
On obtient les mêmes statistiques (en nombre d'iterations et taille de la base) en partant du seul premier fait ou des deux faits décrivant les situations d'interblocage.

c. L'outil Deductive Model Checker de Delzanno & Podelski utilise Sicstus Prolog et la bibliothèque CLPR pour la résolution des contraintes linéaires sans la précision infinie. Dans les deux cas, les fichiers correspondant aux bases de règles utlisées (avec ou sans méta-transitions) sont engendrés par programme à partir de la description du réseau.

d. Exécution avortée au bout d'une heure (plusieurs milliers de faits engendrés après une trentaine d'itérations).

1. www.mpi-sb.mpg.de/~delzanno
2. www.ida.liu.se/labs/logpro/

On peut alors montrer que $\overline{\text{Pre}}^*(\textit{Cible})$ intersecte la région des marquages initiaux. On voit ici à l'œil nu qu'avec c cabines et p paniers ($c, p \geq 1$), on aboutit effectivement à un *deadlock* en suivant le chemin $(r_1 r_2 r_3)^p (r_1)^c$.

Il nous est peu coûteux, en Prolog, d'instrumenter le code pour reconstituer, par énumération, un cheminement type entre un marquage initial et un marquage cible suspecté. En effet, les solutions obtenues par évaluation ascendante construisent naturellement une arborescence dont on peut, *a posteriori*, reconstituer les branches.

Dans le cas du protocole de communication PNCSA (Protocole Normalisé de Connexion au Système d'Autorisation) du GIE CB dont la description est un réseau de Petri sophistiqué, la mémorisation de cet arbre d'accessibilité nous a permis de parachever la preuve de l'existence d'une situation de blocage.

Exemple Le réseau PNCSA a au total 31 places et 38 transitions. Dans le marquage initial, seules les places E et F ont un jeton (les autres places sont vides).

Les états de blocage qui servent de point de départ au calcul de Pre* sont définis par un ensemble de faits de la forme :

```
deadlock(<A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,
    Z0,Z1,Z2,Z3,Z4>) ->

    {Z=0, Q=0, L=0, G=0, I=0, E=0,
     J=0, N=0, Y=0, Z1=0,T=0, U=0, X=0, Z4=0, Z3=0,
     R=1, V=0,
     F=1, H=0,
     P=1,
     A >= 0, B >= 0, C >= 0, D >= 0,
     K >= 0,
     M >= 0,
     O >= 0,
     S >= 0, W >=0, Z0 >=0, Z2 >=0} ;
```

Voici les résultats du calcul sur un réseau simplifié :

Outil de vérification	Nombre d'iterations	Taille de la base de faits	Temps machine (en sec.)
HyTech	9	3	
Prototype Prolog III / CLPQ	5	8	10

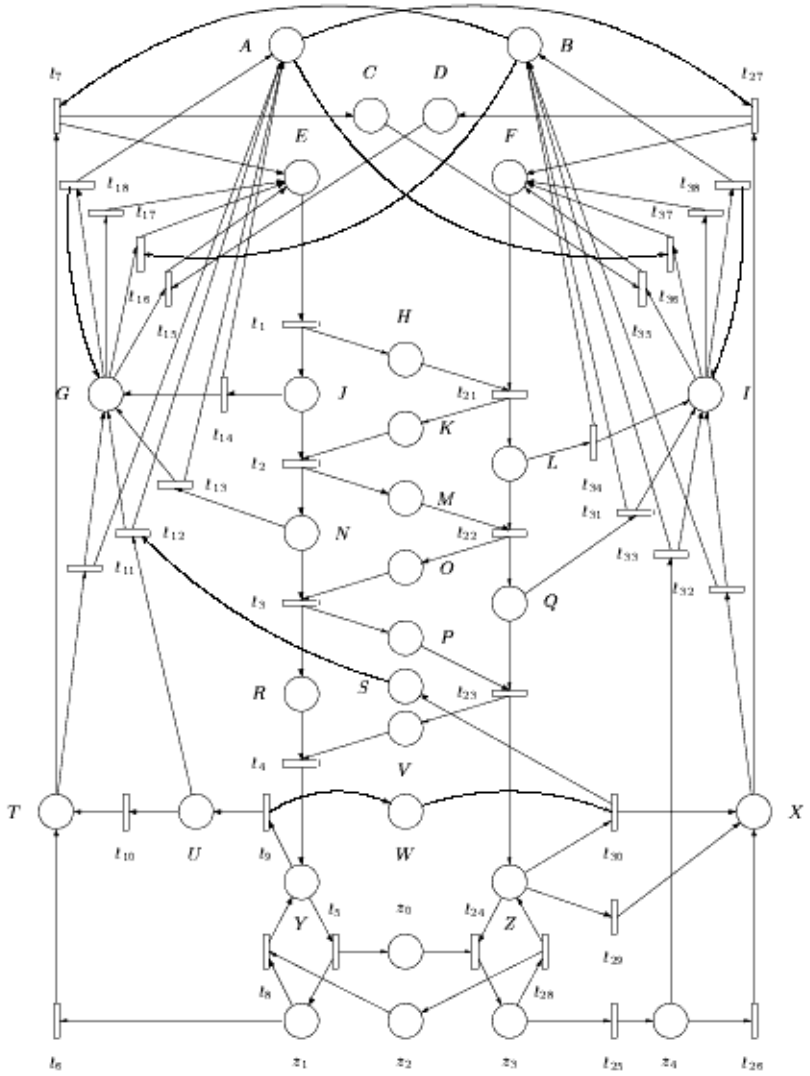
Le marquage initial fait partie de l'union des ensembles d'états calculés. $(t_1 t_{14} t_{17} t_{21} t_{34} t_{37})^\alpha (t_1 t_{21} t_2 t_{13} t_{17} t_{22} t_{31} t_{37})^\beta (t_1 t_2 t_3)^\gamma (t_{21} t_{34} t_{37})^\delta$ avec $\alpha, \beta, \gamma, \delta \geq 1$ est une séquence de transitions (fusions) remontée par le calcul et suspectée de conduire du marquage initial à un état de blocage. Instancier α, β, γ et δ à 1 suffit à démontrer, par simulation symbolique, l'accessibilité d'un tel état.

marquage initial : <0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0>

marquage atteint : <2,3,0,0,0,1,0,0,0,0,1,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0>

Cet exemple souligne l'importance de la découverte de «bonnes fusions» car il suffit d'en suivre un petit nombre pour obtenir une preuve d'accessibilité. Avant l'expérimentation de cette méthode avec HyTech, l'existence d'un *deadlock* était insoupçonnée.

FIGURE PNCSA **Réseau de Petri spécifiant le protocole normalisé de connexion PNCSA**



8 En résumé

Reprenant point par point la méthode de Bérard & Fribourg, nous avons montré comment modéliser un réseau de Petri à travers un programme logique avec contraintes et comment résoudre des problèmes d'accessibilité en utilisant la sémantique de point-fixe de ce programme.

Nous avons souligné les propriétés qui, dans la structure logique des règles (2 littéraux par clause) et dans la structure linéaire des systèmes de contraintes (2 variables par inégalité), nous permettent :

- de parcourir l'espace de recherche de manière complète et efficace
- de traiter les variables entières comme des variables réelles, tout en conservant la complétude de la résolution[1].

Nous avons décomposé le problème de la vérification en deux étapes :

1. le calcul, dans un premier temps, de l'ensemble des marquages accessibles
2. l'interrogation, dans un second temps, de la base constituée pour confirmer ou réfuter une propriété d'accessibilité donnée.

Progressivement, nous sommes passés d'un calcul sur des états à un calcul sur des ensembles d'états ou régions ; puis, dans le but d'accélérer la convergence, à un calcul sur des approximations par excès de ces régions.

En donnant une interprétation polyédrique aux trois opérations fondamentales que sont le test de satisfaisabilité (non vacuité de l'intersection), l'élimination de variables (projection) et le test de consécution (inclusion), nous avons esquissé et justifié deux approches en programmation par contraintes :

- l'une, relativement généraliste, faisant appel à un solveur linéaire, dans la veine de Prolog III
- l'autre, inédite mais de portée plus restreinte, utilisant le solveur d'intervalles de Prolog IV.

