La programmation logique avec contraintes revisitée en termes d'arbres de preuve et de squelettes

Michel Bergère, Gérard Ferrand, François Le Berre,
Bernard Malfon, Alexandre Tessier
Université d'Orléans
Département d'Informatique, LIFO
BP 6759, F-45067 Orléans cedex 2 (France).
Email : {bergere, ferrand, flb, malfon, tessier}@lifo.univ-orleans.fr
Tel : +33 38 41 70 10, Fax : +33 38 41 71 37

Février 1995
RR-LIFO-95-6

Résumé

Nous redéfinissons les bases de la Pogrammation Logique avec Contraintes en utilisant les Arbres de Preuve et les Squelettes, qui permettent de donner à la notion de réponse une définition intrinsèque indépendante de toute règle de calcul. Nous étudions les réponses données par un programme contraint en paramétrant la validité de celles-ci par un Critère de Rejet dont le rôle est d'éliminer celles dont la contrainte n'est jamais satisfaite.

Nous retrouvons, en les prouvant complètement, les résultats classiques du domaine, et ajoutons quelques résultats nouveaux, notamment une caractérisation des branches non échec dans les arbres de recherche équitables.

Mots-clés : Programmation Logique, Contraintes, Sémantique, Arbres de Preuve, Squelettes, Atomes contraints.

Table des matières

1 Introduction **3**

2 De la syntaxe à la notion de réponse **4**
2.1 Interprétation des contraintes 5
2.2 Système de règles sur les $\mathcal{D}$-atomes 8
2.3 Généralités sur les systèmes inductifs 8
2.4 Application aux $\mathcal{D}$-règles 10
2.5 Complété du programme 11
2.6 Squelettes 16
2.7 Contraintes associées à un squelette 18
2.8 Questions, réponses 21
2.9 Contraintes réponses, solutions 22
2.10 Correspondance entre squelettes réponses et arbres de preuve 24
2.11 Réponses générales 26

3 Sémantique opérationnelle **28**
3.1 Réponses 28
3.2 Recherche des réponses 28
3.3 Construction descendante 30
3.4 Résolution SLD 31
3.5 Règle de sélection 34
3.6 Arbre de recherche SLD 36

4 Interprétation en logique **43**
4.1 Réponses selon $\mathcal{D}$ 43
4.2 Réponses selon une théorie $\mathcal{T}$ 48
4.2.1 Théorie associée à une pré-interprétation 49
4.2.2 Théorie associée à un algorithme 50
4.2.3 Cas particulier de la programmation logique 54
4.2.4 Branches non échec 56
4.3 Réponses selon un critère CR en relation avec une théorie $\mathcal{T}$ 57

5 Atomes contraints **59**
5.1 Instances, ensemble succès 59
5.2 Programme normalisé 60
5.3 Règles d'induction sur les atomes contraints 62
5.4 Ensemble succès et arbres de preuve contraints 63
5.5 Atomes contraints et conséquences logiques 68
5.6 Équivalence entre le programme et l'ensemble succès 69
5.7 Ensemble d'échec fini 70

1 Introduction

L'objectif de ce travail est de redéfinir les bases de la programmation logique avec contraintes selon [12] en tirant tout le bénéfice des notions d'*arbre de preuve* et de *squelette* d'arbre de preuve développées dans [8].

Il ne s'agit pas ici de faire un tour d'horizon du domaine comme dans [4, 5], et [13].

L'utilisation des arbres de preuves et de squelettes présente l'avantage important de donner une définition intrinsèque à la notion de réponse fournie par le programme : c'est un squelette fini complet ayant pour racine le but question, accepté d'une certaine manière. Ainsi est clairement expliquée l'indépendance de nombreux résultats par rapport à la règle de sélection d'atome.

Grâce à son degré de généralité, notre formalisme peut s'appliquer aussi bien à Prolog III et au futur Prolog IV, qu'à CLP.

Nous retrouvons, en les prouvant complètement et souvent de manière plus rapide, les résultats classiques de la programmation logique avec contraintes. Nous ajoutons quelques résultats nouveaux, dont une caractérisation simple des branches non échec dans les arbres de recherche équitables.

La définition de la notion de réponse comme squelette fini complet ayant pour racine la question $\leftarrow b$ permet d'élargir le champ d'investigation : tout squelette fini complet est une bonne réponse en ce sens que, si r est la contrainte associée, alors

$$P \models r \rightarrow b$$

C'est ce que nous appelons une réponse généralisée. Mais parmi les implications $r \rightarrow b$ conséquences logiques du programme, certaines sont trivialement vraies car r est toujours fausse (insatisfiable). Du point de vue théorique, elles ne présentent pas d'intérêt. Du point de vue opérationnel, il faut éliminer les explorations inutiles qui risqueraient d'engager dans des calculs infinis.

Il apparaît que la notion essentielle est le critère de rejet qui a pour rôle d'éliminer les $r \rightarrow b$ pour lesquelles on sait que r est insatisfiable. Ce critère peut être déduit d'une interprétation sous-entendue D , ou d'une théorie T qui n'a pas de raison particulière d'être satisfaction-complète (comme le supposent la plupart des auteurs), ou de l'algorithme de satisfiabilité des contraintes d'un système réel. On peut étudier le comportement opérationnel d'un programme en le paramétrant par ce critère de rejet.

La section 2 donne les définitions essentielles : programme contraint, arbre de preuve dans une pré-interprétation D, squelette, réponse, réponse généralisée.

La section 3 étudie la sémantique opérationnelle paramétrée par le critère de rejet : dérivation SLD, arbre de recherche SLD, caractérisation des branches succès et des branches non échec.

La section 4 étudie les interprétations en logique des résultats obtenus sur les réponses dans trois situations : lorsque le critère de rejet correspond à la satisfiabilité dans D, losqu'il correpond à une théorie T, lorsqu'il est en relation avec une théorie T.

La section 5 étudie les atomes contraints introduits dans [12, 16, 13, 10]. Nous démontrons dans un contexte plus général que l'ensemble succès est le plus petit point fixe d'un opérateur sur les ensembles d'atomes contraints, puis qu'il se comporte comme un programme contraint équivalent au programme initial du point de vue des contraintes réponses.

2 De la syntaxe à la notion de réponse

Les notations employées ici sont très proches de celles de [13].

Le langage du programme est défini à partir de quatre ensembles :

- un ensemble infini de variables V,
- un ensemble de symboles de fonctions Σ,
- un ensemble de symboles de prédicats de contraintes Π_C (ce sont les prédicats prédéfinis),
- un ensemble de symboles de prédicats définis par programme Π_P encore appelés prédicats de programme.

On note :

- $TERM(V, \Sigma)$, ou plus simplement $TERM$ s'il n'y a pas de confusion, l'ensemble des termes construits sur V et Σ,
- $\mathcal{L}(V, \Sigma, \Pi_C)$ le langage du premier ordre construit sur V, Σ et Π_C. On suppose qu'il contient, au même titre que les connecteurs logiques,
  - les constantes logiques $vrai$ et $faux$ qui seront toujours interprétées comme les éléments $vrai$ et $faux$ de l'ensemble des valeurs de vérité,
  - le symbole d'égalité $=$ qui sera toujours interprété comme l'égalité.

  Une contrainte est une formule de $\mathcal{L}(V, \Sigma, \Pi_C)$.
- $ATOM_C$ l'ensemble des formules atomiques de $\mathcal{L}(V, \Sigma, \Pi_C)$.

  Nous serons éventuellement amenés à apporter des restrictions sur les formules de $\mathcal{L}(V, \Sigma, \Pi_C)$ acceptables comme contraintes. Mais dans tous les cas le langage des contraintes comprendra $ATOM_C$ et sera fermé par conjonction et quantification existentielle. Nous utiliserons des collections (ensembles) de contraintes, finies ou infinies. Toute collection finie de contraintes est assimilable à une contrainte unique conjonction des contraintes de la collection.
- $\mathcal{L}(V, \Sigma, \Pi_P)$ le langage du premier ordre construit sur V, Σ, Π_P,
- $ATOM_P$ l'ensemble des atomes : formules atomiques de $\mathcal{L}(V, \Sigma, \Pi_P)$.

  Une *clause* est un $(n+2)$-uplet $(0 \leq n) < A, c, B_1, B_2, \ldots, B_n >$ où $A, B_1, \ldots, B_n$ sont des atomes éléments de $ATOM_P$, c est une contrainte. Pour plus de commodité nous la noterons plutôt $A \leftarrow c \ \Box \ B_1, B_2, \ldots, B_n$.

  Etant donné une clause C de cette forme, on note $tête(C)$ le premier atome A, $corps(C)$ la partie droite $c \ \Box \ B_1, B_2, \ldots, B_n$, $contrainte(C)$ la contrainte c, $arité(C)$ le nombre n d'atomes du corps. L'arité peut être nulle, la clause se réduisant alors à $A \leftarrow c$. C'est ce qu'on appelle un *fait*. Un *but* est une clause sans tête de la forme $\leftarrow c \ \Box \ B_1, B_2, \ldots, B_n$. Les notions de contrainte et d'arité définies sur les clauses s'étendent aux buts.

  Un *programme contraint* est une collection de clauses.

Exemple 2.1 *Le programme Fib :*
$fib(0,1) \leftarrow vrai$
$fib(1,1) \leftarrow vrai$
$fib(N, A+B) \leftarrow N > 1 \ \square \ fib(N-2, A), fib(N-1, B)$
est un programme contraint sur un langage défini par des ensembles V, Σ, Π_C, Π_P *tels que :*
V *contient* $\{A, B, N\}$,
Σ *contient* $\{0, 1, 2, +, -\}$,
Π_C *contient* $\{>\}$,
Π_P *contient* $\{fib\}$.

Notation 2.1

- $\tilde{x}$ *désigne une séquence de variables distinctes* $x_1, \ldots, x_n$,
- $\exists \tilde{x}\ c$ *représente la formule* $\exists x_1 \exists x_2 \ldots \exists x_n\ c$,
- $\tilde{\exists}\ c$ *est la clôture existentielle de* c, *soit* $\exists \tilde{x}\ c$ *où* $\tilde{x}$ *est l'ensemble des variables libres de* c,
- $\exists_{-\tilde{x}}\ c$ *est la quantification existentielle de* c *sur les variables libres autres que celles de* $\tilde{x}$,
- $\tilde{\forall}\ c$ *est la clôture universelle de* c *:* $\forall x_1 \forall x_2 \ldots \forall x_n\ c$, *où* $x_1, ..., x_n$ *sont les variables libres de* c,
- *si* $\tilde{x}$ *est une séquence de variables et* $\tilde{t}$ *une séquence de termes de même longueur,* $\tilde{x} = \tilde{t}$ *représente la formule* $x_1 = t_1 \wedge x_2 = t_2 \wedge \cdots x_n = t_n$.

2.1 Interprétation des contraintes

L'écriture d'un programme contraint ne se résume pas à l'aspect syntaxique : le programmeur a toujours en tête une signification sous-entendue des contraintes qu'il utilise. Celles-ci agissent sur un domaine pré-établi (les entiers naturels, les réels, ...) et les symboles de prédicats de contraintes ont une interprétation naturelle sur ce domaine. Autrement dit, il existe, au sens défini ci-dessous, une pré-interprétation sous-entendue dite "pré-interprétation naturelle".

Une *pré-interprétation* $\mathcal{D}$ est constituée de :

- un ensemble non vide D appelé domaine,
- pour chaque symbole de fonction f appartenant à Σ, si n est son arité, une application $f_{\mathcal{D}}$ de D^n dans D. Un symbole de constante est un symbole de fonction d'arité 0 ; la pré-interprétation lui associe un élément de D,
- pour chaque symbole de prédicat de contrainte p appartenant à Π_c, si n est son arité, une relation d'arité n sur D, ou de manière équivalente une application $p_{\mathcal{D}}$ de D^n dans l'ensemble des valeurs de vérité $\{vrai, faux\}$.

Rappelons que les symboles de constantes $vrai$ et $faux$ sont obligatoirement pré-interprétées par les valeurs de vérité $vrai$ et $faux$, et que le symbole $=$ est obligatoirement pré-interprété par l'égalité sur D.

Exemple 2.2 *L'ensemble des entiers naturels $\mathbb{N}$ muni de ses opérations habituelles constitue une pré-interprétation $\mathcal{N}$ pour le programme Fib de l'exemple 2.1 si l'on associe aux symboles 0, 1, 2, +, -, > respectivement les entiers 0, 1, 2, l'addition, la soustraction (limitée à 0 : par exemple 3 - 5 donne 0) et la relation > .*

Pour l'interprétation des prédicats définis par programme, on considère des pseudo-atomes appelés $\mathcal{D}$-*atomes* formés de $(n+1)$-uplets de la forme $< p, d_1, d_2, \ldots, d_n >$ où p est un symbole de prédicat de programme élément de Π_P d'arité n et les d_i sont des éléments du domaine D. Ce $(n+1)$-uplet sera plutôt noté $p(d_1, d_2, \ldots, d_n)$.
La notion de $\mathcal{D}$-atome apparaît sous un nom un peu différent dans [8, 14, 12] et [13].

On appelle $\mathcal{D}$-*base* l'ensemble des $\mathcal{D}$-atomes. Si $\Pi_{p,n}$ est la partie de Π_P comprenant les symboles de prédicats d'arité n, on peut définir plus formellement la $\mathcal{D}$-base par $\bigcup_{n \in N} \Pi_{p,n} \times D^n$.

Étant donné une pré-interprétation $\mathcal{D}$, une valuation v associe à chaque symbole de variable une valeur dans le domaine D. C'est donc une application de V dans D.
La valuation v s'étend en une fonction $v_{\mathcal{D}}$ sur les termes, les contraintes, les atomes et les clauses :

- si t est un terme de $TERM$, $v_{\mathcal{D}}(t)$ est l'élément de D défini par :
  $v(t)$ si t est une variable,
  $f_{\mathcal{D}}(v_{\mathcal{D}}(t_1), \ldots, v_{\mathcal{D}}(t_n))$ si t est le terme $f(t_1, \ldots, t_n)$
- si c est la contrainte atomique $p(t_1, \ldots, t_n)$, $v_{\mathcal{D}}(c)$ est la valeur de vérité
  $p_{\mathcal{D}}(v_{\mathcal{D}}(t_1), \ldots, v_{\mathcal{D}}(t_n))$
  $v_{\mathcal{D}}$ s'étend à toutes les formules de $\mathcal{L}(V, \Sigma, \Pi_C)$ (donc aux contraintes) en une application de $\mathcal{L}(V, \Sigma, \Pi_C)$ dans $\{vrai, faux\}$
- si A est un atome de $ATOM_P$, $A = p(t_1, \ldots, t_n)$, $v_{\mathcal{D}}(A)$ est le $\mathcal{D}$-atome
  $p(v_{\mathcal{D}}(t_1), \ldots, v_{\mathcal{D}}(t_n))$,
- si C est une clause $A \leftarrow c \square B_1, \ldots, B_n$, $v_{\mathcal{D}}(C)$ est ce que nous appellerons une $\mathcal{D}$-*clause* : $v_{\mathcal{D}}(A) \leftarrow v_{\mathcal{D}}(c) \square v_{\mathcal{D}}(B_1), \ldots, v_{\mathcal{D}}(B_n)$ dans laquelle $v_{\mathcal{D}}(c)$ est une valeur de vérité, $v_{\mathcal{D}}(A)$, $v_{\mathcal{D}}(B_1)$, ..., $v_{\mathcal{D}}(B_n)$ sont des $\mathcal{D}$-atomes.

On dit que la contrainte c est *satisfiable* dans la pré-interprétation $\mathcal{D}$ s'il existe une valuation v dans $\mathcal{D}$ telle que $v_{\mathcal{D}}(c) = vrai$. La collection de contraintes E est satisfiable s'il existe une valuation v telle que $v_{\mathcal{D}}(c) = vrai \ \forall c \in E$. La valuation correspondante est dite *solution* de la contrainte ou de la collection de contraintes.

Étant donné une pré-interprétation $\mathcal{D}$, une interprétation J dans $\mathcal{D}$ doit associer à tout symbole de prédicat de programme p de Π_P, si n est son arité, une relation p_J

d'arité n sur D, c'est-à-dire une partie de D^n.

On appelle $\mathcal{D}-interprétation$ une partie I de la $\mathcal{D}$-base (un ensemble de $\mathcal{D}$-atomes). C'est une interprétation car elle définit pour chaque symbole de prédicat $p \in \Pi_P$ d'arité n, la relation d'arité n sur D : $\{< d_1, \ldots, d_n > \mid p(d_1, \ldots, d_n) \in I\}$. Inversement, étant donné une interprétation J dans $\mathcal{D}$, on en déduit la $\mathcal{D}$-interprétation I qui, pour chaque $p \in \Pi_P$ (soit n son arité), contient $\{p(d_1, \ldots, d_n) \in \mathcal{D} - base \mid < d_1, \ldots, d_n > \in p_J\}$. Les interprétations au sens habituel du mot sont donc en bijection avec les $\mathcal{D}$-interprétations.

Étant donné une pré-interprétation $\mathcal{D}$ et une $\mathcal{D}$-interprétation I, une valuation v s'étend en une nouvelle fonction notée v_I sur les termes, les contraintes, les atomes et les clauses :

- si t est un terme $v_I(t) = v_{\mathcal{D}}(t)$,
- si c est une contrainte, ou plus généralement une formule de $\mathcal{L}(V, \Sigma, \Pi_C)$, $v_I(c) = v_{\mathcal{D}}(c)$
- si A est un atome de $ATOM_P$, $v_I(A)$ est la valeur de vérité définie par : $v_I(A) = vrai$ ssi $v_{\mathcal{D}}(A) \in I$,
- si C est la clause $A \leftarrow c \square B_1, \ldots, B_n$, $v_I(C)$ est la valeur $vrai$ si l'implication $v_I(c) \wedge v_I(B_1) \wedge \cdots \wedge v_I(B_n) \Rightarrow v_I(A)$ est vraie, $faux$ sinon ; c'est donc $vrai$ ssi $v_I(c) = faux$, ou $v_I(B_i) = faux$ pour un certain i, ou $v_I(A) = vrai$. Pour faire le rapprochement avec la $\mathcal{D} - clause$ $v_{\mathcal{D}}(C)$, $v_I(C)$ est vraie ssi soit $v_{\mathcal{D}}(c) = faux$, soit $v_{\mathcal{D}}(c) = vrai$ et si tous les $v_{\mathcal{D}}(B_i)$ sont dans I alors $v_{\mathcal{D}}(A)$ est aussi dans I.

Une $\mathcal{D}$-interprétation I est un modèle de la clause C si, pour toute valuation v, $v_I(C) = vrai$.

Remarque 2.1 *Les $\mathcal{D}$-clauses $v_{\mathcal{D}}(C)$ que l'on peut obtenir à partir de la clause C avec n'importe quelle valuation v se divisent en deux catégories :*

- *celles pour lesquelles la contrainte $v_{\mathcal{D}}(c)$ prend la valeur $faux$; alors pour toute $\mathcal{D}$-interprétation I, $v_I(C) = vrai$,*
- *celles pour lesquelles la contrainte prend la valeur vrai. Alors, la $\mathcal{D}$-interprétation I est un modèle de la clause C si et seulement si, considérant toutes les $\mathcal{D}$-clauses $v_{\mathcal{D}}(C)$ pour lesquelles la contrainte a la valeur vrai : $a \leftarrow vrai \square b_1, ..., b_n$, si tous les b_i sont dans I alors a est aussi dans I.*

Un $\mathcal{D}$-*modèle* du programme P est une $\mathcal{D}$-interprétation modèle de toutes les clauses de P.

Exemple 2.3 $M_1 = \{fib(0,1), fib(1,1), fib(2,2), fib(3,3), fib(4,5), fib(5,8), fib(6,13), fib(7,21), fib(8,34), ...\}$ *qui est en quelque sorte le graphe de la fonction de Fibonacci, et* $M_2 = \{fib(n,p) | n, p \in \mathbb{N}\}$ *sont des $\mathcal{N}$-modèles du programme Fib (exemples 2.1 et 2.2).*

Remarquons que la $\mathcal{D}$-base tout entière est toujours un $\mathcal{D}$-modèle du programme.

2.2 Système de règles sur les $\mathcal{D}$-atomes

La notion de $\mathcal{D}$-règle introduite ici est sous-jacente dans [12, 13] et [16]. Sa mise en valeur nous permet d'utiliser les techniques habituelles sur les systèmes inductifs selon [1] comme dans [8].

On appelle *$\mathcal{D}$-règle* un $(n+1)$-uplet de $\mathcal{D}$-atomes $< a, b_1, \ldots, b_n >$, que nous noterons plutôt $a \leftarrow b_1, \ldots, b_n$. La tête de la $\mathcal{D}$-règle est le $\mathcal{D}$-atome a, son corps est constitué des $\mathcal{D}$-atomes b_1, ..., b_n. L'arité est le nombre de $\mathcal{D}$-atomes du corps. Une $\mathcal{D}$-règle d'arité 0 de la forme $a \leftarrow$ est appelée un $\mathcal{D}$-fait.

Définition 2.1 *($\mathcal{D}$-règles associées à une clause)*
Étant donné la clause C, nous avons remarqué que, parmi les $\mathcal{D}$-clauses engendrées par C avec toutes les valuations possibles, celles qui ont une contrainte vraie jouent un rôle particulier. Les $\mathcal{D}$-règles associées à C sont ces $\mathcal{D}$-clauses auxquelles on a enlevé la contrainte vraie. Plus précisément, le système de $\mathcal{D}$-règles associé à une clause C : $A \leftarrow c \,\Box\, B_1, \ldots, B_n$ est $R_{\mathcal{D}}(C) = \{v_{\mathcal{D}}(A) \leftarrow v_{\mathcal{D}}(B_1), \ldots, v_{\mathcal{D}}(B_n)$ pour les valuations v telles que $v_{\mathcal{D}}(c) = vrai\}$.