Tout en demeurant relativement légère, la mise en œuvre de ces solutions passe par une méta-programmation — commune — des règles (pour l'évaluation ascendante) et une méta-programmation — spécifique — des contraintes (pour le test d'inclusion).

Leur réalisation en Prolog nous a permis de reproduire les expérimentations précédemment menées, en les complétant, dans le cas des méta-transitions, par une automatisation des vérifications manuelles de l'accessibilité.

A l'instar de la démarche qu'elles appuient, ces solutions techniques devraient être applicables à la vérification formelle d'autres types d'automates à nombre fini ou infini d'états discrets — quoique plus difficilement avec les intervalles, car des problèmes de convergence lente pourraient bien resurgir.

1. Hors méta-transitions.

CHAPITRE VIII

Conclusion

Bilan, références, perspectives

> «Indulge your passion for science, says she, but let your science be human, and such as may have a direct reference to action and society. Abstruse thought and profond researches I prohibit, and will severely punish, by the pensive melancholy which they introduce, by the endless uncertainty in which they involve you, and by the cold reception which your pretended discoveries shall meet with, when communicated.
> Be a philosopher; but, amidst all your philosophy, be still a man.»
>
> D. HUME

Tout au long de cette thèse, nous avons exploité de manière systématique un fait syntaxique simple, facilement repérable dans un modèle linéaire exprimé sous la forme $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$, puisqu'il est lié à la structure de la matrice **A**. Cette particularité — avoir au plus un coefficient strictement positif par ligne — confère une propriété remarquable à l'ensemble des solutions du système, réelles ou entières (si **A** et $\boldsymbol{b}$ sont à coefficients entiers). En particulier, toute intersection avec un cône rectangulaire $\boldsymbol{x} \geq \boldsymbol{c}$ (où $\boldsymbol{c}$ est un vecteur d'entiers) a, quand elle est non vide, un plus petit élément.

Sur un ensemble borné inférieurement, le problème de la satisfaisabilité d'un tel système se ramène donc à un problème d'optimisation particulier.

Nous avons montré notamment que ce problème était abordable par les techniques de réduction d'intervalles, mises en avant en programmation logique avec contraintes. D'autres méthodes, plus classiques, sont naturellement candidates dans cette voie. Un réexamen de problèmes informatiques classiques nous a permis de les comparer et de les rapprocher, avant de les appliquer au problème de la vérification des réseaux de Petri.

On retrouve, en effet, comme extrêmum global dans ces cas bien connus :

- le plus petit modèle
  — pour une base de clauses de Horn propositionnelles cohérente
- les plus courts chemins entre une origine et toutes les autres destinations
  — sur un graphe ne présentant pas de circuit absorbant.

1 Un dernier exemple

Avant de refermer cette étude, et juste après l'évocation ces problèmes combinatoires discrets, arrêtons-nous un instant sur un problème de résolution numérique de nature continue, de manière à brosser un tableau tout à fait complet.

Considérons tout simplement un système linéaire $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ de n équations à n inconnues dans lequel la matrice $\mathbf{A}$ possède au plus un coefficient > 0 par ligne. Ce système est la conjonction de deux systèmes d'inégalités monotones : $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$, $\mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}$.

Supposons de plus la matrice $\mathbf{A}$ inversible. D'après le théorème III.2, l'inverse de $\mathbf{A}$ est dans ce cas une matrice positive[1] et le système admet une solution unique, $\boldsymbol{x}^* = \mathbf{A}^{-1}\boldsymbol{b}$, qui est à la fois le minimum de $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$ et le maximum de $\mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}$.

Remarque Avec en particulier $\boldsymbol{b} \geq 0$, on obtient $\boldsymbol{x}^* \geq 0$ et par conséquent $\boldsymbol{x} \geq 0$.
On a ainsi la propriété : $\mathbf{A}\boldsymbol{x} \geq 0 \Rightarrow \boldsymbol{x} \geq 0$.

Donnons nous maintenant un minorant $\boldsymbol{c}$ de $\boldsymbol{x}$ (si l'on a $\boldsymbol{b} \geq 0$, 0 fait l'affaire). Le théorème IV.3 nous assure qu'avec une grille de discrétisation convenablement choisie, le minimum du pavé $\boldsymbol{X}^-$ des domaines réduits des variables du système $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}$, $\boldsymbol{x} \geq \boldsymbol{c}$ peut approcher $\boldsymbol{x}^*$ par défaut, à moins de ε près. De même, le maximum du pavé $\boldsymbol{X}^+$ des domaines réduits de variables du système $\mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}$, $\boldsymbol{x} \geq \boldsymbol{c}$ peut approcher $\boldsymbol{x}^*$ par excès, à moins de ε près[2].

Pour le système global, $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$, $\boldsymbol{x} \geq \boldsymbol{c}$, le pavé $\boldsymbol{X}$ des domaines réduits des variables est inclus dans $\boldsymbol{X}^-$ et dans $\boldsymbol{X}^+$ et vérifie :

$$\min \boldsymbol{X}^- \leq \min \boldsymbol{X} \leq \boldsymbol{x}^* \leq \max \boldsymbol{X} \leq \max \boldsymbol{X}^+.$$

On obtient donc, avec les intervalles, un encadrement à moins de 2ε près de la solution $\boldsymbol{x}^*$ du système d'équations linéaires.

On explique ainsi la qualité de l'approximation obtenue avec les intervalles lors de la résolution de l'équation de la chaleur.

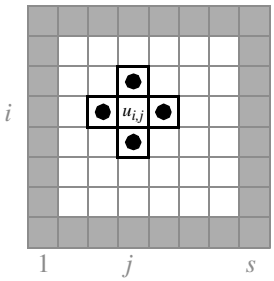
Exemple L'équation de Laplace, $\Delta\boldsymbol{x} = 0$, modélise dans le monde physique des phénomènes «dissipatifs» de nature très variée. C'est en particulier elle qui régit la répartition stationnaire des températures dans un milieu conduisant la chaleur, comme une plaque métallique. Elle est, à ce titre, représentative d'une large classe de problèmes numériques.

1. Au sens où tous ses coefficients sont positifs ou nuls.
Une matrice carrée inversible dont l'inverse est positive est une matrice *monotone*.
2. La donnée du système et du minorant de $\boldsymbol{x}$ suffisent pour que la propagation déduise un majorant de $\boldsymbol{x}$.

La discrétisation de cette équation aux dérivées partielles selon un schéma à 5 points donne, pour une plaque carrée de côté s, un système de s^2 équations à s^2 inconnues de la forme :

$$4\,x_{i,j} - x_{i,j+1} - x_{i,j-1} - x_{i-1,j} - x_{i+1,j} = 0$$

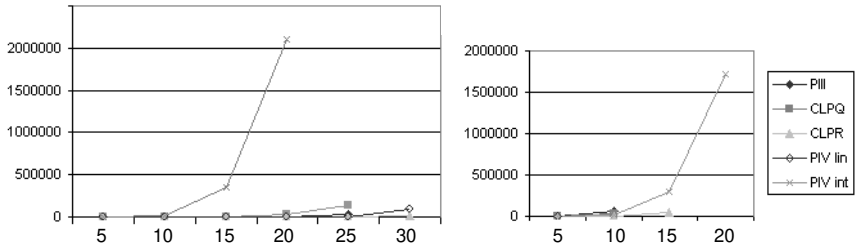
Les conditions aux limites expriment les températures sur les bords (par exemple : $x_{0,j} = x_{s+1,j} = x_{i,0} = x_{i,s+1} = 1$).



La matrice **A** de ce système est bien connue est l'on peut montrer qu'elle est inversible[1]. Sa diagonale est égale à 4**I**. Elle est très creuse, tous ses coefficients hors diagonaux étant négatifs ou nuls.