Remarque 2.2 *D'après la définition 2.1 et la remarque 2.1, une $\mathcal{D}$-interprétation I est modèle de C ssi I est close pour les $\mathcal{D}$-règles de $R_{\mathcal{D}}(C)$: si $a \leftarrow b_1, \ldots, b_n$ est dans $R_{\mathcal{D}}(C)$ et si b_1, ... , b_n sont dans I alors a est dans I.*

Définition 2.2 *($\mathcal{D}$-règles associées à un programme)*
L'ensemble de $\mathcal{D}$-règles associé au programme P est la réunion des systèmes de $\mathcal{D}$-règles associés aux différentes clauses de P : $R_{\mathcal{D}}(P) = \bigcup_{C \in P} R_{\mathcal{D}}(C)$

Exemple 2.4 *Le programme Fib pré-interprété par $\mathcal{N}$ (exemples 2.1 et 2.2) donne le système de $\mathcal{N}$-règles :*
$\{fib(0,1) \leftarrow, fib(1,1) \leftarrow\} \bigcup$
$\{fib(n, a+b) \leftarrow fib(n-2, a), fib(n-1, b) \mid n, a, b \in I\!N, n > 1\}$.
Il contient par exemple les $\mathcal{N}$-règles:
$fib(2,2) \leftarrow fib(0,1), fib(1,1)$,
$fib(2,0) \leftarrow fib(0,0), fib(1,0)$,
$fib(2,5) \leftarrow fib(0,3), fib(1,2)$.

2.3 Généralités sur les systèmes inductifs

Étant donné un ensemble E, un *système inductif* sur E est un ensemble R de règles de la forme $e \leftarrow e_1, \ldots, e_n$ où e, e_1, ... , e_n sont des éléments de E. Une règle est un fait si $n = 0$, elle se résume alors à $e \leftarrow$.

À tout système inductif on associe un *opérateur de conséquence immédiate* T application de 2^E dans 2^E défini par, si X est une partie de E :

$$T(X) = \{e \in E \mid \exists \text{ une règle } e \leftarrow e_1, \ldots, e_n \text{ dans } R \text{ et } e_1, \ldots, e_n \in X\}$$

Un *arbre de preuve* est un arbre fini étiqueté par des éléments de E tel que, pour tout nœud N, soit e son étiquette, $e_1, \ldots, e_n$ les étiquettes des nœuds fils de N, alors $e \leftarrow e_1, \ldots, e_n$ est une règle de R. Remarquons que si N est un feuille, alors $e \leftarrow$ est un

fait. Par abus de langage, le mot *racine* désignera en même temps le nœud racine de l'arbre de preuve et l'étiquette de ce nœud. On note Rac l'ensemble des racines d'arbres de preuve, c'est-à-dire l'ensemble des $e \in E$ "prouvés" par l'existence d'un arbre de preuve ayant e comme racine.

Lemme 2.1 *L'opérateur de conséquence immédiate T est croissant et continu.*

Preuve :

1. Si A est inclus dans B et si $e \in T(A)$ alors il existe une règle $e \leftarrow e_1, \ldots, e_n$ dans R telle que les e_i sont dans A. Ils sont aussi dans B, et donc $e \in T(B)$.

2. La continuité de T résulte de la finitude des règles. En effet, soit X une partie dirigée du treillis des parties de E . X est un ensemble de parties de E. Rappelons que X est dite dirigée si chacune de ses parties finies comprend sa propre borne supérieure (ici donnée par l'union). Il s'agit de montrer que $T(sup(X)) = sup(\{T(x) \mid x \in X\})$.

   - L'inclusion du second ensemble dans le premier est due à la croissance de T : pour chaque $x \in X$, x est inclus dans $sup(X)$, donc $T(x)$ est inclus dans $T(sup(X))$; il s'en suit que sup $(\{T(x) \mid x \in X\})$ est inclus dans $T(sup(X))$.

   - L'inclusion de $T(sup(X))$ dans $sup(\{T(x) \mid x \in X\})$ se prouve de la manière suivante : si $e \in T(sup(X))$, alors $e \in T(y)$ pour un certain $y \in X$. En effet, $e \in T(sup(X))$ signifie qu'il existe une règle $e \leftarrow e_1, \ldots, e_n$ telle que $e_1, \ldots, e_n \in sup(X)$. Pour chaque i, $e_i \in sup(X)$ signifie : e_i appartient à l'union des éléments de X ; donc il existe un élément x_i de X tel que $e_i \in x_i$. La famille $\{x_1, \ldots, x_n\}$ est une partie finie de X, laquelle est dirigée. Donc $\{x_1, \ldots, x_n\}$ contient sa borne supérieure y. Chaque e_i est élément de x_i donc aussi de y. Finalement e est tête d'une règle $e \leftarrow e_1, \ldots, e_n$ telle que tous les e_i sont dans y. Donc e est dans $T(y)$.

On note $T \uparrow \omega$ la partie de E définie inductivement par :
$T \uparrow 0 = \emptyset$
$T \uparrow (n+1) = T(T \uparrow n)$
$T \uparrow \omega = \bigcup_{n \in N} T \uparrow n$

Lemme 2.2 *(Plus petit point fixe)*
L'opérateur de conséquence immédiate T a un plus petit point fixe $pppf(T)$ et

$$pppf(T) = Rac = T \uparrow \omega$$

Preuve :

- $T(Rac)$ est inclus dans Rac, autrement dit Rac est un pré-point fixe de T. Soit en effet e un élément de $T(Rac)$. Alors e est tête d'une règle de R : $e \leftarrow e_1, \ldots, e_n$ et chaque $e_i \in Rac$. Donc e_i est racine d'un arbre de preuve A_i. Considérons alors l'arbre A dont la racine est étiquetée par e, et qui a n sous-arbres : $A_1, \ldots, A_n$. A est un arbre de preuve de racine e, donc $e \in Rac$.

- *Rac* est inclus dans les pré-points fixes de T : les X tels que $T(X) \subseteq X$. Ceci se prouve par récurrence sur la hauteur de l'arbre dont e est racine :

  - Si e est racine d'un arbre de hauteur 1, celui-ci est réduit à sa racine. Donc $e \leftarrow$ est une règle de R. Alors e est dans $T(\emptyset)$, donc dans $T(X)$, donc dans X.

  - Supposons que toute racine d'arbre de preuve de hauteur $\leq h$ soit dans X, et que e soit racine d'un arbre de preuve de hauteur $h+1$. Les fils de la racine dans cet arbre sont étiquetés par e_1, ... , e_n, et $e \leftarrow e_1, \ldots, e_n$ est une règle de R. Chaque e_i est racine d'un arbre de preuve de hauteur $\leq h$, donc $e_i \in X$. Donc $e \in T(X)$, donc $e \in X$.

- On en déduit que *Rac* est le plus petit pré-point fixe de T. D'après le théorème de Knaster-Tarski, c'est le plus petit point fixe de T.

- $T(T \uparrow \omega) = T(sup(T \uparrow n))$, et les $T \uparrow n$ constituent une partie dirigée. Donc $T(sup(T \uparrow n)) = sup(T \uparrow (n+1)) = sup(T \uparrow n)$. Donc $T(T \uparrow \omega) = T \uparrow \omega$, ce qui signifie que $T \uparrow \omega$ est point fixe de T. C'est le plus petit car tout point fixe M contient $\emptyset = T \uparrow 0$; donc $T(M) = M$ contient $T(T \uparrow 0) = T \uparrow 1$; M contient $T(T \uparrow 1) = T \uparrow 2$; ... donc M contient tous les $T \uparrow n$; donc M contient leur union $T \uparrow \omega$.

2.4 Application aux $\mathcal{D}$-règles

L'ensemble de $\mathcal{D}$-règles $R_{\mathcal{D}}(P)$ constitue un système inductif sur la $\mathcal{D}$-base. On note $T_{\mathcal{D}}^P$ l'opérateur de conséquence immédiate associé. Les arbres de preuve sont appelés *$\mathcal{D}$-arbres de preuve*, et $Rac_{\mathcal{D}}^P$ désigne l'ensemble des racines de $\mathcal{D}$-arbres de preuve.

La notion d'arbre de preuve, liée aux systèmes inductifs, est largement exploitée dans [8].

Lemme 2.3 *Une $\mathcal{D}$-interprétation I est modèle de P ssi I est pré-point fixe de $T_{\mathcal{D}}^P$.*

Preuve : Nous avons vu qu'une $\mathcal{D}$-interprétation I est modèle de la clause C : $A \leftarrow c \square B_1, \ldots, B_n$ ssi I est close pour les $\mathcal{D}$-règles de $R_{\mathcal{D}}(C)$. On en déduit que I est modèle de P ssi I est close pour toutes les $\mathcal{D}$-règles de $R_{\mathcal{D}}(P)$, c'est-à-dire ssi $T_{\mathcal{D}}^P(I)$ est inclus dans I.

Proposition 2.1 *Le programme P a un plus petit $\mathcal{D}$-modèle $M_{\mathcal{D}}^P$ tel que*

$$M_{\mathcal{D}}^P = Rac_{\mathcal{D}}^P = pppf(T_{\mathcal{D}}^P) = T_{\mathcal{D}}^P \uparrow w$$

Preuve : $Rac_{\mathcal{D}}^P = pppf(T_{\mathcal{D}}^P) = T_{\mathcal{D}}^P \uparrow \omega$ résulte immédiatement du lemme sur les points fixes dans les systèmes inductifs. $pppf(T_{\mathcal{D}}^P)$ est en même temps le plus petit pré-point fixe de $T_{\mathcal{D}}^P$; c'est donc le plus petit des $\mathcal{D}$-modèles de P.

Exemple 2.5 *Le programme Fib admet pour plus petit $\mathcal{N}$-modèle l'ensemble des $\mathcal{N}$-atomes représentant la fonction de Fibonacci :*
$M_1 = \{fib(0,1), fib(1,1), fib(2,2), fib(3,3), fib(4,5), fib(5,8), fib(6,13), fib(7,21)...\}$

Exemple 2.6 *Pour le programme Fib pré-interprété par $\mathcal{N}$, le $\mathcal{N}$-atome $fib(3,3)$ est racine du $\mathcal{N}$-arbre de preuve A de la figure 1.*

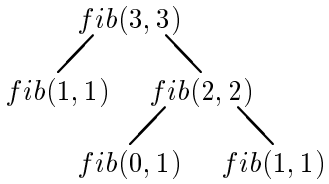

Figure 1 : Arbre de preuve A

2.5 Complété du programme

Le complété du programme P noté P^* est un ensemble de formules qui ne sont plus des clauses. Ce n'est donc pas un programme contraint tel que nous l'avons défini. Il présente un intérêt certain pour l'étude des réponses négatives du programme au sens de la négation par l'échec. Dans un premier temps nous définissons P^* et étudions ses $\mathcal{D}$-modèles.

La notion de complété est classique aussi bien en programmation logique qu'en programmation logique avec contraintes. Nous nous inspirons particulièrement de la présentation faite par [2].

Pour tout $p \in \Pi_P$, on appelle *définition* du prédicat p et on note $def(p, P)$ (en abrégé $def(p)$ si P est sous-entendu) l'ensemble des clauses de P dont le symbole de prédicat de tête est p. $Def(p)$ est de la forme :

$$\begin{array}{ll} \{ & p(\tilde{t_1}) \leftarrow c_1 \ \Box \ B_1^1, \ldots, B_{n_1}^1, \\ & p(\tilde{t_2}) \leftarrow c_2 \ \Box \ B_1^2, \ldots, B_{n_2}^2, \\ & \ldots \\ & p(\tilde{t_k}) \leftarrow c_k \ \Box \ B_1^k, \ldots, B_{n_k}^k \quad \} \end{array}$$

$Def(p)$ est vide lorsque P ne comprend aucune clause ayant p comme prédicat de tête.

On note $SI(p, P)$, ($SI(p)$ lorsque P est sous-entendu) la formule :

$$\begin{array}{lll} p(\tilde{x}) \leftarrow & (& \exists \tilde{y_1}(\tilde{x} = \tilde{t_1} \wedge c_1 \wedge B_1^1 \wedge \ldots \wedge B_{n_1}^1) \\ & \vee & \exists \tilde{y_2}(\tilde{x} = \tilde{t_2} \wedge c_2 \wedge B_1^2 \wedge \ldots \wedge B_{n_2}^2) \\ & & \ldots \\ & \vee & \exists \tilde{y_k}(\tilde{x} = \tilde{t_k} \wedge c_k \wedge B_1^k \wedge \ldots \wedge B_{n_k}^k) \end{array}$$

où $\tilde{y_i}$ est l'ensemble des variables libres de la $i^{\text{ème}}$ clause et $\tilde{x}$ est un ensemble de n (n = arité de p) variables distinctes entre elles et distinctes de toutes les $\tilde{y_i}$. Lorsque $def(p)$ est vide, $SI(p)$ comprend une disjonction indexée par $\emptyset$ qui se résume donc à $faux$. Alors $SI(p)$ est : $p(\tilde{x}) \leftarrow faux$.

On note $SSI(p, P)$ (en abrégé $SSI(p)$) la formule obtenue en remplaçant dans $SI(p)$ l'implication $\leftarrow$ par une équivalence $\leftrightarrow$.

Le complété P^* est l'ensemble $\{SSI(p, P) \mid p \in \Pi_P\}$.

Exemple 2.7 *Le programme Fib-Fact :*

$fib(0, 1) \leftarrow vrai$

$fib(1, 1) \leftarrow vrai$

$fib(N, A+B) \leftarrow N > 1 \ \square \ fib(N-2, A), fib(N-1, B)$
$fact(0, 1) \leftarrow vrai$
$fact(N, N*F) \leftarrow N > 0 \ \square \ fact(N-1, F)$

admet pour complété, si Π_P *est réduit à* $\{fib, fact\}$*, l'ensemble de formules Fib-Fact* :*

$$\begin{array}{rl} \{fib(X,Y) \leftrightarrow & ((X=0 \wedge Y=1) \\ & \vee (X=1 \wedge Y=1) \\ & \vee \exists N \exists A \exists B (X = N \wedge Y = A+B \wedge N > 1 \wedge fib(N-2, A) \\ & \wedge fib(N-1, B))) \\ fact(X,Y) \leftrightarrow & ((X=0 \wedge Y=1) \\ & \vee \exists N \exists F (X = N \wedge Y = N*F \wedge N > 0 \wedge fact(N-1, F)))\} \end{array}$$

Lemme 2.4 *Pour un programme P et une pré-interprétation $\mathcal{D}$ fixés, une $\mathcal{D}$-interprétation I est modèle de $SI(p)$ ssi elle est modèle de $def(p)$.*

Preuve :

1. Si $def(p)$ est vide, toute I est modèle de $def(p)$. $SI(p)$ se résume à $p(\tilde{x}) \leftarrow faux$ qui est vraie dans toute interprétation et toute valuation.

2. Si $def(p)$ est non vide, $def(p) =$

$$\begin{array}{ll} \{ & p(\tilde{t_1}) \leftarrow c_1 \ \square \ B_1^1, \ldots, B_{n_1}^1, \\ & p(\tilde{t_2}) \leftarrow c_2 \ \square \ B_1^2, \ldots, B_{n_2}^2, \\ & \ldots \\ & p(\tilde{t_k}) \leftarrow c_k \ \square \ B_1^k, \ldots, B_{n_k}^k \quad \} \end{array}$$

   et $SI(p) =$

$$\begin{array}{rll} p(\tilde{x}) \leftarrow & (& \exists \tilde{y_1} (\tilde{x} = \tilde{t_1} \wedge c_1 \wedge B_1^1 \wedge \ldots \wedge B_{n_1}^1) \\ & \vee & \exists \tilde{y_2} (\tilde{x} = \tilde{t_2} \wedge c_2 \wedge B_1^2 \wedge \ldots \wedge B_{n_2}^2) \\ & & \ldots \\ & \vee & \exists \tilde{y_k} (\tilde{x} = \tilde{t_k} \wedge c_k \wedge B_1^k \wedge \ldots \wedge B_{n_k}^k) \end{array}$$

   - Les formules (i) à (iv) ci-dessous sont sémantiquement équivalentes (elles admettent les mêmes $\mathcal{D}$-modèles) :
     (i) $p(\tilde{t_i}) \leftarrow c_i \ \square \ B_1^i, \ldots, B_{n_i}^i$,
     (ii) $p(\tilde{x}) \leftarrow \tilde{x} = \tilde{t_i} \wedge c_i \ \square \ B_1^i, \ldots, B_{n_i}^i$,
     (iii) $p(\tilde{x}) \leftarrow \tilde{x} = \tilde{t_i} \wedge c_i \wedge B_1^i \wedge \ldots \wedge B_{n_i}^i$,
     (iv) $p(\tilde{x}) \leftarrow \exists \tilde{y_i} (\tilde{x} = \tilde{t_i} \wedge c_i \wedge B_1^i \wedge \ldots \wedge B_{n_i}^i)$,

     (i) $\Leftrightarrow$ (ii) : (i) vraie dans I signifie : pour toute valuation v, $v_{\mathcal{D}}(c_i) = vrai$ et $v_{\mathcal{D}}(B_j^i) \in I \ \forall j \Rightarrow p(v_{\mathcal{D}}(\tilde{t_i})) \in I$. Alors $\forall v \ v_{\mathcal{D}}(c_i) = vrai$ et $v(\tilde{x}) = v_{\mathcal{D}}(\tilde{t_i})$ et $v_{\mathcal{D}}(B_j^i) \in I \ \forall j \Rightarrow p(v(\tilde{x})) \in I$ parce que $v(\tilde{x}) = v_{\mathcal{D}}(\tilde{t_i})$ (au sens de l'identité sur $\mathcal{D}$). Donc (ii) est vraie dans I.
     Inversement, si (ii) est vraie dans I, considérons une valuation v telle que $v_{\mathcal{D}}(c_i) = vrai$ et $v_{\mathcal{D}}(B_j^i) \in I \forall j$. Cette valuation porte sur les variables $\tilde{y_i}$ de (i). Comme $\tilde{x}$ est un ensemble de variables disjoint de $\tilde{y_i}$, on peut étendre v à v' portant sur $\tilde{y_i} \bigcup \tilde{x}$ en donnant à $\tilde{x}$ les valeurs $v_{\mathcal{D}}(\tilde{t_i})$. Alors la partie droite de (ii) est vraie dans v' ; donc $p(v(\tilde{x})) \in I$. Sachant que $v(\tilde{x}) = v_{\mathcal{D}}(\tilde{t_i})$,

$p(v_{\mathcal{D}}(\tilde{t_i})) \in I$.

(ii) $\Leftrightarrow$ (iii) est évident car le corps de (ii) est vrai dans une valuation v ssi les contraintes $\tilde{x} = \tilde{t_i}$ et c_i sont vraies et si les B^i_j sont vrais.

(iii) $\Leftrightarrow$ (iv) peut s'exprimer sous une forme plus générale : soit $\tilde{x}$ et $\tilde{y}$ deux ensembles de variables disjoints, A une formule ayant $\tilde{x}$ pour variables libres, B une formule ayant $\tilde{x} \bigcup \tilde{y}$ pour variables libres. Alors les formules (a) et (b) suivantes sont sémantiquement équivalentes :

- (a) $A \leftarrow B$
- (b) $A \leftarrow \exists \tilde{y}\ B$

(a) $\Leftrightarrow \forall$ valuation v (sur $\tilde{x} \bigcup \tilde{y}$) $v_I(B) = vrai \Rightarrow v_I(A) = vrai$
(b) $\Leftrightarrow \forall$ valuation v (sur $\tilde{x}$), s'il existe un prolongement de v en v' (sur $\tilde{x} \bigcup \tilde{y}$) tel que $v'_I(B) = vrai$, alors $v_I(A) = vrai$.
(a) $\Rightarrow$ (b) : soit v sur $\tilde{x}$, et un prolongement v' sur $\tilde{x} \bigcup \tilde{y}$ tel que $v'_I(B) = vrai$; alors (a) $\Rightarrow v'_I(A) = vrai$; et $v_I(A) = v'_I(A)$; donc $v_I(A) = vrai$.
(b) $\Rightarrow$ (a) : soit v sur $\tilde{x} \bigcup \tilde{y}$ telle que $v_I(B) = vrai$; alors la restriction v_0 de v à $\tilde{x}$ admet un prolongement v à $\tilde{x} \bigcup \tilde{y}$ tel que $v_I(B) = vrai$; (b) $\Rightarrow v_{0\,I}(A) = vrai$; et $v_I(A) = v_{0\,I}(A)$; donc $v_I(A) = vrai$.

- $def(p)$ est donc sémantiquement équivalent à l'ensemble d'implications :

$$\begin{array}{lll} \{ & p(\tilde{x}) \leftarrow & \exists \tilde{y_1}(\tilde{x} = \tilde{t_1} \wedge c_1 \wedge B^1_1 \wedge \ldots \wedge B^1_{n_1}) \\ & p(\tilde{x}) \leftarrow & \exists \tilde{y_2}(\tilde{x} = \tilde{t_2} \wedge c_2 \wedge B^2_1 \wedge \ldots \wedge B^2_{n_2}) \\ & & \ldots \\ & p(\tilde{x}) \leftarrow & \exists \tilde{y_k}(\tilde{x} = \tilde{t_k} \wedge c_k \wedge B^k_1 \wedge \ldots \wedge B^k_{n_k})\} \end{array}$$

- l'ensemble de formules ci-dessus de la forme $E = \{A \leftarrow B_1, \ldots, A \leftarrow B_k\}$ est sémantiquement équivalent à l'unique formule $A \leftarrow B_1 \vee \ldots \vee B_k$ car selon E, dès que l'une au moins des B_i est vraie, alors A doit aussi être vraie ; cela équivaut à: si $B_1 \vee \ldots \vee B_k$ est vraie alors A est vraie, donc à : $A \leftarrow B_1, \ldots, B_k$.

Corollaire 2.1 *(évident) Pour toute $\mathcal{D}$-interprétation I, I est modèle de P ssi I est modèle de $SI(P) = \{SI(p, P) \mid p \in \Pi_P\}$. Donc I est modèle de $SI(P)$ ssi I est pré-point fixe de $T^P_{\mathcal{D}}$.*

Lemme 2.5 *Soit $IS(P)$ l'ensemble de formules obtenu en retournant les implications de $SI(P)$. Alors I est un modèle de $IS(P)$ ssi $T^P_{\mathcal{D}}(I)$ contient I.*

Preuve : $IS(P)$ est un ensemble de formules du type :
(1) $\quad p(\tilde{x}) \rightarrow \bigvee_{i=1}^{k} \exists \tilde{y_i}(\tilde{x} = \tilde{t_i} \wedge c_i \wedge B^i_1 \wedge \ldots B^i_{n_i})$

1. Soit I un modèle de $IS(P)$. Montrons qu'alors $p(\tilde{d}) \in I \Rightarrow p(\tilde{d}) \in T^P_{\mathcal{D}}(I)$. I est modèle de (1), donc dans la valuation v: $\tilde{x} = \tilde{d}$, $p(\tilde{d})$ étant dans I, il existe un i $1 \leq i \leq k$ et une valuation v' prolongeant v à $\tilde{x} \bigcup \tilde{y}$ tels que $v_I(\tilde{x} = \tilde{t_i} \wedge c_i \wedge B^i_1 \wedge \ldots B^i_{n_i}) = vrai$. Donc

   - $v_{\mathcal{D}}(\tilde{t_i}) = v_{\mathcal{D}}(\tilde{x}) = \tilde{d}$,

- $v'_{\mathcal{D}}(c_i) = vrai$,
- $v'_{\mathcal{D}}(B^i_1) \in I$,
- ... ...
- $v'_{\mathcal{D}}(B^i_{n_i}) \in I$.

Alors $p(v_{\mathcal{D}}(\tilde{t_i})) \leftarrow v_{\mathcal{D}}(B^i_1), \ldots, v_{\mathcal{D}}(B^i_{n_i})$ fait partie des règles qui définissent l'opérateur $T^P_{\mathcal{D}}$, et chaque $v_{\mathcal{D}}(B^i_j)$ est dans I. Donc $p(v_{\mathcal{D}}(\tilde{t_i})) = p(\tilde{d})$ est dans $T^P_{\mathcal{D}}(I)$.

2. Supposons que I est inclus dans $T^P_{\mathcal{D}}(I)$. Il nous faut montrer que I est modèle de (1). Pour toute valuation $v : \tilde{x} = \tilde{d}$

   (i) soit $p(\tilde{d})$ n'est pas dans I, alors (1) est trivialement vraie dans I et v,

   (ii) soit $p(\tilde{d}) \in I$. Alors $p(\tilde{d}) \in T^P_{\mathcal{D}}(I)$. Donc il existe une $\mathcal{D}$-règle $p(\tilde{d}) \leftarrow a_1, \ldots, a_m$ telle que $a_1, \ldots, a_m \in I$. Cette règle est issue d'une clause de $def(p, P) : p(\tilde{t_i}) \leftarrow c_i \square B^i_1, \ldots, B^i_m$

   et il existe une valuation v" sur les variables libres $\tilde{y_i}$ telle que

   $v''_{\mathcal{D}}(\tilde{t_i}) = \tilde{d}$

   $v''_{\mathcal{D}}(c_i) = vrai$

   $v''_{\mathcal{D}}(B^i_1) = a_1$

   ...

   $v''_{\mathcal{D}}(B^i_m) = a_m$

   La forme équivalente de cette clause intégrée dans $SI(p)$ est $p(\tilde{x}) \leftarrow \exists \tilde{y_i}(\tilde{x} = \tilde{t_i} \wedge c_i \wedge B^i_1 \wedge \ldots \wedge B^i_m)$.

   La partie gauche $p(\tilde{x})$ est vraie dans I et v car $v_{\mathcal{D}}(p(\tilde{x})) = p(\tilde{d}) \in I$. La partie droite est vraie dans I et v grâce à l'existence de la valuation v''. (1) est donc vraie dans I et v car l'une des formules de la disjonction est vraie.

Proposition 2.2 *Les $\mathcal{D}$-modèles de P^* sont les points fixes de $T^P_{\mathcal{D}}$.*

Preuve : $SSI(P)$ est équivalent à la réunion de $SI(P)$ et de $IS(P)$. Or I est modèle de $SI(P)$ ssi $T^P_{\mathcal{D}}(I)$ est inclus dans I. I est modèle de $IS(P)$ ssi $T^P_{\mathcal{D}}(I)$ contient I. Donc I est modèle de $SSI(P)$ ssi $T^P_{\mathcal{D}}(I) = I$.

Corollaire 2.2 *Pour toute pré-interprétation $\mathcal{D}$,*

- *tout $\mathcal{D}$-modèle de P^* est un $\mathcal{D}$-modèle de P,*
- *les plus petits $\mathcal{D}$-modèles de P et de P^* sont identiques,*
- $P^* \models P$.

Preuve :

- évident car les $\mathcal{D}$-modèles de P sont les pré-points fixes de $T^P_{\mathcal{D}}$ (les I tels que $T^P_{\mathcal{D}}(I) \subseteq I$), alors que ceux de P^* sont les points fixes de $T^P_{\mathcal{D}}$; tout point fixe est a fortiori un pré-point fixe.

- le plus petit $\mathcal{D}$-modèle de P^* est le plus petit point fixe de $T_{\mathcal{D}}^P$; c'est en même temps le plus petit pré-point fixe de $T_{\mathcal{D}}^P$.
- tout $\mathcal{D}$-modèle de P^* est un modèle de P.

Nous connaissons donc les $\mathcal{D}$-modèles de P^* : ce sont les points fixes de $T_{\mathcal{D}}^P$. Les plus petits modèles de P et de P^* sont identiques, mais pas les plus grands modèles. En effet, le plus grand modèle de P est la $\mathcal{D}$-base tout entière. Il n'y a aucune raison que ce soit un point fixe de $T_{\mathcal{D}}^P$, donc un modèle de P^*. Afin de décrire plus précisément le plus grand modèle de P^*, nous introduisons une nouvelle notion: celle de $\mathcal{D}\infty$-arbre de preuve qui admet la même définition que le $\mathcal{D}$-arbre de preuve, la finitude en moins.

Définition 2.3 *Un $\mathcal{D}\infty$-arbre de preuve est un arbre éventuellement infini dont les nœuds sont étiquetés par des $\mathcal{D}$-atomes ayant la propriété : pour tout nœud N, soit a son étiquette,*

- si N est une feuille alors $a \leftarrow$ est un $\mathcal{D}$-fait,

- si N n'est pas une feuille, il a n fils étiquetés $a_1, \ldots, a_n$ et $a \leftarrow a_1, \ldots, a_n$ est une $\mathcal{D}$-règle.

On remarquera que les $\mathcal{D}$-arbres de preuves (finis) en font partie.

Proposition 2.3 *Le plus grand $\mathcal{D}$-modèle de P^* est l'ensemble des racines de $\mathcal{D}\infty$-arbres de preuve.*

Preuve : Notons R l'ensemble des racines des $\mathcal{D}\infty$-arbres de preuve.

1. $T_{\mathcal{D}}^P(R)$ contient R.

   Soit $a \in R$. Alors a est racine d'un $\mathcal{D}\infty$-arbre de preuve A qui a n sous-arbres, eux-mêmes $\mathcal{D}\infty$-arbres de preuve A_1, ... , A_n.

   - Si $n = 0$, $a \leftarrow$ est un $\mathcal{D}$-fait, et alors $a \in T_{\mathcal{D}}^P(\emptyset)$, donc $a \in T_{\mathcal{D}}^P(R)$.

   - Si $n > 0$, chaque A_i a pour racine un $\mathcal{D}$-atome a_i qui est dans R et $a \leftarrow a_1, \ldots, a_n$ est une $\mathcal{D}$-règle. Donc $a \in T_{\mathcal{D}}^P(R)$.

2. Toute $\mathcal{D}$-interprétation I telle que $T_{\mathcal{D}}^P(I)$ contient I est incluse dans R.

   Soit $a \in I$ telle que $T_{\mathcal{D}}^P(I)$ contient I. Considérons alors la suite (éventuellement infinie) d'arbres étiquetés par des $\mathcal{D}$-atomes éléments de I : A_1, A_2, ... , A_i, A_{i+1}, ...

   - A_1 est l'arbre réduit à sa racine étiquetée par a,

   - les arbres A_1, ..., A_i étant construits, A_i a éventuellement un successeur A_{i+1} défini de la manière suivante : si toutes les feuilles de A_i correspondent à des $\mathcal{D}$-faits, alors A_i n'a pas de successeur (cas particulier : si a correspond au $\mathcal{D}$-fait $a \leftarrow$ alors la suite s'arrête à A_1) ; sinon, soit F la première feuille de A_i ne correspondant pas à un $\mathcal{D}$-fait dans le parcours en largeur de A_i ; F est étiquetée par $x \in I$, donc $x \in T_{\mathcal{D}}^P(I)$, donc il existe une $\mathcal{D}$-règle $x \leftarrow x_1, \ldots, x_k$ ($k > 0$ sinon F correspondrait à un $\mathcal{D}$-fait) et $x_1, \ldots x_k \in I$. Alors A_{i+1} est obtenu en greffant sur F (qui devient donc un nœud interne) k feuilles étiquetées par x_1, ... , x_k.

Si la suite d'arbres est finie, le dernier arbre A_i est un $\mathcal{D}$-arbre de preuve (fini) de racine a ; donc $a \in Rac^P_{\mathcal{D}}$ qui est inclus dans R.

Si la suite A_i est infinie, elle définit un $\mathcal{D}\infty$-arbre de preuve A de racine a : ses nœuds sont l'union (croissante) des ensembles de nœuds des A_i. Donc a est dans R.

3. Donc R est le plus grand I tel que $T^P_{\mathcal{D}}(I)$ contient I. D'après le théorème de Knaster-Tarski, c'est le plus grand point fixe de $T^P_{\mathcal{D}}$.

2.6 Squelettes

Un $\mathcal{D}$-arbre de preuve utilise, pour "prouver" sa racine, des $\mathcal{D}$-règles qui sont issues de clauses du programme. L'ensemble des clauses utilisées peut être représenté par un arbre, étiqueté par des clauses, qui a même forme que le $\mathcal{D}$-arbre de preuve. C'est ce que nous appellerons un squelette.

Le squelette est la notion primordiale de [8] que nous reprenons ici dans le contexte de la programmation logique avec contraintes en assouplissant légèrement la définition.

Définition 2.4 *(Squelette)*

Un squelette *pour le programme P est un arbre S étiqueté par des clauses de P tel que, pour tout nœud N, soit E son étiquette,*

- si N est une feuille, alors E est d'arité 0 : c'est un fait,

- si N n'est pas une feuille, alors il a n fils ($n > 0$) et E est d'arité n.

Comme pour les arbres de preuve, il n'est pas nécessaire de distinguer les feuilles : ce sont les nœuds ayant 0 fils, les clauses qui les étiquettent sont d'arité 0.

Le même abus de langage que pour les arbres de preuve conduit à nommer *racine* d'un squelette aussi bien l'étiquette du nœud racine (c'est une clause) que le nœud racine proprement dit. Remarquons qu'un squelette n'est pas nécessairement un arbre fini.

Remarque 2.3 *(importante)*

Les squelettes ne font intervenir que les clauses du programme P ; ils ne dépendent pas de la pré-interprétation $\mathcal{D}$.

On dit que S est un *squelette pour le $\mathcal{D}$-arbre de preuve A* si A résulte de S par valuations. Plus précisément,

1. les arbres A et S sont identiques si l'on fait abstraction des étiquettes (il en résulte que S est fini),

2. soit N_A un nœud de A étiqueté par a ; il a n fils ($0 \leq n$) étiquetés par a_1, ..., a_n ; soit N_S son correspondant dans S. Alors l'étiquette de N_S est une clause d'arité n : $H \leftarrow c \,\square\, B_1, \ldots, B_n$ et il existe une valuation v telle que $v_{\mathcal{D}}(c) = vrai$, $v_{\mathcal{D}}(H) = a$, $v_{\mathcal{D}}(B_1) = a_1$, ..., $v_{\mathcal{D}}(B_n) = a_n$.

Exemple 2.8 *Pour le programme Fib pré-interprété par $\mathcal{N}$, le squelette S_1 de la figure 2 est un squelette pour le $\mathcal{N}$-arbre de preuve A_1 de la figure 3 grâce aux valuations :*

$\{N = 3, A = 1, B = 2\}$ pour le nœud racine,

$\{N = 2, A = 1, B = 1\}$ pour son fils droit.

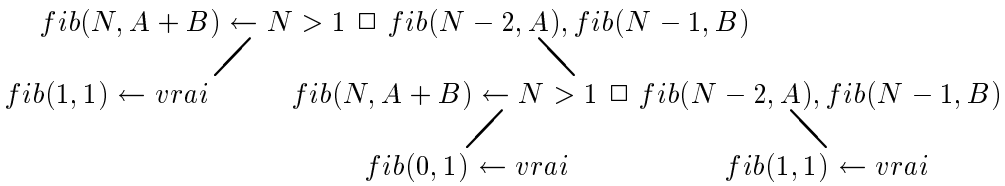

Figure 2 : Squelette S_1

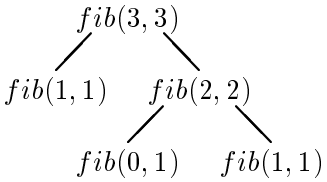

Figure 3 : Arbre de preuve A_1

Pour tout $\mathcal{D}$-arbre de preuve A il existe au moins un squelette S tel que S soit une squelette pour A. En effet, pour tout nœud N_A de A, si a est son étiquette et si a_1, ..., a_n sont les étiquettes de ses fils, alors $a \leftarrow a_1, \ldots, a_n$ est une $\mathcal{D}$-règle de $R_{\mathcal{D}}(P)$. Cette $\mathcal{D}$-règle résulte d'une clause $H \leftarrow c \ \square \ B_1, \ldots, B_n$ de P et d'une valuation v telles que $v_{\mathcal{D}}(c) = vrai$, $v_{\mathcal{D}}(H) = a$, $v_{\mathcal{D}}(B_1) = a1$, ... , $v_{\mathcal{D}}(B_n) = a_n$. Donc pour tout nœud de A il existe au moins une clause de P qui convient. On obtient le squelette (fini) S en choisissant, pour chaque nœud de A, une clause parmi les clauses possibles.

On remarquera que le choix des clauses n'est pas unique, donc que le squelette pour A n'est pas unique, lorsqu'une règle utilisée par A peut être obtenue à partir de plusieurs clauses différentes.

Exemple 2.9 *Pour le programme*
$p(a) \leftarrow vrai$
$p(X) \leftarrow vrai$
où a est un constante dont le correspondant dans D est d, et X est une variable, la $\mathcal{D}$-règle $p(d) \leftarrow$ est obtenue de deux manières différentes : à partir de la clause $p(a) \leftarrow vrai$ avec une valuation quelconque, à partir de la clause $p(X) \leftarrow vrai$ avec une valuation v telle que $v(X) = d$.

En sens inverse un squelette S, même fini, peut n'être squelette pour aucun arbre de preuve.

Exemple 2.10 *Pour le programme Fib, le squelette S_2 de la figure 4 ne correspond à aucun $\mathcal{N}$-arbre de preuve parce qu'il n'existe pas de valuation dans laquelle $N - 2 = 1$ et $N - 1 = 0$.*

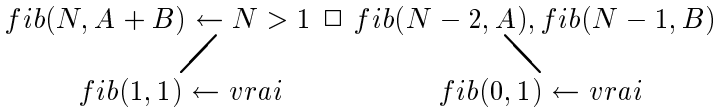

Figure 4 : Squelette S_2

2.7 Contraintes associées à un squelette

Nous définissons ici le système de contraintes associé à un squelette en généralisant le système d'équations associé à un squelette selon [8].

Etant donné un squelette S, pour savoir s'il existe un $\mathcal{D}$-arbre de preuve dont il soit squelette, il faut chercher pour chaque nœud une valuation, de telle sorte qu'en appliquant toutes les valuations en tous les nœuds on obtienne les $\mathcal{D}$-règles d'un $\mathcal{D}$-arbre de preuve.

Nous avons vu que si S est un squelette pour le $\mathcal{D}$-arbre de preuve A, à chaque nœud N de S sont associées une clause C et une valuation v telles que $v_{\mathcal{D}}(C)$ privée de sa contrainte vraie soit la $\mathcal{D}$-règle utilisée par le nœud correspondant de A. Cette valuation est locale au nœud : elle ne sert qu'à assurer la correspondance entre la clause et la $\mathcal{D}$-règle. La partie utile de v se résume à sa restriction aux variables de C, en ce sens que toute autre valuation identique à v sur les variables de C pourrait être mise en place de v. Les clauses intervenant dans S peuvent avoir des variables communes; une même clause peut d'ailleurs intervenir plusieurs fois (voir notamment l'exemple 2.8).

Si les clauses de S n'avaient pas de variables communes, trouver les valuations à appliquer localement aux différents nœuds de S reviendrait à trouver une *valuation globale* applicable à l'ensemble des clauses de S. Nous allons nous ramener à ce cas.

Une clause C' est une *variante* de la clause C si C' est obtenue à partir de C par un *renommage*, c'est-à-dire l'application d'une substitution dans laquelle les variables de C sont remplacées par des variables toutes distinctes.

La relation "être une variante" est une équivalence sur l'ensemble des clauses. Chaque classe d'équivalence peut être représentée par une clause C, et l'on nommera les *variantes de* C les clauses de cette classe.

On considère désormais que l'étiquette $E(N)$ de chaque nœud N du squelette S est, non une clause, mais une classe d'équivalence de clauses; cette classe est représentée par une clause de la classe : une clause de P.

Avant de chercher s'il existe une valuation globale regroupant toutes les valuations locales, on commence par choisir, pour tout nœud N, une clause $C(N)$ dans la classe d'équivalence $E(N)$, de telle sorte que pour tout couple de nœuds distincts les clauses choisies n'aient pas de variables communes. Cette fonction C, qui a de multiples valeurs possibles ne différant que par renommage des variables, sera appelée fonction de choix de variantes de clauses, ou en abrégé *fonction de choix*.

La fonction de choix C étant fixée, on peut associer à tout nœud N autre que la racine un atome noté $atome(N)$ défini par : si N est le $i^{ème}$ fils de N', si $C(N')$ est $A' \leftarrow c' \ \square \ B'_1, \ldots, B'_p$, alors $atome(N) = B'_i$. $Atome(N)$ dépend de la fonction C qui choisit pour chaque nœud une variante de clause; mais le remplacement de C par une autre fonction C' ne fait que renommer les variables de $atome(N)$.

Chercher un $\mathcal{D}$-arbre de preuve se résume à chercher une valuation v telle que pour tout nœud N, si $C(N)$ est $A \leftarrow c \square B_1, \ldots, B_n$

1. la contrainte c est vraie dans v : $v_{\mathcal{D}}(c) = vrai$
2. si N_i est le $i^{ème}$ fils de N et est étiqueté par la clause $A' \leftarrow c' \square B'_1, \ldots, B'_p$ alors $v_{\mathcal{D}}(B_i) = v_{\mathcal{D}}(A')$ (au sens de l'identité sur la $\mathcal{D}$-base), autrement dit $v_{\mathcal{D}}(atome(N_i)) = v_{\mathcal{D}}(tête(C(N_i)))$.

Etant donné deux atomes $A : p(t_1, \ldots, t_k)$ et $B : q(u_1, \ldots, u_l)$, $v_{\mathcal{D}}(A) = v_{\mathcal{D}}(B)$ (au sens de l'identité sur la $\mathcal{D}$-base) si et seulement si

- p et q sont le même symbole de prédicat (élément de Π_P), auquel cas $k = l$,
- et de plus $v_{\mathcal{D}}(t_1) = v_{\mathcal{D}}(u_1)$, ... , $v_{\mathcal{D}}(t_k) = v_{\mathcal{D}}(u_k)$ (au sens de l'identité sur D).

Nous avons supposé que $\mathcal{L}(V, \Sigma, \Pi_C)$ comprend le symbole d'égalité (toujours interprété comme l'égalité sur D), et les constantes booléennes $vrai$ et $faux$ (toujours interprétées comme $vrai$ et $faux$). Alors $v_{\mathcal{D}}(A) = v_{\mathcal{D}}(B)$ si et seulement si la contrainte c_{AB} est vraie dans $\mathcal{D}$ et v, où c_{AB} est construite de la manière suivante :

- si A et B n'ont pas le même symbole de prédicat, alors c_{AB} est $faux$,
- sinon, c_{AB} est $t_1 = u_1 \wedge \ldots \wedge t_k = u_k$ (ici $=$ est le symbole du langage des contraintes).