La résolution du système d'équations $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{c}$ par les intervalles donne, pour un système 400×400 ($s = 20$), jusqu'à 5 décimales exactes après la virgule pour des calculs en 32 bits en Prolog IV.
A titre de comparaison, l'élimination de Gauss effectuée de manière incrémentale, sans contrôle de la précision (avec des nombres à virgule flottante), devient rapidement instable et donne numériquement des résultats complètement faux (de `-1859861.3` à `3808783.7` par exemple en CLPR pour $s = 20$, alors que la solution exacte vaut uniformément 1 !).

Système $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{c}$ Système $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{c}$



Temps cpu (en ms), hors affichage, en fonction de s (test de satisfaisabilité).

On constate que la résolution approchée par les intervalles se trouve accélérée lorsqu'on ne lui soumet que le sous-système $\mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{c}$ (puisqu'il y a moins de réductions à faire) — au contraire de la résolution exacte qui, après avoir été la plus performante, s'effondre (avec les solveurs incrémentaux testés). Le fait est que l'on passe d'une élimination de Gauss au simplexe.

1. Elle est rangée sous la catégorie des M-matrices (matrices de Minkowski).

Remarque Quand tous les coefficients diagonaux de $\mathbf{A}$ sont strictement positifs, comme dans l'exemple précédent, on peut se ramener au cas où ils sont égaux à 1. $\mathbf{A}$ se décompose alors en $\mathbf{I} - \mathbf{N}$ où $\mathbf{I}$ est la matrice identité et $\mathbf{N}$ une matrice positive à diagonale nulle (formée des valeurs absolues des coefficients négatifs de $\mathbf{A}$).

L'équation $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ se réécrit : $(\mathbf{I} - \mathbf{N})\boldsymbol{x} = \boldsymbol{b}$, soit : $\boldsymbol{x} = \mathbf{N}\boldsymbol{x} + \boldsymbol{b}$.

Cette équation implicite est à la base du schéma d'itération de Jacobi pour calculer $(\mathbf{I} - \mathbf{N})^{-1}\boldsymbol{b}$, à l'aide d'une suite croissante (correspondant au développement en série de l'inverse) :

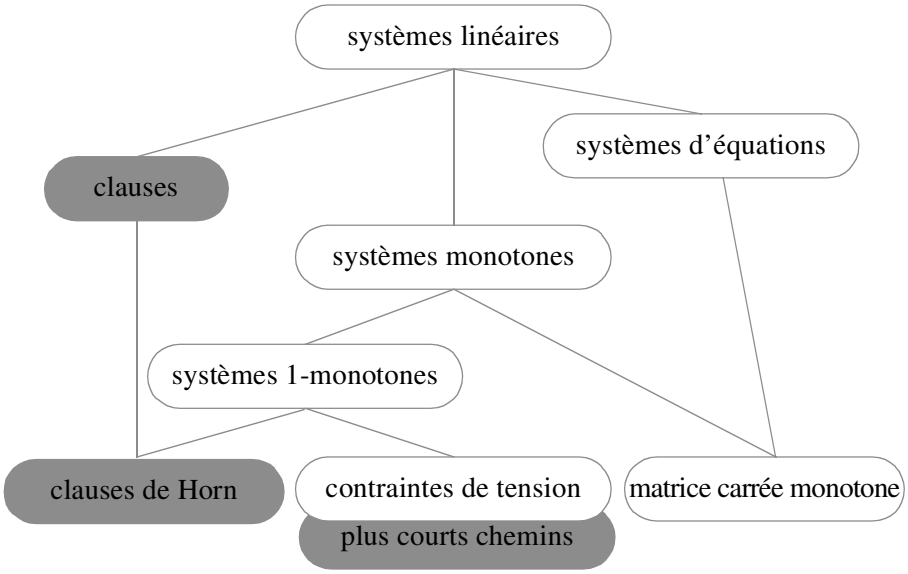
$$\boldsymbol{x}_{n+1} = \mathbf{N}\boldsymbol{x}_n + \boldsymbol{b}.$$

Les réductions d'intervalles $\boldsymbol{X}_n \to \boldsymbol{X}_{n+1}$ appliquent similairement les coupes :

$$\inf(\boldsymbol{X}_{n+1}) \geq \lfloor \mathbf{N} \cdot \inf(\boldsymbol{X}_n) + \boldsymbol{b} \rfloor.$$

2 Bilan

En nous concentrant sur la classe des systèmes linéaires monotones, nous avons exploré la hiérarchie suivante :



Sur le schéma figurent en gris les domaines d'application où la résolution s'entend sur les entiers[1].

1. On rappelle que dans le cas des clauses de Horn et des plus courts chemins — et plus généralement, dans le cas des systèmes 1-monotones, la relaxation sur les réels suffit à résoudre formellement ces énoncés. Pour les plus courts chemins, cette relaxation correspond à la résolution d'un problème de flot minimum ou de tension maximum.

Nous avons comparé des méthodes de résolution exacte (Fourier) et approchée (Intervalles) pour la résolution de ces systèmes.

- Les méthodes exactes sont des méthodes directes qui procèdent par élimination de variables.
- Les méthodes approchées sont des méthodes itératives qui procèdent par réduction de domaines .

Nous pouvons donc dresser le récapitulatif suivant :

Problème type		Equation de Laplace discrétisée
Matrice		carrée monotone [a]
Propriété		**A** et ^{t}**A** ont la propriété (m)[b]
Ensemble des solutions		réduit à un point
Elimination de Fourier		*se spécialise en élimination de Gauss*
Réduction d'intervalles		*réalise les itérations de Jacobi*
Complexité	**F**	$O(n^5)$ en précision infinie[c]
Exactitude	**I**	résolution approchée
Complétude	**I**	non[d]
Simplification	**I**	encadrement de la solution à ε près

a. Autres classifications : M-matrice, matrice totalement Leontief (à diagonale > 0).

b. Avoir au plus un coefficient > 0 par ligne.

c. Chiffre donné pour une matrice dense (cf. analyse input-output). Ne tient pas compte de la structure tri-diagonale par blocs de la matrice du laplacien discrétisé (où $n = s^2$).

d. N'apporte pas la preuve que **A** est inversible.

Problème type		Clauses de Horn propositionnelles
Matrice		matrice d'incidence d'hypergraphe [a]
Propriété		**A** a la propriété $(1\text{-}m)$[b]
Ensemble des solutions		inf. demi-treillis (dans le treillis booléen)
Elimination de Fourier		*se spécialise en résolution unitaire*
Réduction d'intervalles		*réalise la résolution unitaire*
Complexité	**F**	quadratique[c]
Exactitude	**I**	résolution exacte
Complétude	**I**	oui
Simplification	**I**	test supplémentaire exigé (1 par variable)

a. Hypergraphe dont les arcs ont au plus un sommet d'arrivée.

b. Cas particulier de (m) où tous les coefficients > 0 sont égaux à 1.

c. En $O(n.l)$ où n est le nombre de variables et l le nombre total de leurs occurrences.

Problème type		Tensions et plus courts chemins
Matrice		matrice d'incidence de graphe
Propriété		**A** et –**A** ont la propriété (1-*m*)
Ensemble des solutions		treillis (dans $\mathbb{R}$ ou $\mathbb{Z}$)
Elimination de Fourier		*se spécialise en algorithme de Pratt*[a]
Réduction d'intervalles		*réalise l'algorithme de Ford*
Complexité	**F**	cubique
Exactitude	**I**	résolution exacte
Complétude	**I**	si domaines bornés[b]
Simplification	**I**	calcul des projections[c]

a. Voir aussi : algorithme de Floyd.

b. Avec une grille de calcul suffisamment étendue.

c. Les projections sont exactes si les domaines sont bornés.

Ces comparaisons font apparaître une analogie profonde entre la résolution approchée par les intervalles et les techniques itératives de l'analyse numérique (pour les problèmes continus) ou de la programmation dynamique (pour les problèmes discrets).

Cette dernière se fonde sur une vue graphique des contraintes, en l'occurrence, des hyperarcs ayant un unique sommet destination (Jeroslow *et al.*, 1992). D'un système d'inégalités, elle ne s'occupe que de traquer la solution particulière $\boldsymbol{m} = \min P$ où P est le polyèdre des solutions du système.