Cette contrainte c_{AB} sera appelée contrainte d'égalité entre A et B et notée $A = B$.

Définition 2.5 *(Collection de contraintes)*
La collection de contraintes $cont(S)$ associée au squelette S *est l'union des collections de contraintes associées aux nœuds de* S *pour la fonction de choix des variantes de clauses* C *: à tout nœud* N *on associe*
1- la contrainte de la variante de clause associée à N *:* $contrainte(C(N))$
2- et si N *n'est pas la racine, la contrainte d'égalité* $atome(N) = tête(C(N))$.

Dans les squelettes selon [8], $atome(N)$ et $tête(C(N))$ ont nécessairement le même symbole de prédicat. Le passage aux contraintes et l'utilisation de la contrainte $faux$ nous permettent de supprimer cette restriction.

$Cont(S)$ dépend de la fonction C, mais le remplacement d'une fonction C par une autre ne fait que renommer les variables apparaissant dans la collection de contraintes.

Remarque 2.4 *Squelettes et collections de contraintes associées sont des notions indépendantes de la pré-interprétation* $\mathcal{D}$.

Définition 2.6 *(Squelette $\mathcal{D}$-soluble)*
Un squelette S *est dit* $\mathcal{D}$-soluble *si la collection de contraintes qui lui est associée (définie à un renommage près) est satisfiable dans* $\mathcal{D}$.

Remarquons qu'un squelette comportant un nœud N autre que la racine tel que $atome(N)$ et $tête(C(N))$ n'ont pas le même symbole de prédicat n'est certainement pas $\mathcal{D}$-soluble car sa collection de contraintes contient $faux$.

Si $cont(S)$ est satisfiable, c'est qu'il existe une valuation v qui soit solution de $cont(S)$. On dira par extension que v *est une solution* de S dans $\mathcal{D}$. Par analogie avec le prolongement $v_{\mathcal{D}}$ de v aux contraintes, atomes et clauses, v se prolonge à tout squelette S dont elle est $\mathcal{D}$-solution, et $v_{\mathcal{D}}(S)$ est le $\mathcal{D}$-arbre de preuve A dont chaque

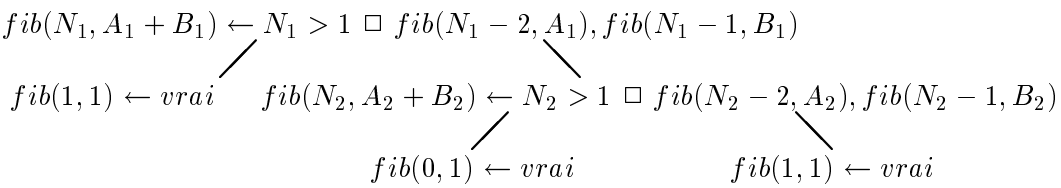

Figure 5 : Squelette S_3

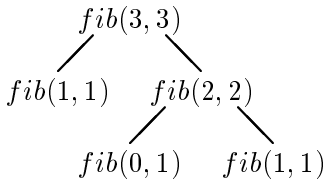

Figure 6 : Arbre de preuve A_3

nœud N_A est étiqueté par $v_{\mathcal{D}}(tête(C(N_S)))$ où N_S est le nœud correspondant à N_A dans S.

Nous avons dit plus haut que pour tout $\mathcal{D}$-arbre de preuve A il existe un squelette S qui soit un squelette pour A. Cela veut dire que S est fini et $\mathcal{D}$-soluble, et qu'il existe une valuation v telle que $v_{\mathcal{D}}(S) = A$. La correspondance entre $\mathcal{D}$-arbres de preuves et squelettes $\mathcal{D}$-solubles s'établit donc de la manière suivante :

- pour tout $\mathcal{D}$-arbre de preuve A il existe un squelette $\mathcal{D}$-soluble qui soit un squelette pour A,

- pour tout squelette fini $\mathcal{D}$-soluble S il existe un $\mathcal{D}$-arbre de preuve A tel que S soit un squelette pour A.

La correspondance entre $\mathcal{D}$-arbres de preuve et squelettes finis $\mathcal{D}$-solubles s'étend aux arbres infinis de la manière suivante :

- pour tout $\mathcal{D}\infty$-arbre de preuve A il existe un squelette (éventuellement infini) $\mathcal{D}$-soluble qui soit un squelette pour A,

- pour tout squelette (éventuellement infini) $\mathcal{D}$-soluble S il existe un $\mathcal{D}\infty$-arbre de preuve A tel que S soit un squelette pour A.

Exemple 2.11 *Le squelette S_3 de la figure 5 où les clauses ont été renommées selon une certaine fonction de choix a pour collection de contraintes :*

$\{N_1 > 1, vrai, N_1 - 2 = 1, A_1 = 1, N_2 > 1, N_1 - 1 = N_2, B_1 = A_2 + B_2, vrai, N_2 - 2 = 0, A_2 = 1, vrai, N_2 - 1 = 1, B_2 = 1\}$

qui admet pour solution dans $\mathcal{N}$ la valuation :

$\{N_1 = 3, A_1 = 1, B_1 = 2, N_2 = 2, A_2 = 1, B_2 = 1\}$

Le $\mathcal{N}$-arbre de preuve A_3 correspondant est représenté figure 6.

Exemple 2.12 *Le squelette S_4 de la figure 7 a pour collection de contraintes :*

$\{N > 1, vrai, N - 2 = 1, A = 1, vrai, N - 1 = 0, B = 1\}$

qui est insatisfiable dans $\mathcal{N}$ car $N - 2 = 1$ et $N - 1 = 0$ sont contradictoires.

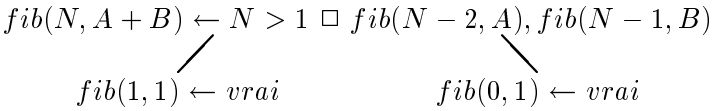

Figure 7 : Squelette S_4

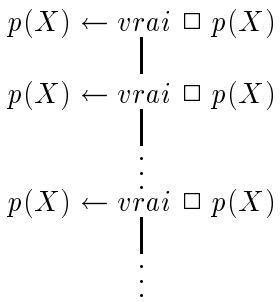

Figure 8 : Squelette infini S_5

Exemple 2.13 *Un programme qui contient la clause*
$p(X) \leftarrow vrai \ \square \ p(X)$
pré-interprété par un $\mathcal{D}$ quelconque admet le squelette infini S_5 de la figure 8. Celui-ci est $\mathcal{D}$-soluble car sa collection de contraintes : $\{vrai, vrai, X_1 = X_2, vrai, X_2 = X_3, \ldots, vrai, X_n = X_{n+1}, \ldots\}$ admet pour solution toute valuation qui attribue aux variables X_1, X_2, ... , X_n, ... la même valeur $d \in D$.

Exemple 2.14 *Le programme pré-interprété par $\mathcal{N}$:*
$p(0,0) \leftarrow vrai$
$p(X, R) \leftarrow X > 0 \ \square \ p(X+1, R+1)$
admet le squelette infini $\mathcal{N}$-soluble S_6 de la figure 9. Sa collection de contraintes
$\{X_1 > 0, X_2 > 0, X_1+1 = X_2, R_1+1 = R_2, \ldots, X_{n+1} > 0, X_n+1 = X_{n+1}, R_n+1 = R_{n+1}, \ldots\}$
admet en effet pour solution dans $\mathcal{N}$ la valuation
$\{X_1 = R_1 = 1, X_2 = R_2 = 2, \ldots, X_n = R_n = n, \ldots\}$.

2.8 Questions, réponses

La notion de réponse que nous définissons ici est largement inspirée de celle de [8] pour les programmes logiques.

L'utilité d'un programme réside en bonne partie dans les réponses qu'il peut donner aux questions qui lui sont posées.

Une *question* est un but (clause sans tête) : $\leftarrow c \ \square \ B_1, \ldots, B_n$.

Pour définir ce qu'est une réponse, nous avons besoin d'étendre légèrement la notion de squelette. Jusqu'à présent, les nœuds d'un squelette sont étiquetés par des clauses,

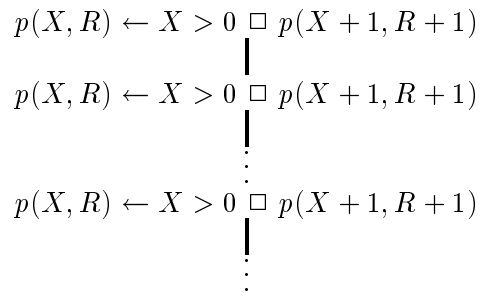

Figure 9 : Squelette infini S_6

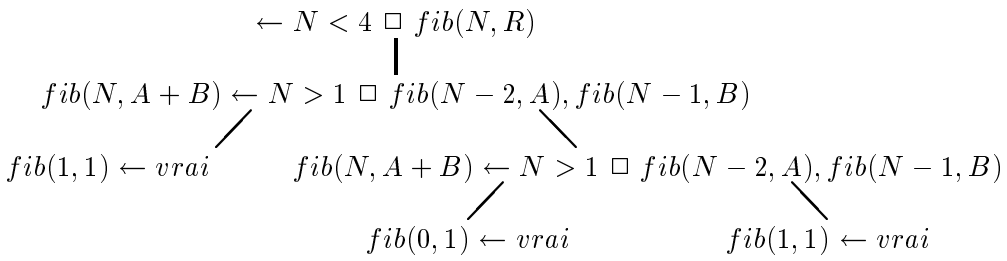

Figure 10 : Squelette S_7

qui représentent leurs classes d'équivalence par variante. Désormais, la racine d'un squelette peut être étiquetée par un but. La construction de la collection de contraintes $cont(S)$ associée à S n'a guère à être modifiée car la tête de la clause associée au nœud N n'intervient que si N n'est pas la racine. *Nous imposerons seulement que les variables du but question ne soient pas renommées.*

Définition 2.7 *($\mathcal{D}$-Réponse)*
On appelle $\mathcal{D}$-réponse *pour le but B tout squelette fini $\mathcal{D}$-soluble de racine B ; on dit également que c'est un $\mathcal{D}$-squelette réponse.*

Exemple 2.15 *Le squelette S_7 de la figure 10 pour le programme Fib est une $\mathcal{N}$-réponse pour le but $\leftarrow N < 4 \;\square\; fib(N,R)$ car sa collection de contraintes :* $\{N < 4, N_1 > 1, N = N_1, R = A_1 + B_1, vrai, N_1 - 2 = 1, A_1 = 1, N_2 > 1, N_1 - 1 = N_2, B_1 = A_2 + B_2, vrai, N_2 - 2 = 0, A_2 = 1, vrai, N_2 - 1 = 1, B_2 = 1\}$ *a pour solution la valuation* $\{N = 3, R = 3, N_1 = 3, A_1 = 1, B_1 = 2, N_2 = 2, A_2 = 1, B_2 = 1\}$.

2.9 Contraintes réponses, solutions

Une $\mathcal{D}$-réponse pour le but B est un squelette S fini $\mathcal{D}$-soluble de racine B. Sa racine est donc étiquetée par $B : \leftarrow c \;\square\; B_1, \ldots, B_n$. Si S est $\mathcal{D}$-soluble, c'est que la collection de contraintes $cont(S)$, définie à un renommage près des variables, est satisfiable dans $\mathcal{D}$. $Cont(S)$ étant finie, elle est assimilable à une unique contrainte : la conjonction

des contraintes de la collection que nous noterons $\mathcal{C}(S)$. Mais le résultat intéressant est l'incidence de cette contrainte sur les variables libres (non quantifiées) de la question B, les autres variables apparaissant dans $cont(S)$ n'étant que des auxiliaires.

Définition 2.8 *(Contrainte associée à un squelette fini, $\mathcal{D}$-contrainte réponse)*
La contrainte $ca(S)$ associée au squelette fini S est obtenue en :

- *construisant la collection de contraintes $cont(S)$ (à un renommage près des variables),*
- *rassemblant, par conjonction, cette collection en une unique contrainte $\mathcal{C}(S)$,*
- *prenant la fermeture existentielle de $\mathcal{C}(S)$ sur les variables auxiliaires : si $\tilde{x}$ désigne les variables libres de B et $\tilde{y}$ les variables auxiliaires, la contrainte associée est : $ca(S) = \exists_{-\tilde{x}}\mathcal{C}(S)$ ou encore $ca(S) = \exists\tilde{y}\ \mathcal{C}(S)$.*

On appelle $\mathcal{D}$-contrainte réponse la contrainte $ca(S)$ associée à un $\mathcal{D}$-squelette réponse S.

Remarque 2.5
1. Les variables libres de $ca(S)$ sont les variables libres du but B, sans renommage.
2. Les variables auxiliaires, qui sont connues à un renommage près, sont quantifiées existentiellement. On peut donc leur appliquer un renommage sans changer la signification de la contrainte résultat.
3. Les notions de squelette fini de racine B et de contrainte associée sont indépendantes de $\mathcal{D}$. Ceci sera exploité au paragraphe 2.11 (réponses générales).

Une valuation v solution de la $\mathcal{D}$-contrainte réponse $ca(S)$ rend vraie la formule $\exists\tilde{y}\ \mathcal{C}(S)$ quantifiée existentiellement sur l'ensemble $\tilde{y}$ des variables auxiliaires. Cette valuation doit donner des valeurs aux variables $\tilde{x}$ de la question, les valeurs attribuées aux autres variables ne nous important pas. Mais prouver que $\exists\tilde{y}\ \mathcal{C}(S)$ est vraie dans v, c'est prouver qu'on peut attribuer des valeurs aux variables $\tilde{y}$ de telle sorte que $\mathcal{C}(S)$ soit vraie, et donc que soient satisfaites toutes les contraintes de $cont(S)$. Il en ressort que $ca(S)$ est satisfiable dans $\mathcal{D}$ si et seulement si $cont(S)$ est sastisfiable dans $\mathcal{D}$ c'est-à-dire S est $\mathcal{D}$-soluble.

Donc si S est une $\mathcal{D}$-réponse, pour en trouver une solution que nous appellerons *$\mathcal{D}$-solution*, on cherche une valuation v qui prend en compte toutes les variables apparaissant dans la collection de contraintes $cont(S)$; la $\mathcal{D}$-solution correspondante est la restriction de v aux variables libres de la question.

Exemple 2.16 *Soit Fib-Fact le programme pré-interprété par $\mathcal{N}$:*
$fib(0,1) \leftarrow vrai$
$fib(1,1) \leftarrow vrai$
$fib(N, A+B) \leftarrow N > 1 \ \square\ fib(N-2, A), fib(N-1, B)$
$fact(0,1) \leftarrow vrai$
$fact(N, N*F) \leftarrow N > 0 \ \square\ fact(N-1, F)$
et B_8 le but
$\leftarrow X < 3 \ \square\ fib(X,Y), fact(X,Z).$
La question B_8 admet des $\mathcal{N}$-squelettes réponses, dont le squelette S_8 de la figure 11

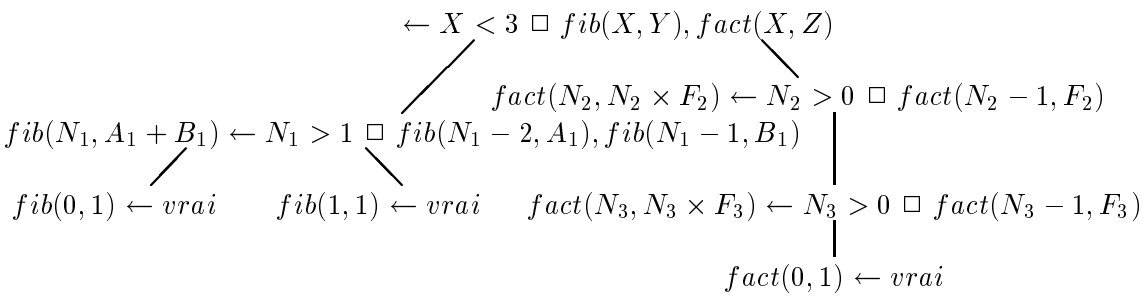

Figure 11 : Squelette S_8

dans lequel les clauses sont renommées pour faciliter la construction du système de contraintes.

La collection de contraintes associée est $cont(S_8) = \{X < 3, N_1 > 1, X = N_1, Y = A_1 + B_1, N_2 > 0, X = N_2, Z = N_2 * F_2, N_1 - 2 = 0, A_1 = 1, N_1 - 1 = 1, B_1 = 1, N_3 > 0, N_2 - 1 = N_3, F_2 = N_3 * F_3, N_3 - 1 = 0, F_3 = 1\}$.

Cont(S_8) *est satisfiable car vraie dans la valuation :* $\{X = 2, Y = 2, Z = 2, N_1 = 2, A_1 = 1, B_1 = 1, N_2 = 2, F_2 = 1, N_3 = 1, F_3 = 1\}$.

La partie intéressante de cette valuation est constituée des valeurs attribuées aux variables de la question, soit : $\{X = 2, Y = 2, Z = 2\}$.

Les autres variables : $N_1, A_1, B_1, N_2, F_2, N_3, F_3$ *sont quantifiées existentiellement dans la* $\mathcal{N}$*-contrainte réponse associée à* S_8 *:*

$\exists N_1 \exists A_1 \exists B_1 \exists N_2 \exists F_2 \exists N_3 \exists F_3 (X < 3 \wedge N_1 > 1 \wedge X = N_1 \wedge Y = A_1 + B_1 \wedge N_2 > 0 \wedge X = N_2 \wedge Z = N_2 * F_2 \wedge N_1 - 2 = 0 \wedge A_1 = 1 \wedge N_1 - 1 = 1 \wedge B_1 = 1 \wedge N_3 > 0 \wedge N_2 - 1 = N_3 \wedge F_2 = N_3 * F_3 \wedge N_3 - 1 = 0 \wedge F_3 = 1)$.

Elle implique : $X = 2 \wedge Y = 2 \wedge Z = 2$.

Exemple 2.17 *Pour le même programme Fib-Fact, le but :*

$\leftarrow \exists Y (X = Y * Y) \ \square \ fact(X, F)$

contient les variables libres X *et* F*, et la variable liée* Y*.* Y *ne sert que comme auxiliaire pour exprimer le fait que* X *est un carré. Cette question admet une infinité de* $\mathcal{N}$*-réponses, dont le squelette* S_9 *de la figure 12. La* $\mathcal{N}$*-contrainte réponse associée est :*

$\exists N_1 \exists F_1 (\exists Y (X = Y * Y) \wedge X = N_1 \wedge F = N_1 * F_1 \wedge N_1 > 0 \wedge N_1 - 1 = 0 \wedge F_1 = 1)$.

Elle implique $X = 1 \wedge F = 1$.

2.10 Correspondance entre squelettes réponses et arbres de preuve

Nous avons précédemment étudié la correspondance entre les squelettes, au sens primitif du terme : étiquetés par des classes d'équivalence de clauses, et les $\mathcal{D}$-arbres de preuve. L'extension de la notion de squelette autorisant la racine à être étiquetée par un but modifie un peu la correspondance avec les $\mathcal{D}$-arbres de preuve.

Étant donné un $\mathcal{D}$-squelette réponse S et une valuation v solution de $cont(S)$, l'extension $v_{\mathcal{D}}$ de v appliquée aux différents nœuds de S donne :

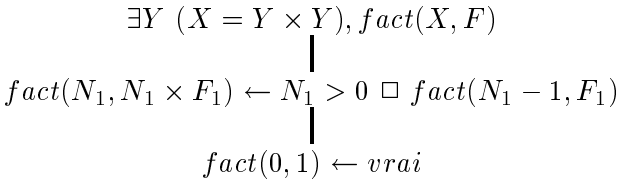

Figure 12 : Squelette S_9

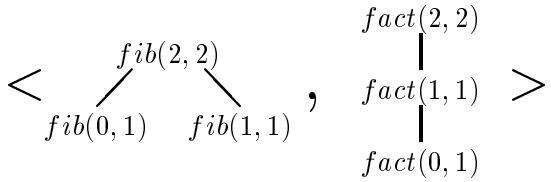

Figure 13 : Forêt de preuve F_1

- pour la racine : un $\mathcal{D}$-but avec contrainte vraie, donc une suite de $\mathcal{D}$-atomes
- pour tout autre nœud : une $\mathcal{D}$-clause avec contrainte vraie, donc une $\mathcal{D}$-règle. À tout nœud de ce type on associe la tête de règle.

On obtient ainsi un arbre A qui ressemble à un $\mathcal{D}$-arbre de preuve, à ceci près qu'à la racine on ne sait associer que la séquence de $\mathcal{D}$-atomes : $< v_{\mathcal{D}}(B_1), \ldots, v_{\mathcal{D}}(B_n) >$. Les n sous-arbres de A sont des $\mathcal{D}$-arbres de preuve ayant pour racines les $\mathcal{D}$-atomes $v_{\mathcal{D}}(B_1)$, ..., $v_{\mathcal{D}}(B_n)$. Donc A est assimilable à un n-uplet de $\mathcal{D}$-arbres de preuve, que nous appellerons *$\mathcal{D}$-forêt* de preuve. C'est cette $\mathcal{D}$-forêt de preuve que nous noterons $v_{\mathcal{D}}(S)$.

Exemple 2.18 *La réponse S_1 au programme Fib-Fact de l'exemple 2.16, associée à la $\mathcal{N}$-solution :*

$\{X = 2, Y = 2, Z = 2, N_1 = 2, A_1 = 1, B_1 = 1, N_2 = 2, F_2 = 1, N_3 = 1, F_3 = 1\}$

donne la $\mathcal{N}$-forêt de preuve F_1 de la figure 13.

Donc pour tout squelette fini $\mathcal{D}$-soluble S il existe une $\mathcal{D}$-forêt de preuve F dont S soit le squelette. Inversement, à partir de la $\mathcal{D}$-forêt de preuve F, on peut construire un squelette S de la manière suivante :

- pour chaque $\mathcal{D}$-arbre de preuve A_i de F on sait trouver un squelette $\mathcal{D}$-soluble S_i qui soit squelette pour A_i ; soit H_i la tête de la règle étiquetant la racine de S_i
- la racine de S, qui a ces S_i comme sous-arbres, est étiquetée par un but B : $\leftarrow vrai \ \square \ B_1, \ldots, B_n$ où les B_i sont des renommages des H_i utilisant des ensembles de variables disjoints.

Exemple 2.19 *La $\mathcal{N}$-forêt de preuve F_2 de la figure 14 admet pour squelette le squelette S_{10} de la figure 15.*

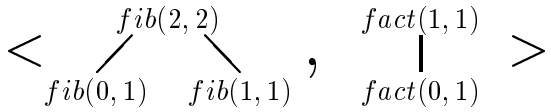

Figure 14 : Forêt de preuve F_2

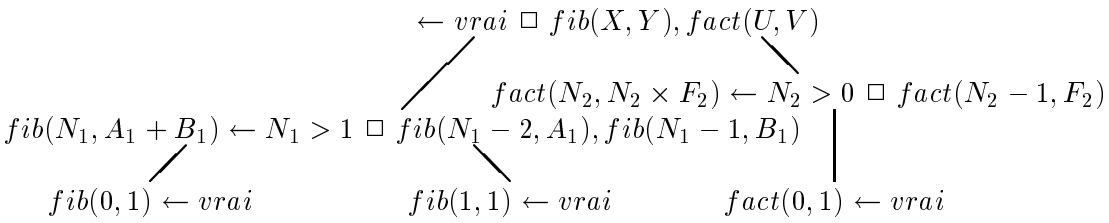

Figure 15 : Squelette S_{10}

2.11 Réponses générales

Un but B étant donné, les squelettes finis de racine B et les contraintes associées ne dépendent en aucune manière de la pré-interprétation sous-entendue $\mathcal{D}$. C'est ce que nous appelons réponses générales. Elles présentent un intérêt indéniable en vertu du théorème qui suit.

Définition 2.9 *(Réponse générale)*
On appelle réponse générale pour le but B et le programme P tout squelette fini de racine B. La contrainte associée à un tel squelette est une contrainte réponse générale.

Rappelons que B est un but de la forme $\leftarrow c \,\square\, B_1, \ldots, B_n$.
Nous noterons b le corps de B, soit $c \,\square\, B_1, \ldots, B_n$. Donc B s'écrit : $\leftarrow b$.

Théorème 2.1 *Si r est une contrainte réponse générale pour le programme P et le but $\leftarrow b$ alors $P \models r \rightarrow b$.*

Autrement dit, l'implication $r \rightarrow b$ est conséquence logique du programme.

Preuve : $P \models r \rightarrow b$ signifie : dans toute pré-interprétation $\mathcal{M}$, dans tout $\mathcal{M}$-modèle I de P, pour toute valuation v dans $\mathcal{M}$ solution de r, $v_I(b) = vrai$, c'est-à-dire : $v_{\mathcal{M}}(c) = vrai$ et chaque $v_{\mathcal{M}}(B_i)$ est dans I.

v et I étant indépendants l'un de l'autre on peut intervertir leurs rôles : pour toute pré-interprétation $\mathcal{M}$ et toute valuation v dans $\mathcal{M}$, si $v_{\mathcal{M}}(r) = vrai$ alors $v_{\mathcal{M}}(c) = vrai$ et chaque $v_{\mathcal{M}}(B_i)$ est dans tous les $\mathcal{M}$- modèles de P, ce qui revient à dire que chaque $v_{\mathcal{M}}(B_i)$ est dans le plus petit $\mathcal{M}$-modèle de P : $M^P_{\mathcal{M}} = Rac^P_{\mathcal{M}}$.

Finalement, $P \models r \rightarrow b$ équivaut à : dans toute pré-interprétation $\mathcal{M}$, pour toute valuation v dans $\mathcal{M}$ solution de r, $v_{\mathcal{M}}(c) = vrai$ et chaque $v_{\mathcal{M}}(B_i)$ est dans $Rac^P_{\mathcal{M}}$.

Soit donc r une contrainte réponse générale, $\mathcal{M}$ une pré-interprétation et v une valuation dans $\mathcal{M}$ solution de r. Montrons que $v_{\mathcal{M}}(c) = vrai$ et que $v_{\mathcal{M}}(B_i) \in Rac^P_{\mathcal{M}} \forall i$.

Puisque r est une contrainte réponse générale, c'est la contrainte associée à un squelette fini de racine $\leftarrow b$. Notons $\tilde{x}$ l'ensemble des variables libres de b, $\tilde{y}$ l'ensemble des variables auxiliaires apparaissant dans S et quantifiées existentiellement dans r. r est de la forme $\exists\tilde{y}(c_1 \wedge \ldots \wedge c_k)$. Si v est une valuation sur $\tilde{x}$ telle que $v_{\mathcal{M}}(r) = vrai$, v se prolonge en une valuation v' sur $\tilde{x} \bigcup \tilde{y}$ telle que $v'_{\mathcal{M}}(c_1 \wedge \ldots \wedge c_k) = vrai$. La contrainte c fait partie des c_j, donc $v_{\mathcal{M}}(c) = v'_{\mathcal{M}}(c) = vrai$. D'autre part, $c_1 \wedge \ldots \wedge c_k$ représente la collection de contraintes $\{c_1, \ldots, c_k\}$ associée au squelette S, qui est donc $\mathcal{M}$-soluble avec pour solution v'. $v'_{\mathcal{M}}(S)$ est une $\mathcal{M}$-forêt de preuve dans laquelle les racines des $\mathcal{M}$-arbres de preuve sont les $v'_{\mathcal{M}}(B_i) = v_{\mathcal{M}}(B_i)$. Donc chaque $v_{\mathcal{M}}(B_i)$ est racine d'un $\mathcal{M}$-arbre de preuve ; il est donc dans $Rac^P_{\mathcal{M}}$.

Lorsque r est une contrainte réponse pour $\leftarrow b$, $r \rightarrow b$ est également une conséquence logique du complété P^* en vertu du théorème plus général suivant :

Théorème 2.2 *Pour toute contrainte c et tout but $\leftarrow b$, $P \models c \rightarrow b$ ssi $P^* \models c \rightarrow b$.*

Preuve :
- $P \models c \rightarrow b \Rightarrow P^* \models c \rightarrow b$ parce que $P^* \models P$.
- Inversement, supposons que $P^* \models c \rightarrow b$. Alors $\forall D \forall v \forall I$ $\mathcal{D}$-modèle de P^* $v_{\mathcal{D}}(c) = vrai \Rightarrow v_{\mathcal{D}}(b) \in I$ donc $\forall D \forall v$ $v_{\mathcal{D}}(c) = vrai \Rightarrow v_{\mathcal{D}}(b) \in$ plus petit $\mathcal{D}$-modèle de $P^* =$ plus petit modèle de $P \Rightarrow v_{\mathcal{D}}(b) \in$ tout $\mathcal{D}$-modèle de P. Donc $P \models c \rightarrow b$.

L'ensemble des $r \rightarrow b$ conséquences logiques de P est une excellente manière de définir la sémantique du programme P. Mais certaines des implications $r \rightarrow b$ conséquences de P sont inutiles : celles pour lesquelles la contrainte r est insatisfiable. Dans ce cas $P \models r \rightarrow b$ est vrai pour une raison triviale : parce que r est toujours fausse. Nous sommes donc amenés à *éliminer les $r \rightarrow b$ pour lesquelles on sait que la contrainte r n'a pas de solution.*

Nous pouvons éliminer les $r \rightarrow b$ pour lesquelles r est insatisfiable dans la préinterprétation naturelle $\mathcal{D}$. Nous obtenons ainsi les $\mathcal{D}$-réponses.

Mais pour modéliser les systèmes existants, il faudra envisager d'autres *critères de rejet*. En effet, tout interprète comprend un algorithme de satisfiabilité de contrainte qui le plus souvent, pour des raisons d'indécidabilité ou de complexité, ne traduit pas exactement la satisfiabilité dans $\mathcal{D}$. L'algorithme est parfois incomplet : pour certaines contraintes il ne sait pas si elles sont satisfiables ou insatisfiables. Dans ce cas sont rejetées les réponses dans lesquelles la contrainte est jugée insatisfiable par l'algorithme.

Enfin, nous serons amenés, pour des considérations logiques, à envisager une théorie $\mathcal{T}$ pour modéliser la satisfiabilité : la contrainte c est satisfiable dans tout modèle de $\mathcal{T}$ si sa clôture existentielle est conséquence logique de $\mathcal{T}$; elle est insatisfiable, et les réponses qui la comprennent sont rejetées, si sa négation est conséquence logique de $\mathcal{T}$.

3 Sémantique opérationnelle

Nous supposerons dans tout ce chapitre que nous avons à notre disposition un critère de rejet CR qui permet d'éliminer certaines des réponses générales. Nous insistons sur le fait qu'il ne s'agit pas de ne garder que les réponses pour lesquelles on sait que la contrainte est vraie dans telle ou telle circonstance ; il s'agit au contraire de *garder toutes les réponses sauf celles qui sont rejetées par le critère CR*. Ceci permet de rendre compte du fonctionnement d'un interprète qui donne certaines réponses assorties de la mention *peut-être*, signalant ainsi que l'algorithme de satisfiabilité ne répond ni *oui* ni *non* ; ces réponses sont incertaines en ce sens qu'elles pourraient être éliminées par un algorithme plus proche de la satisfiabilité dans $\mathcal{D}$.

Le critère de rejet agit sur les collections de contraintes : son rôle est de refuser les squelettes dont la collection de contraintes ne convient pas. Il est très général ; il n'est pas nécessairement effectif ; nous supposons seulement qu'il a les deux propriétés :

1. il rejette la contrainte $faux$,
2. il est monotone sur les collections de contraintes : si une collection A est rejetée et si B contient A alors B est également rejetée,

On remarque que toute collection contenant $faux$ est rejetée.

3.1 Réponses

Définition 3.1 *(Squelette rejeté, squelette acceptable, réponse)*
Un squelette S (fini ou infini) est rejeté si sa collection de contraintes $cont(S)$ est rejetée par le critère CR.
Un squelette est acceptable s'il n'est pas rejeté par CR.
Une réponse pour le but B et le programme P (selon le critère CR) est un squelette fini de racine B acceptable (selon CR).
Une contrainte réponse (selon CR) est la contrainte associée à un squelette réponse (selon CR).

3.2 Recherche des réponses

La notion de squelette partiel et le procédé de calcul des réponses sont largement inspirés de [8].

Il s'agit de décrire un procédé permettant de trouver toutes les réponses pour un programme P et un but B donnés. On peut envisager différentes méthodes qui consistent en : générer des squelettes finis, ne conserver que ceux qui sont acceptables et de racine B.

Pour être efficace, il faut construire des squelettes susceptibles d'être des réponses en faisant grandir un squelette déjà construit, soit vers le haut (méthode ascendante), soit vers le bas (méthode descendante), ou dans toute direction imaginable. Quelle que soit la méthode de progression choisie, on devra envisager des squelettes incomplètement construits, que nous appellerons squelettes partiels.

Un squelette partiel ne se différencie d'un squelette (complet) que par le fait que certaines de ses feuilles peuvent être étiquetées par un symbole spécial représentant l'indéterminé.

Définition 3.2 *(Squelette partiel)*
Un squelette partiel est un arbre dont les nœuds sont étiquetés par des clauses de P ou le symbole ? de telle sorte que

1. *si N est une feuille, son étiquette $E(N)$ est, soit le symbole ?, soit une clause de P d'arité 0 (un but),*
2. *si N n'est pas une feuille, alors N a n fils ($0 < n$) et son étiquette $E(N)$ est une clause d'arité n.*

Les *nœuds indéfinis* du squelette partiel S sont les feuilles de S étiquetées par ?. Les autres nœuds de S sont dits *définis*. On note $def(S)$ l'ensemble des nœuds définis de S, $indef(S)$ l'ensemble de ses nœuds indéfinis. Le squelette partiel S est dit *complet* s'il n'a pas de nœud indéfini, *incomplet* s'il a au moins un nœud indéfini.

Désormais, le mot squelette signifie squelette partiel ; un squelette au sens primitif du mot est dit *squelette complet*.

La collection de contraintes $cont(S)$ associée à un squelette (partiel) S se construit comme pour un squelette complet, mais en ignorant les nœuds indéfinis. À tout nœud N de S autre que la racine, on associe, comme dans un squelette complet, l'atome $atome(N)$ issu du père de N ; $atome(N)$ est défini pour tout N autre que la racine, même si N est indéfini.
Le squelette (partiel) S est acceptable si sa collection de contraintes $cont(S)$ n'est pas rejetée.

Exemple 3.1 *La figure 16 représente un squelette partiel SP_1 pour le programme Fib. Sa collection de contraintes est : $\{N < 4, N_1 > 1, N = N_1, R = A_1 + B_1, N_2 > 1, N_1 - 1 = N_2, B_1 = A_2 + B_2, vrai, N_2 - 2 = 0, A_2 = 1\}$. Elle n'est pas rejetée par le critère correspondant exactement à la satisfiabilité dans $\mathcal{N}$ car elle a pour solution la valuation : $\{N = 3, R = 3, N_1 = 3, A_1 = 1, B_1 = 2, N_2 = 2, A_2 = 1, B_2 = 1\}$.*

Définition 3.3 *(Sous-squelette)*
Étant donné deux squelettes de même racine S et S', S' est un sous-squelette de S si S' coïncide avec S sur $def(S')$.

Donc $def(S')$ est inclus dans $def(S)$, et S' est obtenu en rendant indéfinis certains nœuds de S ; les descendants d'un nœud devenu indéfini sont supprimés.

Exemple 3.2 *Le squelette partiel SP_1 de la figure 16 est un sous-squelette du squelette complet S_7 de la figure 10.*

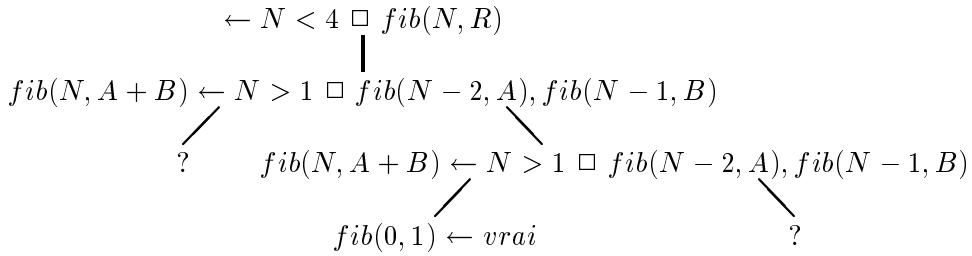

Figure 16 : Squelette partiel SP_1

Pour un squelette fixé S, tout sous-squelette S' de S est caractérisé par l'ensemble de ses nœuds définis. On peut en effet reconstituer S' à partir de S et de $def(S')$: les nœuds de $def(S')$ ont chacun un nombre de fils imposé par l'arité de la clause qui l'étiquette ; certains des fils sont dans $def(S')$; ceux qui ne sont pas dans $def(S')$ doivent être ajoutés sous forme de feuilles indéfinies. Mais $def(S')$ n'est pas une partie quelconque de $def(S)$: soit elle est vide, soit elle contient la racine et elle est connexe. Nous appellerons *convenable* une partie de $def(S)$ qui a cette propriété.

La construction de $cont(S)$ fait intervenir une fonction de choix des variantes de clauses pour les nœuds définis de S. Sa restriction à $def(S')$ peut servir à construire $cont(S')$. Donc, à un renommage près, $cont(S')$ est incluse dans $cont(S)$. Donc *tout sous-squelette d'un squelette acceptable est acceptable.*

3.3 Construction descendante

La construction descendante des réponses à la question posée par le but B consiste en la construction de squelettes obtenus par remplacement progressif de feuilles indéfinies par des squelettes, jusqu'à ce que l'on obtienne des squelettes complets. On ne construira que des squelettes finis acceptables et de racine B.

Supposons que nous ayons déjà construit un squelette S acceptable et de racine B. Deux cas se présentent :

1- S est complet, alors c'est une réponse,

2- S est incomplet, alors il a au moins une feuille indéfinie. Dans ce cas, on progresse vers une réponse en remplaçant l'une des feuilles indéfinies F par un squelette (complet ou incomplet) T, sous la condition que le nouveau squelette S' soit acceptable. C'est ce que nous appellerons *greffer* T sur F.

La construction de $cont(S')$ fait intervenir un choix C' sur les variantes de clauses pour les nœuds définis de S'. Les nœuds définis de S en font partie ; un choix de variantes C leur a été appliqué pour construire $cont(S)$. Il existe une fonction de choix C' sur S' qui prolonge C : il suffit de choisir pour les nœuds définis de T des variantes de clauses qui n'ont pas de variables communes avec les variantes déjà présentes dans C. Donc, à un renommage près des variables, $cont(S')$ comprend $cont(S)$ et des contraintes supplémentaires :

- celles de $cont(T)$,
- la contrainte d'égalité $atome(F) = tête(C'(racine(T)))$.

Pour avoir le plus de chances de trouver un squelette S' acceptable, il faut choisir pour T un squelette le plus petit possible. Or la racine de T comprend nécessairement une clause. Donc T est le plus petit possible s'il se résume à un squelette de clause défini comme suit.

Si C est la clause $A \leftarrow c \square B_1, \ldots, B_n$, le *squelette de la clause* C noté $sq(C)$ comprend :

- la racine étiquetée par C,
- n fils tous étiquetés par ?.

Si C est un fait ($n = 0$), $sq(C)$ est complet et se résume à la racine étiquetée par $A \leftarrow c$. Si C est d'arité non nulle, $sq(C)$ est incomplet.

Si B est le but $\leftarrow c \square B_1, \ldots, B_n$, le squelette du but B noté $sq(B)$ comprend :

- la racine étiquetée par B,
- n fils tous étiquetés par ?.

Choisissons donc pour T le squelette $sq(C)$ d'une clause C de P. Alors $cont(T)$ se résume à la contrainte c de la clause C (moyennant renommage). D'autre part, la contrainte d'égalité $atome(F) = tête(C)$ (toujours moyennant renommage) est $faux$ si ces deux atomes ne comportent pas le même symbole de prédicat. Il est donc inutile d'essayer de greffer en F un $sq(C)$ où $tête(C)$ ne comporte pas le même symbole de prédicat que $atome(F)$.

Il reste à définir le démarrage de la construction descendante. Le plus petit squelette initial possible est $sq(B)$. Celui-ci n'est pas nécessairement acceptable, il faut donc vérifier au départ que la contrainte du but B n'est pas rejetée.

Nous pouvons maintenant définir un procédé général de résolution descendante non déterministe qui, étant donné un but B, fabrique des suites de squelettes finis acceptables de racine B que l' on appelle états. Nous appellerons ce procédé résolution SLD.

3.4 Résolution SLD

1. Si la contrainte de B est rejetée, il n'y a aucune réponse à la question ; le calcul s'arrête immédiatement. Dans le cas contraire, on considère l'état initial : le squelette $sq(B)$.

2. À partir d'un état S,

   - si S est complet, c'est une réponse calculée,

   - si S est incomplet, on peut passer de S à tout squelette acceptable S' obtenu de la manière suivante : soit F une feuille indéfinie de S et C une clause de P qui a même symbole de prédicat que $atome(F)$; S' est obtenu en greffant $sq(C)$ sur F. Rappelons que S' doit être acceptable. On dit que S' *dérive de* S et plus précisément que S' dérive de S par la feuille F ; on appelle étape de dérivation le passage de S à S'.

S est un état final s'il n'existe aucun autre état qui en dérive. Ce peut être un état final *succès* : S est complet, ou un état final *échec* : S est incomplet mais pour chacune de ses feuilles indéfinies F, il n'existe pas de clause dont la tête comporte le même symbole de prédicat que $atome(F)$, ou toutes les clauses mises en correspondance avec les feuilles indéfinies produisent des collections de contraintes rejetées.

On appelle *dérivation SLD* toute suite finie ou infinie d'états S_1, S_2, ... , S_i, ... telle que $S_1 = sq(B)$, tout état sauf le premier dérive du précédent, et la suite est infinie ou se termine sur un état final. La dérivation est un succès si elle est finie et son dernier état est un squelette complet S ; dans ce cas on dit qu'elle *calcule la réponse* S. C'est un échec si elle est finie et son dernier état est un état final échec.

Exemple 3.3 *La figure 17 représente une dérivation succès pour le but* $\leftarrow N < 4 \ \square \ fib(N, R)$ *et le programme* Fib *selon le critère de satisfiabilité dans* $\mathcal{N}$. *Pour le même programme et le but* $\leftarrow N < 3 \ \square \ fib(N, 3)$, *la figure 18 représente une dérivation échec, parce que la collection de contraintes déjà construite impose* $N = 2$, *donc* $N - 1 = 1$; *donc la seule clause que l'on puisse espérer mettre à la place de ? est* $fib(1, 1) \leftarrow vrai$; *mais cela aboutit indirectement à* $3 = 1 + 1$.