Elle déduit de : $x_j \geq \max_{i \in I_j}\left(\sum_{k \neq j} \frac{a_{ik}}{a_{ij}} \cdot x_k + \frac{b_i}{a_{ij}}\right)$ (où I_j désigne l'ensemble des lignes dans lesquelles le coefficient de *j* est strictement positif), les équations de récurrence : $m_j = \max_{i \in I_j}\left(\sum_{k \neq j} \frac{a_{ik}}{a_{ij}} \cdot m_k + \frac{b_i}{a_{ij}}\right)$.

Equations que l'on retrouve presqu'à l'identique dans les formules de réduction de domaines. La seule différence tient à la convergence qui est par construction garantie avec les intervalles, du fait de la finitude de la grille d'approximation. Tant que l'on n'atteint pas ses limites, les calculs en nombres entiers ne posent pas de problème de précision.

Nous devons enfin rajouter à la liste précédente, l'exemple de la formulation linéaire du PGCD. Pour résoudre ce système monotone en nombres entiers, nous avons appliqué une technique de coupes particulière. Dans une formulation utilisant les intervalles, ces troncatures ne se programment plus. Elles se déclarent à travers la contrainte non linéaire `int(x)` («*x* est égal à sa partie entière»).

Si, en suivant Montanari, nous appelons «relaxation parfaite» toute forme de relaxation d'un problème linéaire en nombres entiers qui n'exige pas de recourir à l'énumération, nous obtenons ce second tableau récapitulatif :

Problème en nombre entiers	Type de relaxation	Complétude
Horn-SAT	linéaire continue	*parfaite*
Plus courts chemins	linéaire continue	*parfaite*
Système 1-monotone	linéaire continue	*parfaite*
SAT	linéaire continue	imparfaite
	de Horn	imparfaite
Système monotone	relaxation linéaire	imparfaite
	domaines finis	*parfaite*[a]

a. Sous réserve de disposer d'une grille de taille suffisante.

La mention *domaines finis* pour la relaxation signifie : résolution du système linéaire en nombre entiers par réduction d'intervalles, sur des *domaines bornés*, *avec* formulation des *contraintes d'intégrité* `int/1`.

Remarque L'approximation effectuée par le calcul des intervalles est censée donner des résultats toujours corrects, même s'ils ne sont pas toujours précis. Or, dans beaucoup d'implantations des domaines finis, ce principe peut être mis à mal lorsque la limite de précision des calculs est atteinte.

Exemple Equation à deux variables (formant un système monotone) dont la résolution approchée n'est même plus correcte en CLP(FD) [1] :

```
| ?- fd_domain([X,Y],[1,56683]),
   56683*Y #>= 12741*X, 12741*X #>= 56683*Y.

   no
| ?- X#>1, Y#>1, 56683*Y #>= 12741*X, 12741*X #>= 56683*Y.

   no
| ?- X=56683, Y=12741, 56683*Y #>= 12741*X, 12741*X #>= 56683*Y.

   X = 56683
   Y = 12741
   yes
```

En Prolog IV, les domaines infinis sont une échappatoire qui préserve la correction des résultats.

```
>>  X ge 1, Y ge 1, 56683.*.Y ge 12741.*.X, 12741.*.X ge 56683.*.Y.

    Y ~ ge(1),
    X ~ ge(`>4.448865`).

>>  X ge 1, Y ge 1, 56683.*.Y ge 12741.*.X, 12741.*.X ge 56683.*.Y,
    int(X), int(Y).

    Y ~ ge(2161),
    X ~ ge(9614).
```

1. Résultats en GNU-Prolog, communiqués par F. Fages.

3 Références et travaux apparentés

3.1 Chapitre III

Les résultats généraux qui éclairent cette thèse sont ceux, assez anciens, de Veinott (dont on retrouvera une synthèse dans son article de 1989).
Ils établissent la dualité entre les familles de polyèdres $P = \{\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} \geq \boldsymbol{b}\}$ ayant une structure de demi-treillis et les systèmes de substitution dits de Leontief ${}^{t}\mathbf{A}\boldsymbol{y} = \boldsymbol{c}\ ,\ \boldsymbol{y} \geq 0$. Suivant cette terminologie, les matrices $\mathbf{A}$ de notre étude sont des transposées de matrices pré-Leontief (définies comme ayant au plus un coefficient strictement positif par colonne).

La propriété selon laquelle le plus petit élément (s'il existe) est à coordonnées entières, lorsque tous les coefficients strictement positifs sont égaux à 1, fait partie d'un énoncé de Chandrasekaran (1984). Nous avons donné une preuve élémentaire de ce résultat fort, qui simplifie considérablement la résolution de certains problèmes en nombres entiers (il les fait passer, en effet, de la classe des problèmes combinatoires difficiles à la classe des problèmes bien résolus). Notre démonstration repose essentiellement sur une remarque de Williams (1992) permettant de calquer l'élimination d'une variable entière sur celle de Fourier d'une variable réelle.

Nous avons finalement emprunté le terme *inégalités monotones* à Cohen & Megiddo (1994) et Hochbaum & Naor (1994) qui l'utilisent notamment dans l'étude d'algorithmes spécialisés pour des systèmes à deux variables par inégalité.

3.2 Chapitre IV

Les techniques actuelles de réduction d'intervalles, en programmation logique avec contraintes, sont issues d'une synthèse entre les méthodes itératives utilisées pour les domaines finis (*partial look ahead*, Van Hentenryck 1989) et les domaines continus (*bounds consistency*, Lhomme 1994). Cette synthèse, dont Prolog IV est un bel exemple (Colmerauer 1995), permet de traiter à la fois des problèmes linéaires sur les entiers et des problèmes non linéaires sur les réels (Benhamou & Older, 1997).

Pour des problèmes linéaires continus, l'usage des intervalles peut donc sembler à contre-emploi !

Les Problèmes de Satisfaction de Contraintes sur des domaines finis (CSP) sont souvent abordés avec des techniques de cohérence d'arc (Mackworth, 1977) — une forme plus sophistiquée d'approximation locale.
Jeavons *et al.* (1995) identifient un certain nombre de classes polynômiales. Leurs résultats sur les contraintes *max-closed* sont philosophiquement proches de ceux que nous énonçons pour les systèmes en nombres entiers. Ils montrent qu'une généralisation est possible, au delà du linéaire.

3.3 Chapitres V et VI

Loin d'opposer le point de vue de la satisfaisabilité (typiquement, celui de l'Intelligence artificielle) au point de vue de l'optimisation (celui de la Recherche opérationnelle), les travaux de Williams puis de Hooker (Chandru & Hooker, 1999) autour du problème SAT montrent l'intérêt d'un enrichissement mutuel entre méthodes logiques et méthodes numériques. Nous avons tenté d'œuvrer dans cet esprit.

Les vues croisées du projet BAHIA (1992-95) sur les différents formalismes du calcul propositionnel offrent un état de l'art et une synthèse des plus clairs.

La complétude de la «propagation», sur des domaines bornés, pour les systèmes à deux variables que l'on retrouve dans les problèmes d'ordonnancement, fait un peu partie du folklore de la programmation par contraintes (cf. par exemple Laburthe, 1998). Dechter *et al.* (1991) notent que l'élimination de Fourier (qui est complète sur des domaines non bornés) permet de réaliser de façon polynômiale la *cohérence de chemin.*

Toujours sur la résolution de contraintes de tension, l'article de Girodias *et al.* (1995) établit, pour les besoins d'une application industrielle, une équivalence entre l'algorithme de réduction d'intervalles de BNR-Prolog et un algorithme matriciel «lent» de calcul des plus courtes distances entre tous les couples de sommets.

3.4 Chapitre VII

La référence de base pour ce chapitre est l'article de Bérard & Fribourg (1999) dont nous présentons une implantation des méthodes (avec plusieurs variantes, en fonction du solveur). Toutes les contraintes exprimées par le modèle logique ne pouvant être gérées en même temps, nous dissocions les disjonctions (liées à l'aspect non-déterministe du choix des transitions) des conjonctions (liées au déclenchement d'une transition). D'où la combinaison contraintes + recherche.