Exemple 3.4 *Le but* $\leftarrow vrai \ \square \ p(1, R)$ *pour le programme de l'exemple 2.14 admet une dérivation infinie représentée figure 19.*

Remarque 3.1 *Le procédé indiqué est doublement non déterministe : on a le choix d'une feuille parmi les feuilles incomplètes, et pour une feuille incomplète donnée d'une clause parmi les clauses possibles.*

Remarque 3.2 *On peut optimiser la construction de la collection de contraintes associée à chaque squelette, nécessaire pour vérifier l'acceptabilité, en la bâtissant progressivement. Dans ce cas, un état est un couple* $< S, E >$ *où* S *est un squelette de racine* B, E *sa collection de contraintes* $cont(S)$. *L'état initial est* $< sq(B), \{contrainte(B)\} >$, *et on passe de* $< S, E >$ *à* $< S', E' >$ *en*

- *choisissant une feuille indéfinie* F *de* S,
- *choisissant dans* P *une clause* C *dont la tête a le même symbole de prédicat que* $atome(F)$,
- *choisissant une variante* C' *de* C *n'ayant pas de variable commune avec* S,
- *vérifiant que* $E' = E \bigcup \{contrainte(C'), \ atome(F) = tête(C')\}$ *n'est pas rejetée,*
- *construisant* S' *par greffe de* $sq(C)$ *sur* F.

Théorème 3.1 *Le procédé de résolution SLD associé à un critère de rejet donné a les propriétés suivantes :*

1. *toute réponse calculée est une réponse,*
2. *il calcule toutes les réponses.*

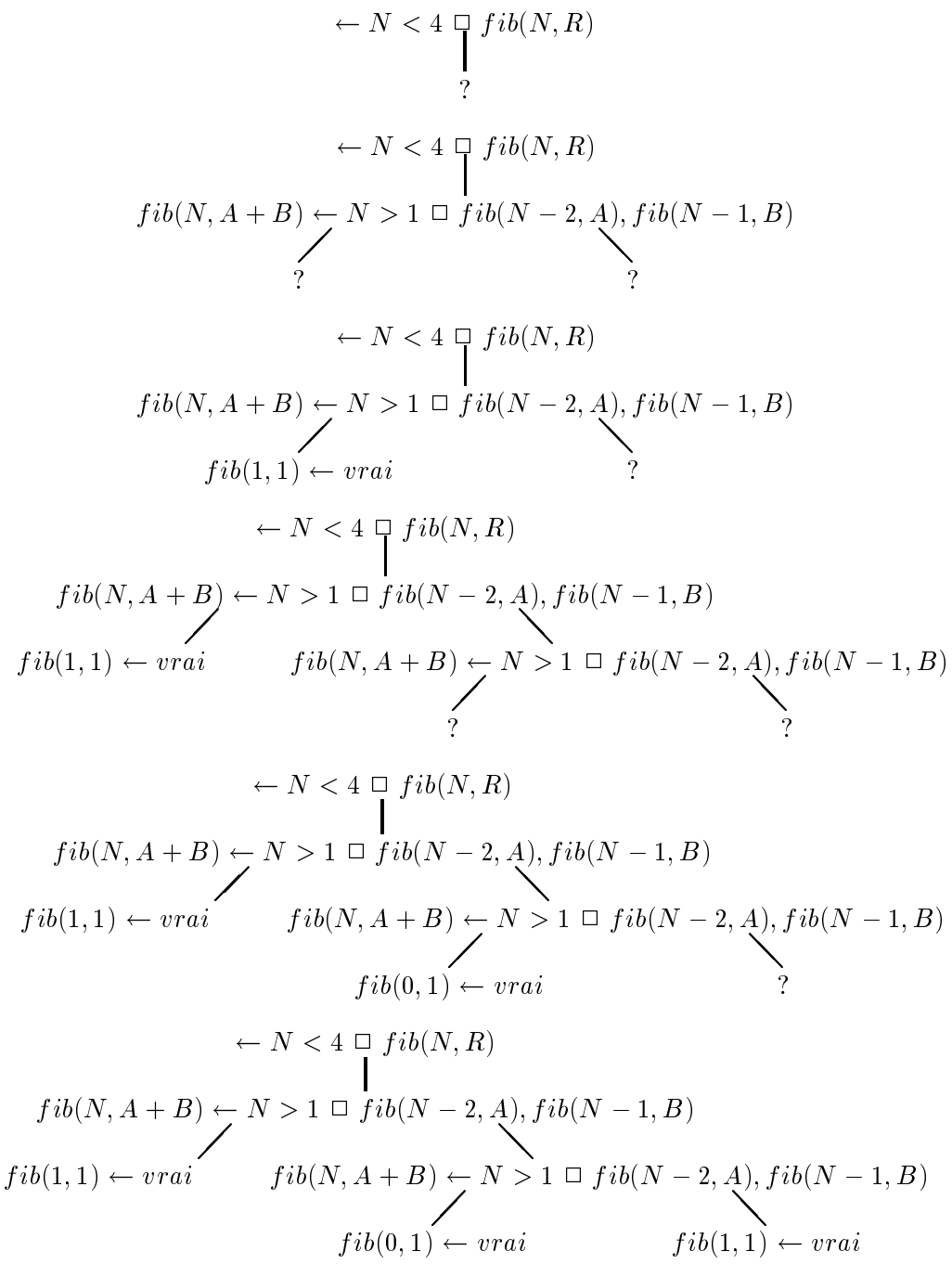

Figure 17 : Une dérivation succès

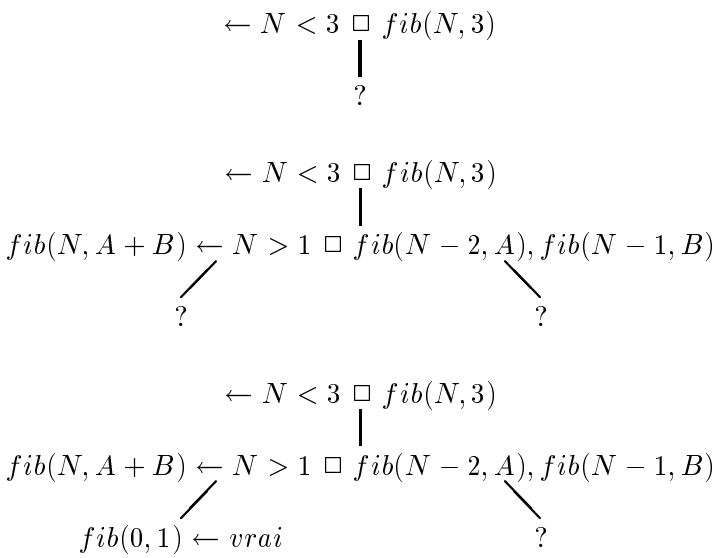

Figure 18 : Une dérivation échec

Preuve : La première proposition est évidente : la résolution SLD ne construit que des squelettes finis acceptables de racine B. Donc ses réponses calculées sont des squelettes finis complets acceptables et de racine B : ce sont des réponses.

Pour prouver la seconde proposition, considérons un squelette S fini, complet, acceptable et de racine B. Il nous faut prouver que c'est la réponse calculée par une dérivation. Considérons un parcours en largeur de l'arbre S. Cela donne une numérotation N_1, N_2, ..., N_p des nœuds de S telle que toute partie initiale $\{N_1, ..., N_i\}$ est une partie convenable de nœuds (tous définis) de S. À cette suite de nœuds on associe la suite de squelettes S_1, S_2, ..., S_p ainsi définie : S_i est le sous-squelette de S tel que $def(S_i) = \{N_1, ..., N_i\}$. La suite S_1, S_2, ..., S_p est une dérivation succès qui donne la réponse S. En effet, S_1 est le squelette initial $sq(B)$. S_p a les mêmes nœuds définis que S : c'est S lui- même. Enfin, le passage de S_i à S_{i+1} remplace l'étiquette ? de N_{i+1} par la clause correspondante dans S, et N_{i+1} est muni de ses fils tous étiquetés ?. C'est exactement l'opération de greffe. Il reste à vérifier que cette greffe est autorisée, c'est-à-dire que S_{i+1} est acceptable. Or S_1, S_2, ..., S_p sont des sous-squelettes du squelette acceptable S. Ils sont donc tous acceptables.

3.5 Règle de sélection

Le procédé de résolution SLD décrit ci-dessus comporte un double indéterminisme : sur la feuille indéfinie sur laquelle on greffe le squelette d'une clause, sur la clause dont on greffe le squelette. Pour se rapprocher d'un procédé de résolution effectif, il faut réduire l'indéterminisme. On peut dans ce but fixer une règle de choix de la feuille indéterminée ; il ne restera plus qu'à explorer toutes les solutions possibles engendrées par les clauses qui conviennent pour une feuille donnée.

On appelle *règle de sélection* une fonction R qui, pour tout squelette incomplet S

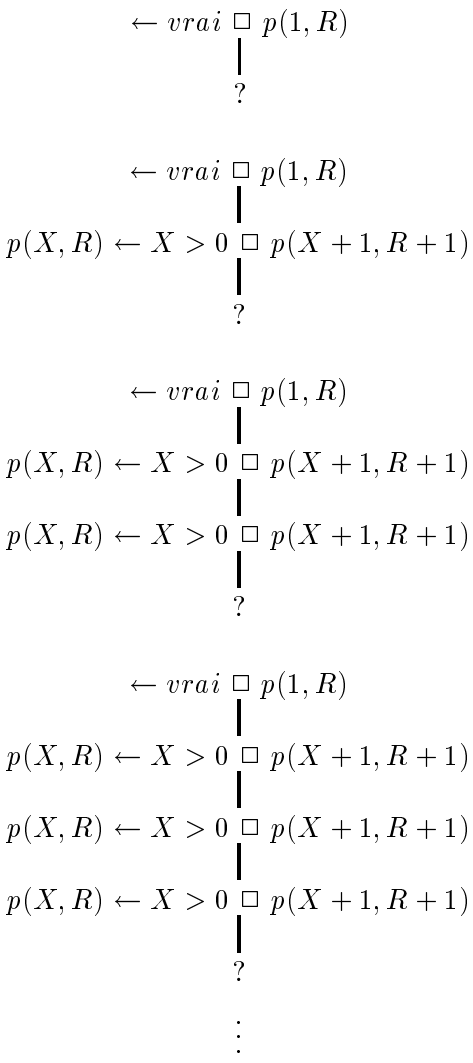

Figure 19 : Une dérivation infinie

rencontré dans une dérivation donne une feuille indéfinie F.

Étant donné une règle de sélection R (et un critère de rejet), la résolution SLD contrôlée par R, ou R-*résolution* se définit de la manière suivante :

1. Si la contrainte de B est rejetée, il n'y a aucune réponse à la question ; le calcul s'arrête immédiatement. Dans le cas contraire, on considère l'état initial : le squelette $sq(B)$.

2. À partir d'un état S,

   - si S est complet, c'est un état final succès et S est une réponse calculée,

   - si S est incomplet, on peut passer de S à tout squelette acceptable S' obtenu de la manière suivante : soit F la feuille indéfinie $R(S)$ et C une clause de P qui a même symbole de prédicat que $atome(F)$; S' est obtenu en greffant $sq(C)$ sur F. On dit que S' dérive de S selon R et que le passage de S à S' est une R-étape de dérivation.

   Les dérivations correspondantes sont des R-dérivations.

Exemple 3.5 *La première dérivation de l'exemple 3.3 représentée figure 17 est contrôlée par la règle qui sélectionne la première feuille indéfinie dans le parcours en largeur.*

Théorème 3.2 *La R-résolution a les deux propriétés :*

1. *toutes les réponses calculées sont des réponses,*

2. *elle calcule toutes les réponses.*

Preuve : La première proposition est évidente : la R-résolution n'est qu'un cas particulier de la résolution SLD définie précédemment.

La seconde proposition se prouve de la manière suivante : pour toute réponse S il existe une R-dérivation succès qui a S pour état final. On construit une suite de sous-squelettes de S : $S_1 = sq(B)$, S_2, ..., S_i, S_{i+1}, ... de la manière suivante : S_{i+1} est le sous-squelette de S tel que $def(S_{i+1}) = def(S_i) \bigcup \{R(S_i)\}$. Il est clair que S_{i+1} résulte de la greffe du squelette de clause défini par S sur la feuille indéfinie $F_i = R(S_i)$. Les squelettes successifs S_1, S_2, ... sont des sous-squelettes de S qui est acceptable, ils sont donc acceptables. D'autre part, S_i a exactement i nœuds définis ; donc, si S a p nœuds (qui sont tous définis), $S_p = S$. La dérivation S_1, ..., S_p calcule la réponse S.

3.6 Arbre de recherche SLD

La définition des arbres de recherche SLD est très analogue à celle de [14].

Définition 3.4 *(Arbre SLD)*
Un arbre de recherche *SLD (ou plus simplement arbre SLD) pour le but B, est un arbre A éventuellement infini tel que :*

- *les nœuds de A sont étiquetés par des squelettes finis acceptables de racine B (pas nécessairement complets),*

- *la racine de A est étiquetée par le squelette initial $sq(B)$,*
- *un nœud étiqueté par une réponse (un squelette complet) n'a pas de fils, c'est un nœud succès,*
- *pour tout nœud N étiqueté par un squelette incomplet S, l'une des feuilles indéfinies de S est marquée : c'est la feuille choisie F, et les nœuds fils de N correspondent exactement aux squelettes qui dérivent de S par la feuille F (tous les squelettes acceptables obtenus en greffant un squelette de clause sur F); ces nœuds fils sont étiquetés par ces squelettes. Le nœud N peut ne pas avoir de fils, c'est alors un nœud échec.*

Les branches de l'arbre de recherche SLD correspondent à des dérivations de B. Les branches finies correspondent à des dérivations finies, succès ou échec. Les branches infinies correspondent à des dérivations infinies.

Exemple 3.6 *La figure 20 représente un arbre SLD pour le programme Fib et le but $\leftarrow N < 3 \ \square \ fib(N, 1)$. S_1 et S_2 sont des succès, S_4 est un échec.*

Théorème 3.3 *Les branches succès d'un arbre SLD pour un but B correspondent aux réponses pour B.*

Preuve : Soit A un arbre SLD pour B.

- Toute branche succès de A correspond par définition à une réponse pour B.
- Inversement, soit S une réponse pour B. On construit alors une dérivation composée de sous-squelettes de S qui sont étiquettes de nœuds de A :

  $S_1 = sq(B)$: c'est l'étiquette de la racine de A,

  S_{i+1} est obtenu à partir de S_i, supposé être incomplet et acceptable, et étiqueter un nœud N de A, en greffant sur la feuille choisie en N le squelette de clause utilisé dans S ; S_{i+1} est acceptable car sous-squelette de S ; il est nécessairement étiquette d'un nœud fils de N dans A. Cette dérivation, qui correspond à une branche de A, atteint la réponse S en un nombre fini d'étapes : elle calcule S.

Définition 3.5 *(Arbre SLD selon R)*
Étant donné une règle de sélection R, l'arbre SLD du but B selon R est l'arbre SLD dans lequel, en tout nœud étiqueté par un squelette incomplet, la feuille indéfinie choisie est celle qui est désignée par R. L'arbre SLD selon R a des branches succès, des branches échec et des branches infinies.

Théorème 3.4 *Les branches succès ne dépendent pas de la règle de sélection R : il y a une bijection entre les branches succès de deux arbres de recherche pour le même but B avec deux règles de sélection différentes.*

Preuve : Les branches succès correspondent aux réponses pour B, qui sont définies indépendamment de la règle de sélection.

Les branches succès correspondant à une même solution dans deux arbres de recherche différents ont la même longueur : le nombre de nœuds de ce squelette solution.

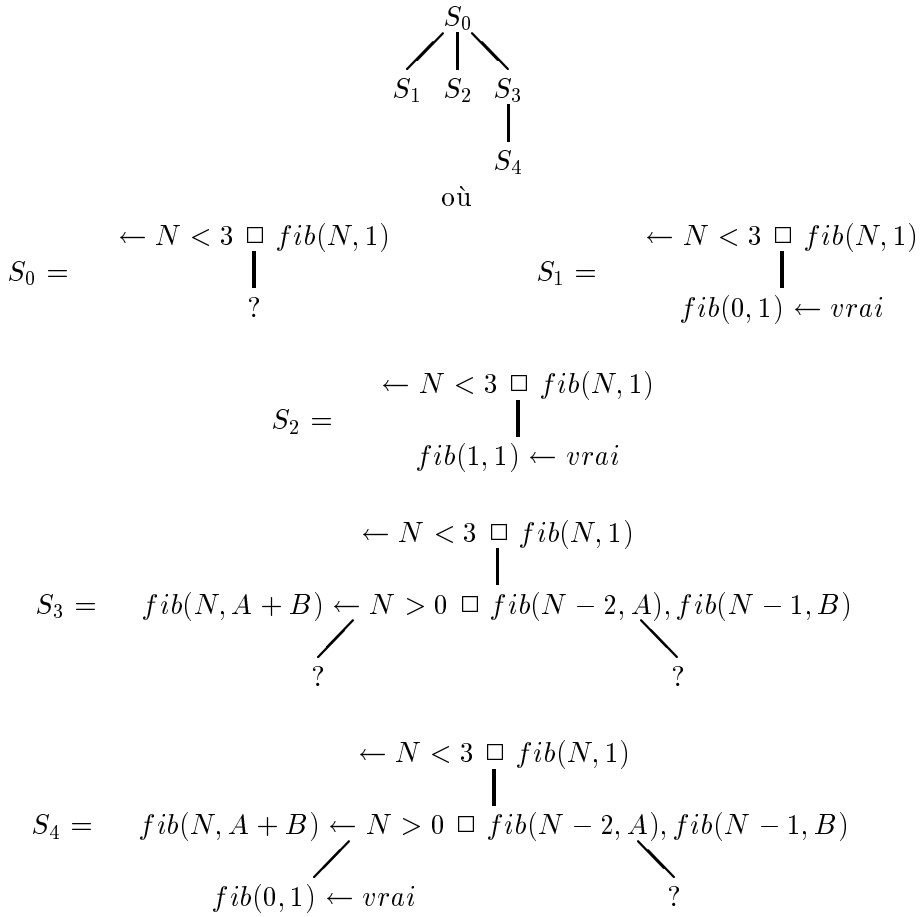

Figure 20 : Un arbre de recherche SLD

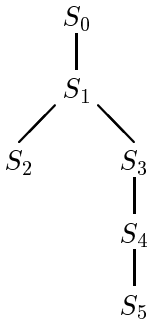

Figure 21 : Un arbre SLD d'échec fini

Définition 3.6 *(échec fini)*
Un but B est dit d'échec fini s'il a un arbre SLD d'échec fini, c'est-à-dire un arbre SLD dans lequel toutes les branches sont des échecs (elles sont donc finies).

Exemple 3.7 *Pour le programme Fib, le but $\leftarrow N < 4 \ \square \ fib(N, 4)$ admet l'arbre SLD d'échec fini représenté figures 21 et 22.*

Définition 3.7 *(Dérivation équitable, arbre SLD équitable)*
Une dérivation est dite équitable si elle est finie, ou si, pour tout état, toute feuille indéfinie est choisie au bout d'un nombre fini d'étapes. Un arbre SLD est équitable si toutes les branches infinies correspondent à des dérivations équitables.

Définition 3.8 *(Squelette finiment acceptable)*
Un squelette S est dit finiment acceptable si tout sous-squelette fini de S est acceptable.

Remarquons que les squelettes finis acceptables sont finiment acceptables.

Le théorème et le lemme qui suivent constituent un résultat nouveau concernant les branches non échec des arbres SLD. Le problème des échecs finis est étudié dans [2, 14] et [8] pour la programmation logique (sans contraintes). Il est étudié par [12] pour la programmation logique avec contraintes. La notion de squelette infaillible de [8] ne donnant pas satisfaction, nous proposons celle de squelette finiment acceptable.

Théorème 3.5 *(Branches non échec des arbres SLD équitables)*
Les branches non échec des arbres SLD équitables pour B correspondent aux squelettes complets finiment acceptables de racine B.

Preuve :

- Considérons une branche non échec d'un arbre SLD équitable pour B. C'est une branche succès ou une branche infinie.
  - Si c'est une branche succès, elle calcule un succès, c'est-à-dire un squelette fini complet acceptable.

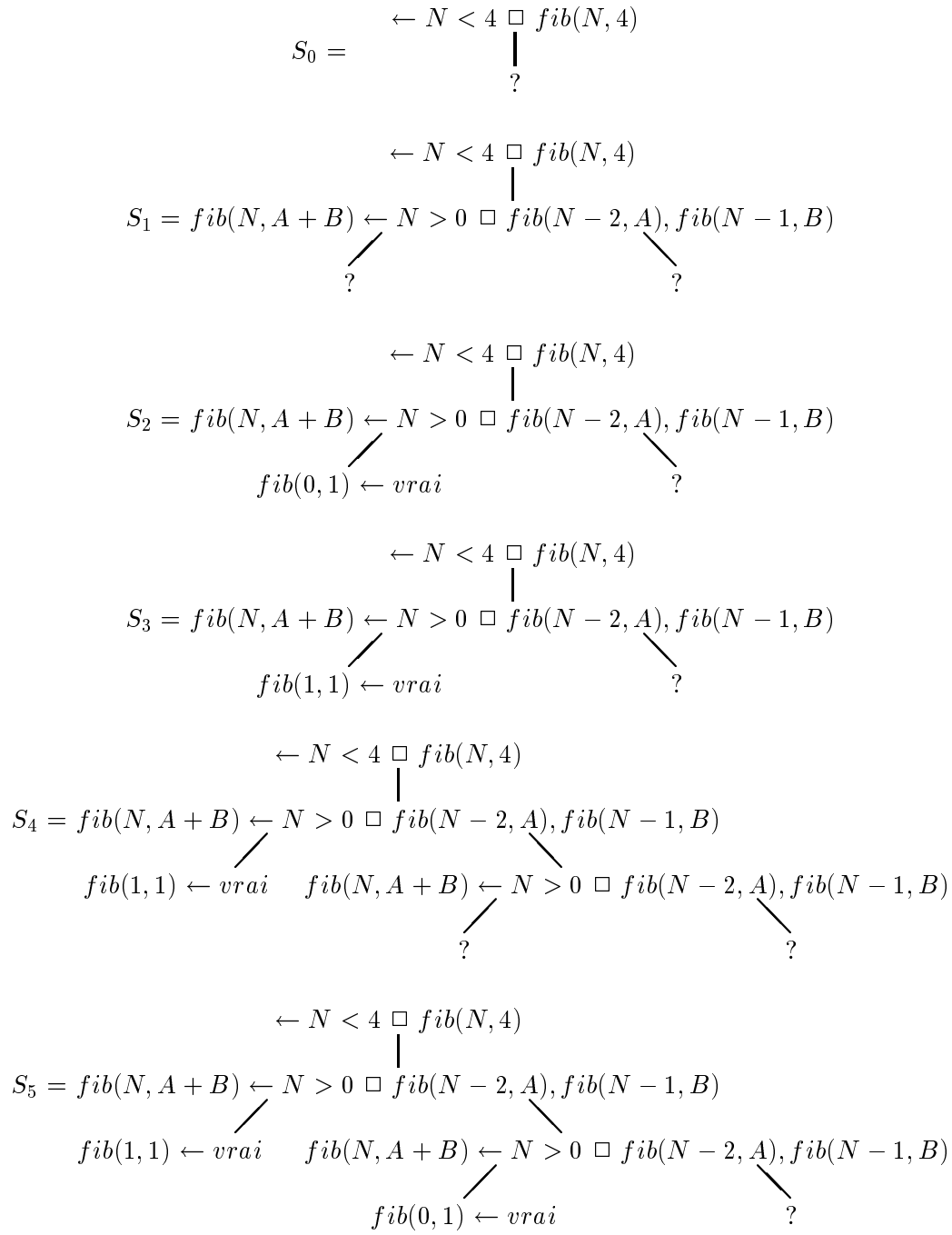

Figure 22 : Squelettes de l'arbre SLD de la figure 21

– Si c'est une branche infinie, c'est une dérivation équitable S_1, S_2, ..., S_i, S_{i+1}, ... composée de squelettes finis incomplets acceptables. S_i est sous-squelette de S_{i+1}. Le squelette S qui a pour ensemble de nœuds la réunion des ensembles de nœuds des S_i est infini et complet : la dérivation étant équitable, toute feuille indéfinie d'un S_i est choisie et devient donc un nœud défini au bout d'un nombre fini d'étapes. Tout sous-squelette fini de S est inclus dans l'un des S_i donc acceptable.

- Inversement, montrons que pour tout squelette complet S de racine B finiment acceptable et tout arbre SLD équitable A pour B, S est calculé par l'une des branches de A.

  – Si S est fini, c'est une réponse pour B. Donc S correspond à une branche succès de A.

  – Si S est infini, nous allons montrer qu'il est calculé par une branche infinie de A (qui correspond à une dérivation équitable). Considérons la suite de sous-squelettes de S :

  - $S_1 = sq(B)$ (qui étiquette la racine de A)

  - supposons que sont déjà construits S_1, ..., S_i et que S_i est un sous-squelette de S, incomplet et acceptable, étiquetant un nœud N_i de A ; soit F_i la feuille indéfinie de S_i choisie en N_i dans A. Alors S_{i+1} est obtenu en greffant sur F_i la clause utilisée dans S. Il est évident que S_{i+1} est aussi un sous-squelette de S ; il est acceptable (car sous-squelette fini de S) et incomplet (car sous-squelette strict de S). D'après la définition des arbres SLD, c'est l'étiquette d'un nœud N_{i+1} fils du nœud N_i dans A. Nous venons de déterminer une branche infinie de A ; elle correspond à une dérivation équitable, donc elle calcule un squelette complet qui est sous-squelette de S ; le seul sous-squelette complet de S est S lui-même.

Lemme 3.1 *À tout squelette S complet finiment acceptable de racine B correspond une branche non échec dans tout arbre SLD pour B ; si S est fini c'est une branche succès, si S est infini c'est une branche infinie.*

Preuve : Soit S un squelette complet finiment acceptable de racine B et A un arbre SLD pour B.

- Si S est fini, c'est une réponse qui est calculée par une branche succès de A.

- Si S est infini, on construit la suite de sous-squelettes de S étiquetant des nœuds de A :

  $S_1 = sq(B)$ (étiquette de la racine de A),

  S_{i+1} est construit à partir de S_i, supposé être incomplet et étiquette du nœud N de A, en greffant sur la feuille choisie en N le squelette de clause utilisé dans S.

  Tous les squelettes S_i sont acceptables car sous-squelettes finis de S. Ils sont incomplets car sous-squelettes stricts de S. La suite ainsi définie constitue une branche infinie de A.

On retrouve le résultat classique de la programmation logique :

Théorème 3.6 *(Arbres d'échec fini)*
Si B a un arbre d'échec fini, alors tout arbre SLD équitable pour B est d'échec fini.

Preuve : Soit A_1 un arbre d'échec fini pour B et A_2 un arbre SLD équitable pour B. Alors A_2 n'a aucune branche non échec, donc n'a que des branches échec, donc est d'échec fini. En effet, si A_2 avait une branche non échec, celle-ci correspondrait à un squelette S complet finiment acceptable de racine B. Le lemme ci-dessus assure que tout arbre SLD pour B comprend une branche non échec correspondant à S. Donc A_1 comprendrait une branche non échec, donc A_1 ne serait pas d'échec fini.

4 Interprétation en logique

Nous retrouvons dans ce chapitre les résultats classiques concernant, d'une part l'existence d'une réponse pour un but donné, d'autre part le lien entre les contraintes réponses et les conséquences logiques du programme selon une pré-interprétation ou une théorie. Ces résultats sont énoncés à peu près sous la même forme dans [12, 13] et [16]. Nous généralisons l'étude en envisageant une théorie non satisfaction-complète, et un critère de rejet en relation avec une théorie. La propriété d'indépendance des contraintes négatives est définie dans [16]. Nous expliquons pourquoi elle est vérifiée en programmation logique classique.

Nous supposons toujours que la notion de réponse pour un programme P et un but $\leftarrow b$ dépend d'un critère de rejet CR. Nous exprimons ici en termes de logique les résultats obtenus sur les réponses dans trois situations particulières :

- lorsque CR correspond exactement à l'insatisfiabilité dans la pré-interprétation "naturelle" $\mathcal{D}$; nous dirons que le critère retenu est $CR(\mathcal{D})$;
- lorsque CR correspond exactement à l'insatisfiabilité selon une théorie $\mathcal{T}$; nous dirons que le critère est $CR(\mathcal{T})$;
- lorsque CR ne correspond pas exactement mais est lié à l'insatisfiabilité selon la théorie $\mathcal{T}$: s'il ne rejette que des contraintes insatisfiables selon $\mathcal{T}$.

4.1 Réponses selon $\mathcal{D}$

Nous nous plaçons dans le cas où le critère de rejet est $CR(\mathcal{D})$: les squelettes acceptables sont les S tels que $cont(S)$ est satisfiable dans $\mathcal{D}$. Les réponses selon $CR(\mathcal{D})$ sont donc les squelettes finis complets $\mathcal{D}$-solubles, c'est-à-dire les $\mathcal{D}$-réponses. Nous avons remarqué que S fini et complet est $\mathcal{D}$-soluble ssi la contrainte associée $ca(S)$ est satisfiable dans $\mathcal{D}$.

Définition 4.1 *($\mathcal{D}$-conséquence)*
On dit que la formule F est une $\mathcal{D}$-conséquence *du programme P, ce qui est noté $P \models_{\mathcal{D}} F$, si F est vraie dans tout $\mathcal{D}$-modèle de P, c'est-à-dire si, dans tout $\mathcal{D}$-modèle I et toute valuation v, $v_I(F) = vrai$.*

Si F contient les variables libres $\tilde{x}$, seules interviennent dans le calcul de $v_I(F)$ les valeurs $v(\tilde{x})$. C'est ce que nous appellerons *valuation sur $\tilde{x}$*. Le cas où F est un atome fait l'objet du lemme qui suit.

Lemme 4.1 *($\mathcal{D}$-conséquences atomiques)*
Pour tout atome A élément de $ATOM_P$, $P \models_{\mathcal{D}} A$ ssi $v_{\mathcal{D}}(A) \in M_{\mathcal{D}}^P$ pour toute valuation v.

Preuve : $P \models_{\mathcal{D}} A$ ssi $v_I(A) = vrai$ pour tout $\mathcal{D}$-modèle I de P et toute valuation v, c'est-à-dire $v_{\mathcal{D}(\mathcal{A})} \in I$ pour tout $\mathcal{D}$-modèle I. Si $v_{\mathcal{D}}(A)$ est dans tous les $\mathcal{D}$-modèles de P, il est en particulier dans le plus petit : $M_{\mathcal{D}}^P$. Inversement, si $v_{\mathcal{D}}(A)$ est dans $M_{\mathcal{D}}^P$,

il est dans tout $\mathcal{D}$-modèle I de P (car I contient $M^P_{\mathcal{D}}$).

Lorsque la formule F ne fait intervenir comme symboles de prédicats que des éléments de Π_C, c'est-à-dire si F est une formule de $\mathcal{L}(V, \Sigma, \Pi_C)$ (en particulier si c'est une contrainte), sa valeur de vérité ne dépend pas d'un quelconque modèle de P ; elle ne dépend que de la pré-interprétation $\mathcal{D}$. On notera $\models_{\mathcal{D}} F$ le fait que F est vraie dans $\mathcal{D}$, c'est-à-dire $v_{\mathcal{D}}(F) = vrai$ pour toute valuation v dans $\mathcal{D}$.

Application aux réponses : une $\mathcal{D}$-réponse pour $\leftarrow b$ est un squelette fini complet S de racine $\leftarrow b$ tel que la contrainte associée $r = ca(S)$ est satisfiable dans $\mathcal{D}$; cela s'exprime par

$$non(\models_{\mathcal{D}} \neg r)$$

ou encore

$$\models_{\mathcal{D}} \tilde{\exists} r$$

car la négation de r n'est pas vraie dans $\mathcal{D}$ ssi il existe une valuation v dans laquelle r est vraie, ssi r est satisfiable dans $\mathcal{D}$ ssi $\tilde{\exists} r$ est vraie dans $\mathcal{D}$.

Théorème 4.1 *(Existence d'une réponse)*
Le but $\leftarrow b$ admet une $\mathcal{D}$-réponse pour le programme P si et seulement si $P \models_{\mathcal{D}} \tilde{\exists} b$.

Preuve : $\tilde{\exists} b$ est une formule close, donc sa valeur ne dépend pas d'une valuation, mais seulement de la pré-interprétation I. $\tilde{\exists} b$ est vraie dans I ssi il existe une valuation v sur les variables libres $\tilde{x}$ telle que $v_I(b) = vrai$, c'est-à-dire, si $b = c \square B_1, \ldots, B_n$, $v_{\mathcal{D}}(c) = vrai$ et $v_{\mathcal{D}}(B_i) \in I$ pour chaque i.

- Si B admet un $\mathcal{D}$-squelette réponse S, il existe une valuation v solution de $cont(S)$. Alors $v_{\mathcal{D}}(S)$ est une $\mathcal{D}$-forêt de preuve de racine $< v_{\mathcal{D}}(B_1), \ldots, v_{\mathcal{D}}(B_n) >$. Chaque $v_{\mathcal{D}}(B_i)$ est racine d'un $\mathcal{D}$-arbre de preuve, donc est dans $M^P_{\mathcal{D}} = Rac^P_{\mathcal{D}}$. Et $v_{\mathcal{D}}(c) = vrai$. Donc $v_{\mathcal{D}}(b)$ est vrai dans tout $\mathcal{D}$-modèle de P. Le calcul de $v_{\mathcal{D}}(b)$ ne fait intervenir qu'une partie v' de la valuation v : celle qui concerne les variables libres $\tilde{x}$. Alors $v'_{\mathcal{D}}(b)$ est vrai dans tout $\mathcal{D}$-modèle de P, donc $P \models_{\mathcal{D}} \tilde{\exists} b$.

- En sens inverse, si $P \models_{\mathcal{D}} \tilde{\exists} b$, pour tout $\mathcal{D}$-modèle I il existe une valuation v_I sur $\tilde{x}$ pour laquelle $v^I_{\mathcal{D}}(c) = vrai$ et $v^I_{\mathcal{D}}(B_i) \in I$. Soit v_0 la valuation v^I pour le plus petit $\mathcal{D}$-modèle $I = Rac^P_{\mathcal{D}}$. Alors chaque $v_{0\,\mathcal{D}}(B_i)$ est dans $Rac^P_{\mathcal{D}}$, donc a un $\mathcal{D}$-arbre de preuve A_i. Cet A_i admet au moins un squelette S_i. Soit H_i la tête de la règle étiquetant la racine de S_i. Soit S l'arbre construit de la manière suivante :

  - sa racine est étiquetée par le but $\leftarrow b$,

  - il a n sous-arbres : les squelettes S_1, ..., S_n.

  Cet arbre S est évidemment un squelette, et sa racine est $\leftarrow b$. Il reste à montrer qu'il est $\mathcal{D}$-soluble. La collection de contraintes $cont(S)$ comprend :

  - les $cont(S_i)$ portant sur des ensembles de variables $\tilde{y_i}$ disjoints entre eux et disjoints de $\tilde{x}$,

  - la contrainte c,

  - les contraintes d'égalité $H_i = B_i$.

Figure 23 : Squelettes S_{11} et S_{12}

Chaque S_i admet une valuation v_i portant sur l'ensemble de variables $\tilde{y}_i$ telle que $v_{i\mathcal{D}}(S_i) = A_i$. En particulier, $v_{i\mathcal{D}}(H_i) = v_{0\mathcal{D}}(B_i)$. Il est clair que la valuation sur $\tilde{x} \bigcup \tilde{y_1} \bigcup \ldots \bigcup \tilde{y_n}$ définie par v_0 sur $\tilde{x}$ et v_i sur $\tilde{y}_i$ satisfait $cont(S)$. Donc S est une $\mathcal{D}$-réponse.

Les $\mathcal{D}$-contraintes réponses pour un but donné ont la propriété suivante :

Théorème 4.2 *(Contraintes réponses et conséquences logiques)*
Si r est une $\mathcal{D}$-contrainte réponse pour le but $\leftarrow b$, alors $P \models_{\mathcal{D}} r \rightarrow b$.

Preuve : Ce n'est qu'un cas particulier du résultat énoncé à la fin de la seconde partie : si r est une contrainte réponse générale pour $\leftarrow b$ alors $P \models r \rightarrow b$. En effet,

- si c'est vrai pour les réponses générales, c'est en particulier vrai pour les $\mathcal{D}$-réponses ;

- si $r \rightarrow b$ est vraie dans toute pré-interprétation $\mathcal{M}$ et tout $\mathcal{M}$-modèle de P, $r \rightarrow b$ est en particulier vraie dans la pré-interprétation $\mathcal{D}$ et tout $\mathcal{D}$-modèle de P.

Remarque 4.1 *(importante)*
Il n'est pas vrai que toute contrainte c telle que $P \models_{\mathcal{D}} c \rightarrow b$ est une $\mathcal{D}$-contrainte réponse pour $\leftarrow b$. Considérons en effet le programme suivant avec $\mathcal{N}$ comme pré-interprétation :
$p(0) \leftarrow vrai$
$p(X) \leftarrow X > 0$
Le but $\leftarrow vrai \square p(X)$ admet les deux squelettes réponses S_{11} et S_{12} de la figure 23 auxquels correspondent les contraintes réponses :
$X = 0$,
$\exists X_1(X = X_1 \wedge X_1 > 0)$ *qui équivaut à* $X > 0$.
$X = 0 \rightarrow vrai \square p(X)$ et $X > 0 \rightarrow vrai \square p(X)$ sont bien des $\mathcal{N}$-conséquences de P. Mais $vrai \rightarrow vrai \square p(X)$ est aussi une $\mathcal{N}$-conséquence de P car, pour toute valuation v, $v_{\mathcal{N}}(p(X))$ est dans $Rac^P_{\mathcal{N}}$: si $v(X) = 0$ alors $v_{\mathcal{N}}(p(X)) = p(0)$ et $p(0)$ est prouvé par la règle $p(0) \leftarrow$; si $v(X) = n > 0, v_{\mathcal{N}}(p(X)) = p(n)$ qui est prouvé par la règle $p(n) \leftarrow$. Et pourtant vrai n'est pas une contrainte réponse pour le but $\leftarrow vrai \square p(X)$.

Il existe tout de même une réciproque à la proposition ci-dessus. Nous avons besoin pour l'énoncer d'une définition préliminaire.

Définition 4.2 *Soit c une contrainte et $\{c_i\}_{i \in I}$ un ensemble (éventuellement infini) de contraintes. Alors $\models_{\mathcal{D}} c \rightarrow \bigvee_{i \in I} c_i$ signifie : pour toute valuation v dans $\mathcal{D}$ solution de c il existe $i \in I$ tel que v soit solution de c_i. Lorsque l'ensemble I est fini, cela revient à dire que la formule $c \rightarrow \bigvee_{i \in I} c_i$ est vraie dans la pré-interprétation $\mathcal{D}$.*

Théorème 4.3 *(couverture des conséquences logiques par des contraintes réponses)*
Pour toute contrainte c, si $P \models_{\mathcal{D}} c \rightarrow b$ alors $\models_{\mathcal{D}} c \rightarrow \bigvee_{r \in R} r$ où R est l'ensemble (éventuellement infini) des $\mathcal{D}$-contraintes réponses pour $\leftarrow b$.

Preuve : Soit R l'ensemble des $\mathcal{D}$-contraintes réponses pour le but $\leftarrow b$. Soit $b = c' \square B_1, \ldots, B_n$. Pour toute valuation v solution de c, $v_{\mathcal{D}}(b)$ est vrai dans tous les $\mathcal{D}$-modèles de P, donc dans $Rac^P_{\mathcal{D}}$. En particulier $v_{\mathcal{D}}(c') = vrai$ et chaque $v_{\mathcal{D}}(B_i)$ est dans $Rac^P_{\mathcal{D}}$. Donc $v_{\mathcal{D}}(B_i)$ admet un $\mathcal{D}$-arbre de preuve A_i, qui a lui-même un squelette S_i. Le squelette S de racine $\leftarrow b$ qui a n sous-arbres : les S_i, est une réponse pour $\leftarrow b$ (voir la preuve de $P \models_{\mathcal{D}} \tilde{\exists} b \Rightarrow$ il existe une $\mathcal{D}$-réponse). La contrainte réponse associée r_v (quantifiée sur les variables auxiliaires autres que les variables libres de b) fait partie de l'ensemble R, et est vraie dans la valuation v. Ceci prouve que $\models_{\mathcal{D}} c \rightarrow \bigvee_{r \in R} r$.

Remarque 4.2 *(importante)*
Il n'existe pas toujours de couverture finie : un ensemble fini de contraintes réponses R' tel que $\models_{\mathcal{D}} c \rightarrow \bigvee_{r' \in R'} r'$.

Exemple 4.1 *Considérons le programme Ent :*
$ent(0) \leftarrow vrai$
$ent(X+1) \leftarrow vrai \square ent(X)$
pré-interprété par $\mathcal{N}$. Le but $\leftarrow vrai \square ent(X)$ admet pour $\mathcal{N}$-contraintes réponses :
r_0 : $X = 0$
r_1 : $\exists X_1 (X = X_1 + 1 \wedge X_1 = 0)$
...
r_n : $\exists X_1 \ldots \exists X_n (X = X_1 + 1 \wedge X_1 = X_2 + 1 \wedge \ldots \wedge X_{n-1} = X_n + 1 \wedge X_n = 0)$
...
La contrainte r_n est issue du $\mathcal{N}$-squelette réponse S_n de la figure 24.
r_n est telle que pour une valuation v, $v_{\mathcal{N}}(r_n) = vrai$ ssi $v(X) = n$.
Le programme Ent admettant les r_n comme $\mathcal{N}$-contraintes réponses, on a
$Ent \models_{\mathcal{N}} r_n \rightarrow vrai \square ent(X)$
pour tout entier naturel n. Ceci signifie simplement que $ent(n)$ est racine d'un $\mathcal{N}$-arbre de preuve (issu du squelette S_n).

On a aussi $Ent \models_{\mathcal{N}} vrai \rightarrow vrai \square ent(X)$ donc $Ent \models_{\mathcal{N}} ent(X)$
car toute valuation v associe à X un entier naturel n ; alors $v_{\mathcal{N}}(ent(X)) = ent(n)$, et nous venons de voir que $ent(n)$ est racine d'un $\mathcal{N}$-arbre de preuve, il est donc dans $Rac^P_{\mathcal{N}}$. Il est bien vrai que $\models_{\mathcal{N}} vrai \rightarrow \bigvee_{n \in \mathbb{N}} r_n$ car pour toute valuation v il existe n tel que $v_{\mathcal{N}}(r_n) = vrai$: $n = v(X)$.

Mais il n'y a pas de couverture finie. Considérons en effet une partie finie F de $\{r_n\}_{n \in N}$. Il existe une valuation v telle que $v_{\mathcal{N}}(r) = faux$ pour tout élément r de F : il suffit de donner à X une valeur qui n'est pas dans $\{n | r_n \in F\}$. Alors $\models_{\mathcal{N}} vrai \rightarrow \bigvee_{r \in F} r$ n'est pas vérifié.