On notera que les spécifications, par fermeture transitive, de la recherche des chemins (pour la vérification de l'accessibilité dans un automate à nombre fini d'états) seraient directement exécutables dans un langage de programmation logique comme XSB-Prolog, où la machinerie fait appel à des techniques de tabulation (D.S. Warren, 1992).

L'article de Delzanno & Podelski (1999) décrit de façon détaillée les liens, somme toute assez naturels, entre la programmation avec contraintes et le Model Checking symbolique. Alors que nous nous limitons, pour les besoins de nos preuves, aux seules propriétés d'accessiblité, toute la logique temporelle CTL y est abordée, avec notamment une reformulation des propriétés de vivacité.

Enfin, d'autres techniques d'abstraction seraient susceptibles d'accélérer les calculs en vérification de programmes. Celle de Halbwachs (1993) qui utilise une analyse des relations linéaires nous apparaît séduisante. Mais nous n'avons pas eu besoin d'y recourir pour les exemples étudiés.

4 Perspectives

Brièvement, elles sont doubles.

Du côté des langages de programmation par contraintes, la procédure de Fourier, qui nous a servi de guide, pourrait fournir un algorithme de choix pour «communiquer» avec un solveur d'intervalles et l'aider formellement dans une résolution. Les deux parlent en effet un langage commun : celui des projections. Sa très raisonnable complexité pour les contraintes de tensions (qui épargnerait bien des problèmes de convergence lente) et sa possible généralisation aux systèmes monotones à deux variables en feraient un candidat idéal pour une «contrainte globale».

Du côté de la vérification de programmes, les *SAT solvers* et autres *DBM model checkers* semblent avoir le vent en poupe. Nous avons passé en revue des principes et méthodes qui entrent directement dans ces technologies — et permettent d'en faire un bon usage.

Au delà, c'est au programmeur d'imaginer, face à un problème combinatoire, une solution pratique, un modèle bien construit. Pourquoi pas ... une nouvelle forme de relaxation ?

Bibliographie

A

[1] A. ARNOLD, J. BEAUQUIER, B. BÉRARD & B. ROZOY
Programmes parallèles, modèles et validation. Armand Colin, 1992.

[2] S. ACHMANOV
Programmation linéaire. Editions Mir, 1984.

B

[3] A. BABICHEV, O. KADYROVA, T. KASHEVAROVA & A. LESHCHENKO
Unicalc, a novel approach to solving systems of algebraic equations. *Interval Computations*, Vol.2, 29-47, 1993.

[4] BAHIA (nom collectif)
Etude comparative de trois formalismes en calcul propositionnel. In *Actes des 4èmes journées nationales PRC-GDR IA*, 239-314, Technea, 1992.

[5] BAHIA (nom collectif)
Etude comparative de trois formalismes en calcul propositionnel. In *Actes des 5èmes journées nationales PRC-GDR IA*, 126-158, Technea, 1995.

[6] R. BAPAT & T. RAGHAVAN
Nonnegative matrices and applications. Encyclopedia of Mathematics, N° 64, Cambridge, 1997

[7] R. BELLMAN
Dynamic Programming. Princeton University Press, 1957.

[8] F. BENHAMOU & W. OLDER
Applying interval arithmetic to real, integer and boolean constraintes. *Journal of Logic Programming*, Vol. 32, N° 1, 1-24, 1997.

[9] B. BÉRARD, M. BIDOIT, F. LAROUSSINIE, A. PETIT & PH. SCHNOEBELEN
Vérification de logiciels: techniques et outils du model-checking. Vuibert, 1999.

[10] B. BÉRARD & L. FRIBOURG
Reachability Analysis of (timed) Petri Nets using Real Arithmetic. In *Proc. 10th Int. Conf. on Concurrency Theory*, LNCS 1664, 178-193, Springer, 1999.

[11] C. BERGE
Graphes et hypergraphes. Dunod, 1970.

[12] A. BERMAN & R.J. PLEMMONS
Nonnegative matrices in the mathematical sciences. Academic Press, 1979.

[13] W. BLEDSOE
The sup-inf method in Presburger arithmetic. *Memo ATP-18*, Mathematics Department, University of Texas, 1974.

[14] A. BOCKMAYR, N. PISARUK & A. AGGOUN
Network flow problems in Constraint programming. In *Proc. CP'2001*, to appear.

[15] A. BOCKMAYR & V. WEISPFENNING
Solving numerical constraints. In *Handbook of Automated Reasoning* (Robinson & Voronkov Eds.), 751-840, Elsevier, 2001.

[16] B. BOIGELOT & P. WOLPER
Symbolic verification with periodic sets. In *Proc. CAV'94*, LNCS 818, 55-67, Springer, 1994.

[17] G. BOOLE
An Investigation of the Laws of Thought, on which are founded the mathematical theories of Logic and Probabilities, 1854.

[18] A. BREARLEY, G. MITRA & H.P. WILLIAMS
Analysis of mathematical programming problems prior to applying the simplex algorithm. *Mathematical Programming*, Vol. 8, 54-83, 1975.

C

[19] R. CHANDRASEKARAN
Integer programming problems for which a simple rounding type of algorithm works. In *Progress in Combinatorial Optimization* (Pulleblank ed.), 101-106, Academic Press, 1984.

[20] V. CHANDRU & J. HOOKER
Optimization methods for Logical inference. Wiley, 1999.

[21] V. CHVÁTAL
Linear Programming. Freeman, 1983.

[22] P. CIARLET
Introduction à l'analyse numérique matricielle et à l'optimisation. Masson, 1982.

[23] E. CLARKE, O. GRUMBERG, D. PELED
Model Checking. MIT Press, 1999.

[24] J. CLEARY
Logical arithmetic. *Future Generation Computing Systems*, Vol. 2, N° 2, 125-149, 1987.

[25] PH. CODOGNET & D. DIAZ
Local propagation methods for solving Boolean constraints in Constraint Logic Programming. *Journal of Automated Reasoning*, Vol. 17, N° 1, 97-128, 1996.

[26] E. COHEN & N. MEGIDDO
Improved algorithms for linear inequalities with two variables per inequality. *SIAM Journal on Computing*, Vol. 23, N° 6, 1313-1347, 1994.

[27] A. COLMERAUER
An introduction to Prolog III. *Communications of the ACM*, Vol. 33, N° 7, 69-90, 1990.

[28] A. COLMERAUER
Spécifications de Prolog IV. *Projet ESPRIT Prince 5246*, WP7 / R 25, 1995.

[29] A. COLMERAUER
Espaces d'approximation. *JFPLC*, Paris, 2001.

[30] A. COURNOT
Sur le Calcul des conditions d'inégalité, annoncé par M. Fourier. *Bulletin des Sciences Mathématiques, Astronomiques, Physiques et Chimiques*, Tome VI, 1-8, 1826.

[31] S. COOK
The complexity of theorem-proving procedures.In *Proc. 3rd ACM Symposium on Theory of Computing*, 151-158, 1971.

[32] D. COOPER
Theorem proving in arithmetic without multiplication. In *Machine Intelligence 7* (Meltzer & Michie eds.), 43-59, Elsevier, 1972.

[33] R. COTTLE & A. VEINOTT
Polyhedral sets having a least element. *Mathematical Programming*, Vol. 3, 238-249, 1972.

D

[34] G. DANTZIG
Linear Programming and Extensions. Princeton University Press, 1963.

[35] G. DANTZIG & C. EAVES
Fourier-Motzkin Elimination and its Dual. *Journal of Combinatorial Theory*, Series A, Vol. 14, 288-297, 1973.

[36] M. DAVIS, G. LOGEMANN & D. LOVELAND
A Machine Program for Theorem Proving. *Communications of the ACM*, Vol. 5, N° 7, 1962.

[37] M. DAVIS & H. PUTNAM
A Computing Procedure for Quantification Theory. *Journal of the ACM*, Vol. 7, N° 3, 201-215, 1960.

[38] R. DECHTER, I. MEIRI & J. PEARL
Temporal constraint networks. *Artificial Intelligence*, Vol. 49, N° 1-3, 61-95, 1991.

[39] G. DELZANNO & A. PODELSKI
Constraint-based Deductive Model Checking. *International Journal on Software Tools for Technology Transfer*, to appear.

[40] G. DELZANNO & A. PODELSKI
Model Checking in CLP. In *Proc. TACAS'99*, LNCS 1579, 223-239, Springer, 1999.

[41] D. DILL
Timing Assumptions and Verification of Finite-State Concurrent Systems. In *Proc. Automatic Verification Methods for Finite State Systems*, LNCS 407, 197-212, Springer, 1989.

[42] J. DORAN & D. MICHIE
Experiments with the graph traverser program. In *Proceedings of the Royal Society of London*, Series A, Vol. 294, 235-259, 1966.

[43] W. DOWLING & J. GALLIER
Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, Vol. 1, 267-284, 1984.

F

[44] F. FAGES
Programmation logique par contraintes. Ellipses, 1996.

[45] G. FARKAS
A Fourier-féle mechanikai elv alkamazásai. *Mathematikai és Természettudományi Értesitö*, Vol. 12, 457-472, 1894.

[46] A. FINKEL
The minimal coverability graph for Petri Nets. *Advances in Petri Nets*, LNCS 674, 210-243, Springer, 1993.

[47] R. FLOYD
Algorithm 97: shortest path. *Communications of the ACM*, Vol. 5, 345, 1962.

[48] L. FORD
Network flow theory. The RAND Corporation, 1956.

[49] J. FOURIER
Analyse des travaux de l'Académie pendant l'année 1823, p. xxix et suivantes
Analyse des travaux de l'Académie pendant l'année 1824, p. xlvii-l.
Histoire de l'Académie royale des Sciences et de l'Institut de France, t. VII, Didot, 1827.

[50] J. FOURIER
Solution d'une question particulière du calcul des inégalités. *Nouveau Bulletin des Sciences par la Société philomathique de Paris*, p. 99, 1826.

[51] J. FOURIER
Analyse des équations déterminées. Didot, 1830.

[52] L. FRIBOURG & M. VELOSO PEIXOTO
Automates concurrents à contraintes. *Technique et Science Informatiques*, Vol 13, N° 6, 837-866, 1994.

[53] L. FRIBOURG & H. OLSÉN
A Decompositional Approach for Computing Least Fixed-Points of Datalog Programs with Z-Counters. *Constraints*, Vol. 2, 305-335, 1997.

G

[54] G. GALLO & G. URBANI
Algorithms for testing the satisfiability of propositional formulae. *Journal of Logic Programming*, Vol. 7, 45-61, 1989.

[55] R. GÉNISSON
A propos d'énumération et de polynômialité sur le problème SAT et les problèmes de satisfaction de contraintes. Thèse, Université d'Aix-Marseille, 1996.

[56] G. GOLUB & A. MEURANT
Résolution numérique des grands systèmes linéaires. Eyrolles, 1983.

[57] M. GONDRAN
Introduction aux Systèmes Experts. Eyrolles, 1985.

[58] M. GONDRAN & M. MINOUX
Graphes et algorithmes. Eyrolles, 1995.

[59] P. GIRODIAS, E. CERNY & W. OLDER
Solving Linear, Min and Max Constraint Systems using CLP based on Relational Interval Arithmetic. In *Proc. CP'95* (Montanari & Rossi eds.), LNCS 976, 186-203, Springer, 1995.

H

[60] N. HALBWACHS
Delay analysis in synchronous programs. In *Proc. 5th Computer-Aided Verification*, LNCS 697, 333-346, Springer, 1993.

[61] T. HENZINGER, P-H. HO & H. WONG-TOI
A User Guide to HyTech. In *Proc. TACAS'95*, LNCS 1019, 41-71, 1995.

[62] D. HOCHBAUM & J. NAOR
Simple and fast algorithms for linear and integer programs with two variables per inequality. *SIAM Journal on Computing*, Vol. 23, N° 6, 1179-1192, 1994.

[63] W. HODGES
Logic. Penguin Books, 1977.

[64] J. HOFFMAN & J.B. KRUSKAL
Integral boundary points of convex polyhedra. In *Linear Inequalities and Related Systems* (Kuhn & Tucker eds.), Princeton University Press, 223-246, 1956.

[65] CH. HOLZBAUR
Specification of Constraint Based Inference Mechanisms through Extended Unification. Thèse, Technischen Universität Wien, 1990.

[66] J. HOOKER
Input Proofs and Rank One Cutting Planes. *ORSA Journal on Computing*, Vol.1, N° 3, 137-145, 1989.

[67] J. HOOKER
Logic-based methods for Optimization. Wiley, 2000.

[68] A. HORN
On Sentences which are True of Direct Unions of Algebras. *Journal of Symbolic Logic*, Vol. 16, 14-21, 1951.

I

[69] J-L. IMBERT
Les contraintes linéaires sur les nombres réels dans le cadre de la Programmation en logique avec contraintes. Recueil de Travaux pour Habilitation, Université d'Aix-Marseille, 1993.

J

[70] J. JAFFAR & J-L. LASSEZ
Constraint Logic Programming. In *Proc. 14th Symposium on Principles Of Programming Languages*, 111-119, 1987.

[71] J. JAFFAR & M. MAHER
Constraint Logic Programming: a Survey. *Journal of Logic Programming*, Vol.19-20, 503-581, 1994.

[72] C. JARD & M. RAYNAL
De la nécessité de spécifier des propriétés pour la vérification des algorithmes distribués. Rapport de Recherche INRIA - IRISA, N° 590, 1986.

[73] P. JEAVONS, D. COHEN & M. GYSSENS
A Unifying Framework for Tractable Constraints. In *Proc. CP'95* (Montanari & Rossi eds.), LNCS 976, 276-291, Springer, 1995.

[74] P. JEAVONS & M. COOPER
Tractable Constraints on Ordered Domains. *Artificial Intelligence*, Vol. 79, 327-339, 1996.

[75] R. JEROSLOW & J. WANG
Dynamic programming, integral polyhedra and Horn clause knowledge base. *ORSA Journal on Computing*, Vol. 4, 7-19, 1989.

[76] R. JEROSLOW, K. MARTIN, R. RARDIN & J. WANG
Gainfree Leontief Substitution Flow Problems. *Mathematical Programming*, Vol. 57, 375-414, 1992.

K

[77] L. KANTOROVITCH
Calcul économique et utilisation des ressources (traduction française). Dunod, 1962.

[78] R. KARP & R. MILLER
Parallel program schemata. *Journal of Computer and System Sciences*, Vol. 3, 1969.

[79] A. KAUFMANN & A. HENRY-LABORDÈRE
Méthodes et modèles de la recherche opérationnelle. Dunod, 1974.

[80] L. KHACHIYAN
A polynomial algorithm in linear programming. In *Sov. Math. Dokl.*, Vol. 20, N° 1, 191-194, 1979.

[81] R. KOWALSKI
Logic for Problem Solving. North-Holland, 1979.

[82] H. KUHN
Solvability and consistency for linear equations and inequalities. *The American Mathematical Monthly*, Vol. 63, 217-232, 1956.

L

[83] F. LABURTHE
Contraintes et Algorithmes en Optimisation Combinatoire. Thèse, Université de Paris VII, 1998.

[84] J-L. LAURIÈRE
Intelligence Artificielle : résolution de problèmes par l'Homme et la machine. Eyrolles, 1986.

[85] K. LARSEN, F. LARSSON, P. PETTERSSON & W. YI
Efficient Verification of Real-Time Systems: compact data structure and state-space reduction. In *Proc. 18th IEEE Real Time Systems Symposium*, 14-24, 1997.

[86] J-L. LASSEZ, T. HUYN & K. MCALOON
Simplification and Elimination of Redundant Arithmetic Constraints. In *Proc. North American Conf. on Logic Programming* (Lusk & Overbeek eds.), MIT Press, 37-51, 1989.

[87] J-L. LASSEZ & M. MAHER
On Fourier's Algorithm for Linear Arithmetic Constraints. *Journal of Automated Reasoning*, Vol. 9, 373-379, 1992.

[88] K. LAUTENBACH
Linear Algebraic Techniques for Place/Transition Nets. In *Petri Nets: central models and their properties*, LNCS 254, Springer, 1986.

[89] W. LEONTIEF
Essais d'Economiques. Calman-Levy, 1974.

[90] O. LHOMME
Contribution à la résolution de contraintes sur les réels par propagation d'intervalles. Thèse, Université de Nice - Sophia Antipolis, 1994.

[91] S. LITVINTCHOUK & V. PRATT
A proof checker for dynamic logic. In *Proc. 5th IJCAI*, 552-8, 1977.

[92] J. LUKASIEWICZ
Contribution à l'histoire de la logique des propositions. *Przeglad Filozoficzny*, 1934. Traduction française de Jean Largeault dans *Logique Mathématique - textes*, Armand Colin, 1972.

M

[93] A. MACKWORTH
Consistency in Networks of Relations. Artificial Intelligence, Vol. 8, N° 1, 99-118, 1977.

[94] J-F. MAURRAS
Programmation Linéaire, Complexité, Séparation et Optimisation. Polycopié de cours, Université d'Aix-Marseille, 1999.

[95] E. MAYR
An algorithm for the general Petri net reachability problem. *SIAM Journal on Computing*, Vol. 13, N° 3, 1984.

[96] K. MCALOON & C. TRETKOFF
Optimization and Computational Logic. Wiley, 1996.

[97] G. MINTY
Monotone Networks. In *Proceedings of the Royal Society*, Series A, Vol. 257, N° 1289, 194-212, 1960.

[98] U. MONTANARI
Networks of Constraints; fundamental properties and applications to picture processing. *Information Science*, Vol.7, N° 2, 95-132, 1974.

[99] U. MONTANARI & F. ROSSI
Finite Domain Constraint Solving and Constraint Logic Programming. In *Constraint Logic Programming: Selected Research* (Benhamou & Colmerauer eds.), MIT Press, 201-221, 1993.

[100] R. MOORE
Interval Analysis. Prentice Hall, 1966.

[101] T. MOTZKIN
Beiträge zur Theorie der linearen Ungleichungen. Dissertation, Universität Zürich, 1936
Contributions to the theory of linear inequalities (traduction anglaise), 1952.

N

[102] G. NARBONI
From Prolog III to Prolog IV: The Logic of Constraint Programming Revisited. *Constraints*, Vol 4, N° 4, 313-335, 1999.

[103] H. NAVIER
Sur le Calcul des conditions d'inégalité. *Bulletin de la Société philomathique de Paris*, 66-68, 1825.

[104] S. N'DONG
*Un système approché de résolution numérique: définition et méthodes.*Thèse, Université d'Aix-Marseille, 1999.

[105] G. NEMHAUSER & L. WOLSEY
Integer and Combinatorial Optimization. Wiley, 1988.

[106] A. NERODE & R. SHORE
Logic for Applications. Springer, 1997.

O

[107] W. OLDER & A. VELLINO
Extending Prolog with Constraint Arithmetic on Real Intervals. *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, 1990.

[108] H. OLSÉN
Automatic Verification of Petri Nets in a CLP Framework. PhD Thesis, Linköping University, 1997.

[109] J. OSTROFF
Temporal logic for real-time systems. Research Studies Press, 1989.

P

[110] C. PETRI
Kommunication mit automaten. Institute für Instrumentelle Mathematik, Schriften des IMM, N° 2, 1962.

[111] H. POINCARÉ
Second complément à l'Analysis situs. *Proc. of the London Mathematical Society*, Vol. 32, 277-308, 1900. Œuvres de Henri Poincaré, Tome VI, 338-370, Gauthier Villars, 1953.

[112] M. PRESBURGER
Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zählen in welchem die Addition als einzige Operation hervortritt. *Sprawozdanie z I Kongresu Matematykow Krajow Slowcanskich Warsawa*, 92-101, 1929.

[113] PROLOGIA
Le Manuel de Prolog IV, 1996.

[114] W. PUGH
A practical integer algorithm for exact array dependence analysis. *Communications of the ACM*, Vol. 35, N° 8, 102-114, 1992.

R

[115] A. RAUZY
Polynomial restrictions of SAT: what can be done with an efficient implementation of the Davis and Putnam's procedure. In *Proc. CP'95* (Montanari & Rossi eds.), LNCS 976, 515-532, Springer, 1995.

[116] J-CH. RÉGIN & M. RUEHER
A global constraint combining a sum constraint and difference constraints. In *Proc. 6th Int. Conf. on Principles and Practice of Constraint Programming*, 2000.

[117] J.A. ROBINSON
A machine-oriented logic based on the resolution principle. *Journal of the ACM*, Vol. 12, N° 1, 23-41, 1965.

[118] J.A. ROBINSON
Automatic Deduction with Hyper-Resolution. *Inter. Jour. of Computer Math.*, Vol. 1, 227-234, 1965.

[119] ROSEAUX (nom collectif)
Exercices résolus de Recherche Opérationnelle. Masson, 1983-1995.

[120] B. ROY
Les problèmes d'ordonnancement, applications et méthodes. Dunod, 1964.

S

[121] M. SAKAROVITCH
Optimisation combinatoire. Hermann, 1984.

[122] M-G. SCUTELLA
A note on Dowling and Gallier's top-down algorithm for propositional Horn Satisfiability. *Journal of Logic Programming*, Vol. 8, 265-273, 1990.

[123] R. SHOSTACK
Deciding linear inequalities by computing loop residues. *Journal of the ACM*, Vol. 28, N° 4, 769-779, 1981.

[124] A. SHRIJVER
Theory of Linear and Integer Programming. Wiley, 1986.

[125] P. SIEGEL
Représentation et utilisation de la connaissance en calcul propositionnel. Thèse d'Etat, Université d'Aix-Marseille, 1987.

[126] H. SIMONIS
Test Generation using the Constraint Logic Programming language CHIP. In *Proc. ICLP'89* (Levi & Martelli eds.), 101-112, MIT Press, 1989.

[127] H. SIMONIS & M. DINCBAS
Propositional Calculus Problems in CHIP. In *Constraint Logic Programming: Selected Research* (Benhamou & Colmerauer eds.), MIT Press, 269-285, 1993.

[128] T. SUNAGA
Theory of an Interval Algebra and its Application to Numerical Analysis. *RAAG Memoirs*, Vol. 2, Misc. II, 547-564, 1958.

T

[129] R. TARJAN
Enumeration of the elementary circuits of a directed graph. *SIAM Journal on Computing*, Vol. 2, N° 3, 211-216, 1973.

V

[130] M. VAN EMDEN
The Logic Programming Paradigm in Numerical Computation. In *The Logic Programming Paradigm: a 25-year perspective* (Apt *et al.* eds.), Springer, 257-276, 1999.

[131] M. VAN EMDEM & R. KOWALSKI
The Semantics of Predicate Logic as Programming Language. *Journal of the ACM*, Vol. 23, N° 4, 733-743, 1976.

[132] P. VAN HENTENRYCK
Constraint Satisfaction in Logic Programming. MIT Press, 1989.

[133] P. VAN HENTENRYCK, L. MICHEL & Y. DEVILLE
Numerica: A Modeling Language for Global Optimization. MIT Press, 1997.

[134] R.S. VARGA
Matrix Iterative Analysis. Prentice Hall, 1962.

[135] A. VEINOTT
Representation of General and Polyhedral Subsemilattices and Sublattices of Product Spaces. *Linear Algebra and Its Applications*, Vol. 114-115, 681-704, 1989.

W

[136] D. WALTZ
L'intelligence artificielle. In *L'intelligence de l'informatique* (préfacé par M. Nivat), Dunod, 139-152, 1984.

[137] D.S. WARREN
Memoing for Logic Programs. *Communications of the ACM*, Vol. 35, N° 3, 1992.

[138] V. WEISPFENNING
The Complexity of Linear Problems in Fields. *Journal of Symbolic Computation*, Vol. 5, 3-27, 1988.

[139] V. WEISPFENNING
Mixed Real-Integer Linear Quantifier Elimination. In *Proc. ISSAC'99*, 129-136, 1999.

[140] H.P. WILLIAMS
Fourier-Motzkin elimination extension to integer programming problem. *Journal of Combinatorial Theory*, Vol. 21, N° 1, 118-123, 1976.

[141] H.P. WILLIAMS
The Elimination of Integer Variables. *J. Operational Research Society*, Vol. 43, N° , 387-393, 1992.

[142] H.P. WILLIAMS
Logic applied to integer programming and integer programming applied to logic. *European Journal of Operational Research*, Vol. 81, 605-616, 1995.

Table des matières

Remerciements i
Avertissement ii
Résumé v

CHAPITRE I Introduction
Cadre général, conventions et notations 1

1 Conventions 3
2 Notations 5
3 Plan de la thèse 6

Première partie 11

CHAPITRE II Aux origines du calcul des inégalités
Le procédé d'élimination de Fourier 13

1 L'Analyse des inégalités 15
2 Elimination de Fourier dans le cas linéaire 15
2.1 Notations 15
2.2 Elimination d'une variable 16
2.3 Propriété de l'élimination 17
2.4 Interprétation géométrique 19
2.5 Inégalités redondantes 19
2.6 Procédure de décision 20
2.7 Interprétation logique 21
2.8 Complexité 21
3 Dualité, optimisation et consécution 22
3.1 Théorème de la dualité 22
3.2 Application à la preuve de consécution 24
4 Elimination de Fourier de variables entières 25
4.1 Satisfaisabilité sur les entiers 25
4.2 Incomplétude de l'élimination 25
4.3 Cause de l'incomplétude 27
4.4 Conditions pour y remédier 27
4.5 Cas d'élimination licite 28
5 Une procédure alternative pour l'élimination 29
5.1 La méthode des points-tests 29
5.2 Complexité comparée 30
5.3 Un exemple 31
5.3.1 Résolution par la méthode de Fourier 31
5.3.2 Résolution par la méthode des points-tests 32

CHAPITRE III Des familles de polyèdres ayant un plus petit élément
Propriétés remarquables des inégalités monotones 35

1 Produits et treillis vectoriels 36
2 Systèmes monotones 37
3 Propriétés générales des systèmes monotones 39
3.1 Monotonie 39
3.2 Stabilité par intersection 39
3.3 Existence d'un extrêmum global 40
4 Compléments sur les systèmes à variables réelles 41
4.1 Stabilité par projection 41
4.2 Caractérisation des polyèdres ayant un plus petit élément 42
4.3 Références terminologiques 44
5 Systèmes 1-monotones 44
5.1 Stabilité par projection 45
5.2 Intégrité des extrêmums globaux 46
5.3 Stabilité par troncature 47
5.4 Caractérisations 47
6 Compléments sur les systèmes à variables entières 49
6.1 Optimisation linéaire sous contraintes 1-monotones 49
6.2 Résolution générale par un algorithme de troncatures 49

CHAPITRE IV Résolution approchée des systèmes monotones
Techniques de réduction d'intervalles 53

1 Rappels sur les intervalles 54
2 Discrétisation de la droite et de l'espace 56
3 Approximation d'un ensemble de points 57
4 Mécanisme général de calcul 59
4.1 Résolution approchée par réduction d'intervalles 59
4.2 Algorithme du plus grand point fixe 60
5 Traitement des inégalités linéaires 62
5.1 Réduction d'une inégalité linéaire 62
5.2 Pré-traitements particuliers 64
6 Application aux systèmes linéaires 64
6.1 Constat 64
6.2 Approximation d'une fonction linéaire monotone 65
6.3 Approximation d'une relation linéaire monotone 68
6.3.1 Contrainte de forme (1) 68
6.3.2 Contrainte de forme (2) 68
7 Résolution des systèmes monotones 69
7.1 Condition suffisante de satisfaisabilité 69
7.2 Qualité de l'approximation 70
7.3 Cas des systèmes 1-monotones 73
7.4 Résolution sur les entiers 76
8 Complexité 76

Deuxième partie 79

CHAPITRE V De Fourier à Davis et Putnam
Satisfaisabilité en logique propositionnelle 81

1 Satisfaisabilité d'une formule 83
2 Formes normales 83
2.1 Forme normale disjonctive 83
2.2 Forme normale conjonctive 84
3 Représentation des connaissances 85
4 Traduction dans le calcul des inégalités 86
4.1 Transcription d'un système de clauses 86
4.2 Transcription d'un système de Horn 87
5 Résolution d'un système de Horn 88
5.1 Complétude de la relaxation linéaire 88
5.2 Complétude de la relaxation dans l'approximation des intervalles 88
5.3 Application à la vérification de cohérence de règles 89
6 De Fourier à la résolution unitaire 92
6.1 Simplifications liminaires 92
6.2 Algorithme de résolution unitaire positive 92
6.3 Complexité 94
6.4 Interprétation graphique 94
6.5 Transparence des résultats 96
6.6 Calcul des projections exactes 97
6.7 Algorithme de résolution unitaire 97
7 Résolution dans le cas général 99
8 De la résolution à l'algorithme de Davis & Putnam 101
8.1 Version originale 101
8.2 Version énumérative 102
9 Autres méthodes énumératives 102
9.1 Relaxation reposant sur un système de Horn 102
9.1.1 Premier modèle de relaxation 103
9.1.2 Second modèle de relaxation 103
9.2 Résolution locale en programmation par contraintes 104

CHAPITRE VI Cheminements et ordonnancements
Optimisation combinatoire sur les graphes 107

1 Modélisation par les graphes 108
2 Chemins et plus courts chemins sur un graphe 109
2.1 Rappels terminologiques 110
2.2 Problèmes de plus courts chemins 111
3 Courants minimaux et plus courtes distances 112
3.1 Tableaux d'incidence 112
3.2 Formulation en nombres entiers 112
3.3 Unimodularité et relaxation sur les réels 115
3.4 Interprétation par les courants 116
4 Tensions maximales et plus courtes distances 117
4.1 Un calcul analogique 117
4.2 Formulation en tensions 117

4.3 Conditions de satisfaisabilité 118
4.4 Monotonie et optimalité 119
4.5 Systèmes à variables entières 121
5 Identification des plus courts chemins 121
6 Algorithmes de résolution 123
6.1 Remarques historiques 123
6.2 Réduction d'intervalles et algorithme de Ford 123
6.3 Elimination de Fourier et algorithmes matriciels 128
7 Problèmes d'ordonnancement 132
8 En résumé 133

Troisième partie 135

CHAPITRE VII Vérification de propriétés d'automates synchronisés
Analyse de l'accessibilité dans les Réseaux de Petri 137

1 Philosophie du Model Checking 138
2 Automates et propriétés de sûreté 139
3 Réseaux de Petri 140
3.1 Définition des places et transitions 140
3.2 Programme à compteurs équivalent 142
4 Preuves d'accessibilité 145
4.1 Limitations théoriques 145
4.2 Incomplétude de la stratégie opérationnelle de Prolog 145
4.3 Calcul itératif des états accessibles 146
5 Gestion des règles 147
5.1 Moteur d'interprétation 147
5.2 Compilation 149
6 Gestion des contraintes 151
6.1 Calcul symbolique des états accessibles 151
6.2 Programmation en Prolog III 152
6.3 Programmation en Prolog IV 155
6.3.1 Test de non vacuité 155
6.3.2 Opération de projection 156
6.3.3 Test d'inclusion 158
6.3.4 Application 160
7 Extensions de la méthode 163
8 En résumé 168

CHAPITRE VIII Conclusion
Bilan, références, perspectives 171

1 Un dernier exemple 172
2 Bilan 174
3 Références et travaux apparentés 178
3.1 Chapitre III 178
3.2 Chapitre IV 178
3.3 Chapitres V et VI 179
3.4 Chapitre VII 179
4 Perspectives 180

Bibliographie 181
Table des matières 191