Les conséquences positives de P et du complété P^* sont les mêmes :

Théorème 4.4
$P \models_{\mathcal{D}} \tilde{\exists} b$ *ssi* $P^* \models_{\mathcal{D}} \tilde{\exists} b$
$P \models_{\mathcal{D}} c \rightarrow b$ *ssi* $P^* \models_{\mathcal{D}} c \rightarrow b$

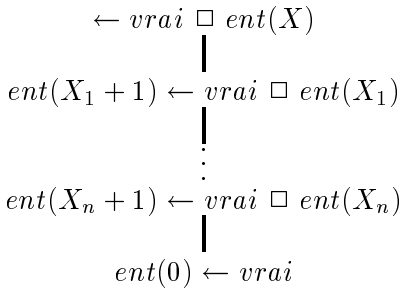

Figure 24 : Squelette S_n

Preuve :

- $P \models_{\mathcal{D}} \tilde{\exists} b \Rightarrow P^* \models_{\mathcal{D}} \tilde{\exists} b$ car si $\tilde{\exists} b$ est vrai dans les $\mathcal{D}$-modèles de P, il est en particulier vrai dans les $\mathcal{D}$-modèles de P^*.
- Inversement, soit $b = c \square B_1, \ldots, B_n$. Si $P^* \models_{\mathcal{D}} \tilde{\exists} b$ alors $\forall I$ $\mathcal{D}$-modèle de P^* $\exists v$ telle que

  $v_{\mathcal{D}}(c) = vrai$

  $v_{\mathcal{D}}(B_i) \in I \ \forall i$

  Dans le plus petit modèle de P^*, $M^P_{\mathcal{D}}$, $\exists v$ telle que

  $v_{\mathcal{D}}(c) = vrai$

  $v_{\mathcal{D}}(B_i) \in M^P_{\mathcal{D}} \ \forall i$

  Puisque $M^P_{\mathcal{D}}$ est en même temps le plus petit $\mathcal{D}$-modèle de P, $v_{\mathcal{D}}(B_i) \in$ tout $\mathcal{D}$-modèle de P. Donc $P \models_{\mathcal{D}} \tilde{\exists} b$.
- $P \models_{\mathcal{D}} c \rightarrow b \Rightarrow P^* \models_{\mathcal{D}} c \rightarrow b$ car tout $\mathcal{D}$-modèle de P^* est un modèle de P.
- Inversement, supposons que $P^* \models_{\mathcal{D}} c \rightarrow b$. Alors pour toute valuation v solution de c, $v_{\mathcal{D}}(b)$ est dans tout $\mathcal{D}$-modèle de P^*, donc dans le plus petit $M^P_{\mathcal{D}}$; donc $v_{\mathcal{D}}(b)$ est dans tout $\mathcal{D}$-modèle de P. Donc $P \models_{\mathcal{D}} c \rightarrow b$.

Les conséquences négatives de P^* sont en rapport avec la négation par l'échec.

Théorème 4.5 *Si le but $\leftarrow b$ est d'échec fini selon $\mathcal{D}$ alors $P^* \models_{\mathcal{D}} \neg b$.*

Preuve : Soit $b = c \square B_1, \ldots, B_n$. Pour montrer que $\leftarrow b$ d'échec fini selon $\mathcal{D}$ $\Rightarrow P^* \models_{\mathcal{D}} \neg b$ nous allons montrer l'implication opposée : $non(P^* \models_{\mathcal{D}} \neg b) \Rightarrow \leftarrow b$ n'est pas d'échec fini selon $\mathcal{D}$. Si $non(P^* \models_{\mathcal{D}} \neg b)$, il existe une $\mathcal{D}$-interprétation I modèle de P^* (c'est-à-dire point fixe de $T^P_{\mathcal{D}}$) et une valuation v telles que $v_I(b) = vrai$. Cela signifie :

$v_{\mathcal{D}}(c) = vrai$

$v_{\mathcal{D}}(B_1) \in I$

...

$v_{\mathcal{D}}(B_n) \in I$

Les $v_{\mathcal{D}}(B_i)$ appartiennent à un $\mathcal{D}$-modèle de P^*, donc au plus grand $\mathcal{D}$-modèle de P^* : l'ensemble des racines des $\mathcal{D}\infty$-arbres de preuve. Donc $v_{\mathcal{D}}(B_i)$ est racine d'un $\mathcal{D}\infty$-arbre de preuve A_i. Le n-uplet $< v_{\mathcal{D}}(B_1), \ldots, v_{\mathcal{D}}(B_n) >$ est racine d'une $\mathcal{D}\infty$-forêt de preuve F (extension de la notion de $\mathcal{D}$-forêt de preuve aux $\mathcal{D}\infty$-arbres de preuve). Comme pour les $\mathcal{D}$-forêts de preuve finies, il existe un squelette (éventuellement infini) S, de racine $\leftarrow b$, complet, qui correspond à F : tout nœud de F (exception faite de la racine) est un $\mathcal{D}$-atome tête d'une $\mathcal{D}$-règle R, il lui correspond donc une clause C de P et une valuation v telles que $R = v_{\mathcal{D}}(C)$; après renommage des clauses, ces valuations correspondent à une unique valuation v telle que $F = v_{\mathcal{D}}(S)$. En particulier, S est $\mathcal{D}$-soluble. D'après le théorème "branches non échecs", S correspond à une branche non échec dans tout arbre SLD équitable. Donc $\leftarrow b$ n'est pas d'échec fini.

Remarque 4.3 *La réciproque est fausse : $P^* \models_{\mathcal{D}} \neg b$ n'implique pas que $\leftarrow b$ est d'échec fini. Ceci est prouvé par le contre-exemple suivant :*

Exemple 4.2 *Soit P le programme constitué de l'unique clause*
$p(X+1) \leftarrow vrai \ \square \ p(X)$.
Son complété P^* est réduit à
$p(Y) \leftrightarrow \exists X(Y = X + 1 \wedge p(X))$.
On a $P^* \models_{\mathcal{N}} \neg p(Y)$.
Supposons en effet que $\neg p(Y)$ ne soit pas $\mathcal{N}$-conséquence de P^*. Alors il existe un $\mathcal{N}$-modèle I de P^* et une valuation v tels que $v_{\mathcal{N}}(p(Y)) \in I$. $v(Y)$ est un entier naturel y. Alors $p(y)$ est dans I. Or I est modèle de la clause $p(Y) \leftrightarrow \exists X(Y = X + 1 \wedge p(X))$ donc $\exists X(Y = X + 1 \wedge p(X))$ est vraie dans la valuation v ; donc il existe un x_1 de N tel que $y = x_1 + 1 \wedge p(x_1)$.

Le même raisonnement appliqué à x_1 implique l'existence d'un x_2 tel que $x_1 = x_2 + 1 \wedge p(x_2)$,
puis d'un x_3 tel que $x_2 = x_3 + 1 \wedge p(x_3)$, ...
On construit ainsi une suite infinie strictement décroissante d'éléments de N, ce qui est exclu. Donc $\neg p(Y)$ est bien $\mathcal{N}$-conséquence de P^*. Et pourtant $\leftarrow vrai \ \square \ p(Y)$ n'est pas d'échec fini : ce but a un seul arbre SLD comportant une seule branche (parce qu'il n'y a qu'une clause et qu'elle est d'arité 1), qui est infinie.

4.2 Réponses selon une théorie $\mathcal{T}$

Le critère de rejet étudié ici est $CR(\mathcal{T})$, donc les squelettes acceptables sont les S pour lesquels $cont(S)$ est satisfiable dans $\mathcal{T}$. Les réponses pour $\leftarrow b$ sont les squelettes S complets finis de racine $\leftarrow b$ pour lesquels la contrainte $\mathcal{C}(S)$ (conjonction des contraintes de $cont(S)$) est telle que $non(\mathcal{T} \models \neg\mathcal{C}(S))$.

Définition 4.3 *Soit $\mathcal{L}$ un langage du premier ordre.*

- *$\mathcal{L}_{clos}$ est l'ensemble des* formules closes *(sans variables libres) de $\mathcal{L}$.*
- *Si E est une partie de $\mathcal{L}_{clos}$, F une formule de $\mathcal{L}_{clos}$, F est* conséquence logique *de E, ce qui se note $E \models F$ si F est vraie dans toute interprétation de $\mathcal{L}$ qui soit modèle de E (en abrégé : dans tout modèle de E).*

- *Une* théorie *est une partie $\mathcal{T}$ de $\mathcal{L}_{clos}$ fermée par conséquence logique : $\forall F \in \mathcal{L}_{clos}$, si $\mathcal{T} \models F$ alors $F \in \mathcal{T}$.*

- *Une partie A de $\mathcal{L}_{clos}$ est un* système d'axiomes *pour la théorie $\mathcal{T}$ si $\forall F \in \mathcal{L}_{clos}$, $A \models F$ ssi $F \in \mathcal{T}$.*

- *La théorie $\mathcal{T}$ est* complète *si $\forall F \in \mathcal{L}_{clos}$, soit F est dans $\mathcal{T}$ soit sa négation $\neg F$ est dans $\mathcal{T}$.*

- *Une théorie est* consistante *si elle admet au moins un modèle ; dans le cas contraire elle est dite inconsistante ou contradictoire. Remarquons que si $\mathcal{T}$ est inconsistante alors $\mathcal{T} \models F$ pour toute F de $\mathcal{L}_{clos}$ (puisque $\mathcal{T}$ n'a aucun modèle) ; $\mathcal{T}$ étant close par conséquence logique, cela implique que $\mathcal{T}$ contient toutes les formules closes, donc une formule et sa négation.*

- *Une partie E de $\mathcal{L}$ est* satisfiable *s'il existe une interprétation $\mathcal{M}$ de $\mathcal{L}$ et une valuation v dans $\mathcal{M}$ telles que $\forall F \in E$ $v_{\mathcal{M}}(F) = vrai$.*

- *Soit A un ensemble de formules closes (une partie de $\mathcal{L}_{clos}$) et E un ensemble de formules (une partie de $\mathcal{L}$). E est* satisfiable dans A *si E est satisfiable dans un modèle de A, autrement dit si $A \bigcup E$ est satisfiable.*

Dans le cas de la programmation logique avec contraintes, le langage du premier ordre est $\mathcal{L}(V, \Sigma, \Pi_C)$. Nous nous intéresserons à une forme affaiblie de la notion de théorie complète : théorie satisfaction complète.

Définition 4.4 *(Théorie satisfaction complète)*
Une théorie $\mathcal{T}$ de $\mathcal{L}(V, \Sigma, \Pi_C)$ est satisfaction complète si pour toute contrainte c, soit $\mathcal{T} \models \tilde{\exists} c$, soit $\mathcal{T} \models \neg c$. Donc non$(\mathcal{T} \models \neg c) \Leftrightarrow \mathcal{T} \models \tilde{\exists} c$.

Remarque 4.4 *$\mathcal{T}$ est satisfaction complète si elle est complète pour une certaine catégorie de formules : celles qui sont de la forme $\tilde{\exists} c$ où c est une contrainte (il se peut que le langage des contraintes soit un sous-langage strict de $\mathcal{L}(V, \Sigma, \Pi_C)$). Dans $\mathcal{T} \models \neg c$, la formule $\neg c$ n'est a priori pas close ; il est sous-entendu que c'est la fermeture universelle de $\neg c$ qui est conséquence logique de $\mathcal{T}$, autrement dit, pour tout modèle $\mathcal{M}$ de $\mathcal{T}$ et toute valuation v dans $\mathcal{M}$, $v_{\mathcal{M}}(c) = faux$. Rappelons que $\mathcal{T} \models \tilde{\exists} c$ équivaut à : dans tout modèle $\mathcal{M}$ de $\mathcal{T}$ il existe une valuation v telle que $v_{\mathcal{M}}(c) = vrai$. Donc $\mathcal{T} \models \neg c$ est équivalent à $\mathcal{T} \models \neg\tilde{\exists} c$ ou encore $\mathcal{T} \models \tilde{\forall}\neg c$.*

On peut envisager différentes théories :

4.2.1 Théorie associée à une pré-interprétation

Une pré-interprétation $\mathcal{D}$ étant donnée, la satisfiabilité des contraintes est déterminée par $\mathcal{D}$: $\models_{\mathcal{D}} \tilde{\exists} c$. On peut représenter cela par la théorie $th(\mathcal{D})$ engendrée par les axiomes :

$\{\tilde{\exists} c \mid c$ est une contrainte telle que $\models_{\mathcal{D}} \tilde{\exists} c\} \bigcup \{\tilde{\forall}\neg c \mid c$ est telle que $\models_{\mathcal{D}} \neg c\}$.

Pour toute contrainte c, soit $\models_{\mathcal{D}} \tilde{\exists} c$ et alors $th(\mathcal{D}) \models \tilde{\exists} c$, soit $\models_{\mathcal{D}} \neg c$ et alors $th(\mathcal{D}) \models \neg c$. Donc $th(\mathcal{D})$ est satisfaction complète.

Lemme 4.2 *Si la théorie $\mathcal{T}$ est satisfaction complète alors $CR(\mathcal{T}) = CR(\mathcal{D})$ pour tout modèle $\mathcal{D}$ de $\mathcal{T}$.*

Preuve : Soit $\mathcal{D}$ un modèle de $\mathcal{T}$. $\mathcal{T}$ étant satisfaction complète, $non(\mathcal{T} \models \neg c) \Rightarrow \mathcal{T} \models \tilde{\exists} c \Rightarrow \models_{\mathcal{D}} \tilde{\exists} c$.

Inversement, si $\models_{\mathcal{D}} \tilde{\exists} c$, $\mathcal{T}$ étant satisfaction complète on a soit $\mathcal{T} \models \tilde{\exists} c$ soit $\mathcal{T} \models \neg c$; mais $\mathcal{T} \models \neg c$ impliquerait $\models_{\mathcal{D}} \neg c$.

Remarque 4.5 *$\models_{\mathcal{D}}$ et $th(\mathcal{D}) \models$ ont le même effet sur les contraintes mais pas nécessairement*

- sur des formules qui ne sont pas des contraintes,

- sur les implications infinies $c \rightarrow \bigvee_{r \in R} r$ qui ne sont pas de véritables formules,

- et sur les conséquences logiques du programme, ce qui est prouvé par l'exemple ci-dessous.

Exemple 4.3 *Considérons de nouveau le programme Ent de l'exemple 4.1. Nous avons vu que $Ent \models_{\mathcal{N}} ent(X)$ et que $\models_{\mathcal{N}} vrai \rightarrow \bigvee_{n \in N} r_n$.*

Il n'est pas vrai que $Ent, th(\mathcal{N}) \models ent(X)$ parce que $th(\mathcal{N})$ a d'autres modèles que $\mathcal{N}$: il existe des modèles non standards non isomorphes à $\mathcal{N}$ dans lesquels les éléments ne sont pas tous de la forme $suc^n(0)$.

4.2.2 Théorie associée à un algorithme

Dans la pratique, même si l'interprète sous-entend une pré-interprétation $\mathcal{D}$, le résolveur de contraintes ne peut pas examiner toutes les valuations possibles pour décider si une contrainte est satisfiable ou non. Il emploie un autre moyen : un algorithme qui répond toujours en un temps fini.

Un algorithme de satisfiabilité A pour les contraintes étant donné, on note $\models_A \tilde{\exists} c$ le fait que A répond OUI à la question de savoir si c est satisfiable, $\models_A \neg c$ le fait que A répond NON. Il n'est pas exclu que A soit incomplet : que pour certaines contraintes il ne réponde ni OUI ni NON, mais seulement $PEUT\text{-}ETRE$; alors ni $\models_A \tilde{\exists} c$ ni $\models_A \neg c$. C'est le cas de l'algorithme intégré dans l'interprète $CLP(\mathcal{R})$ décrit dans [11].

La théorie associée à A notée $th(A)$ est engendrée par les axiomes

$$\{\tilde{\exists} c \mid \models_A \tilde{\exists} c\} \bigcup \{\tilde{\forall} \neg c \mid \models_A \neg c\}$$

$Th(A)$ est consistante si l'ensemble d'axiomes est lui-même consistant. Si A est complet alors $th(A)$ est satisfaction complète.

$th(A)$ est une approximation de A :

- si c est satisfiable selon A alors $th(A) \models \tilde{\exists} c$,
- si c est insatisfiable selon A alors $th(A) \models \neg c$,
- mais il peut exister des c telles que $th(A) \models \tilde{\exists} c$ ou $th(A) \models \neg c$ alors que A répond $PEUT\text{-}ETRE$ pour c.

Nous supposons que le critère de rejet est $CR(\mathcal{T})$ pour une certaine théorie $\mathcal{T}$ consistante mais pas nécessairement satisfaction complète. Les liens entre les réponses selon $CR(\mathcal{T})$ et les conséquences logiques du programme sont exprimés par les quatre théorèmes suivants.

Théorème 4.6 *(Existence d'une réponse)*

1. *si* $\leftarrow b$ *admet une réponse alors* $non(P, \mathcal{T} \models \neg b)$,
2. *si* $P, \mathcal{T} \models \tilde{\exists} b$ *alors* $\leftarrow b$ *admet une réponse.*

Preuve :

1. Si $\leftarrow b$ admet une réponse selon $CR(\mathcal{T})$, c'est un squelette fini complet S de racine $\leftarrow b$ dont la contrainte $\mathcal{C}(S)$ est telle que $non(\mathcal{T} \models \neg\mathcal{C}(S))$. Alors il existe un modèle $\mathcal{M}$ de $\mathcal{T}$ et une valuation v tels que $v_{\mathcal{M}}(\mathcal{C}(S)) = vrai$. S est donc une réponse pour $\leftarrow b$ selon le critère $CR(\mathcal{M})$. Le théorème sur l'existence d'une solution dans une pré-interprétation permet d'en déduire $P \models_{\mathcal{M}} \tilde{\exists} b$, et donc $non(P, \mathcal{T} \models \neg b)$.
2. Si $P, \mathcal{T} \models \tilde{\exists} b$ alors $P \models_{\mathcal{M}} \tilde{\exists} b$ pour tout modèle $\mathcal{M}$ de $\mathcal{T}$. Soit $\mathcal{D}$ un modèle de $\mathcal{T}$. On a $P \models_{\mathcal{D}} \tilde{\exists} b$, donc $\leftarrow b$ admet une réponse S selon $CR(\mathcal{D})$. La contrainte correspondante r est satisfiable dans le modèle $\mathcal{D}$ de $\mathcal{T}$, donc $non(\mathcal{T} \models \neg r)$; donc S est une réponse selon $CR(\mathcal{T})$.

Remarque 4.6 *L'hypothèse* $non(P, \mathcal{T} \models \neg b)$ *ne suffit pas pour assurer l'existence d'une réponse. En effet,* $non(P, \mathcal{T} \models \neg b)$ *signifie que* b *est vrai dans au moins un modèle* $\mathcal{M}$ *de* $\mathcal{T}$*, au moins un* $\mathcal{M}$*-modèle* I *de* P *et au moins une valuation* v*. On ne peut rien en conclure car* I *peut être le plus grand* $\mathcal{M}$*-modèle de* P *: l'ensemble de tous les* $\mathcal{M}$*-atomes.*

Théorème 4.7 *(Existence d'une réponse pour une théorie satisfaction complète)*
Si $\mathcal{T}$ *est satisfaction complète,* $\leftarrow b$ *admet une réponse ssi* $P, \mathcal{T} \models \tilde{\exists} b$.

Preuve :

- Nous savons que $P, \mathcal{T} \models \tilde{\exists} b$ implique l'existence d'une réponse, même lorsque $\mathcal{T}$ n'est pas satisfaction complète.
- Inversement, si $\leftarrow b$ admet une réponse selon $\mathcal{T}$ c'est un squelette S fini complet de racine $\leftarrow b$ dont la contrainte r est telle que $non(\mathcal{T} \models \neg r)$ ou encore $\mathcal{T} \models \tilde{\exists} r$. Dans tout modèle $\mathcal{D}$ de $\mathcal{T}$ on a $\models_{\mathcal{D}} \tilde{\exists} r$, donc $P \models_{\mathcal{D}} \tilde{\exists} b$. Puisque c'est vrai pour tout modèle $\mathcal{D}$ de $\mathcal{T}$, on en déduit $P, \mathcal{T} \models \tilde{\exists} b$.

Théorème 4.8 *(Contraintes réponses et conséquences logiques)*
Si r *est une contrainte réponse pour le but* $\leftarrow b$ *alors* $P, \mathcal{T} \models r \rightarrow b$.

Preuve : Nous savons que $P \models r \rightarrow b$ grâce au théorème sur les réponses générales. $P, \mathcal{T} \models r \rightarrow b$ n'en est qu'un cas particulier.

Théorème 4.9 *(Couverture des conséquences logiques par des contraintes réponses)*
Pour toute contrainte c, si $P, \mathcal{T} \models c \rightarrow b$ alors

1. *propriété de* couverture *: $\mathcal{T} \models c \rightarrow \bigvee_{r \in R} r$ où R est l'ensemble des contraintes réponses pour $\leftarrow b$, ce qui signifie : pour tout modèle $\mathcal{M}$ de $\mathcal{T}$ et tout valuation v dans $\mathcal{M}$, si v est solution de c alors $\exists r \in R$ telle que $v_{\mathcal{M}}(r) = vrai$.*

2. *propriété de* couverture finie *: il existe un ensemble fini R' de contraintes réponses tel que $\mathcal{T} \models c \rightarrow \bigvee_{r' \in R'} r'$.*

Preuve de 1 : Si $P, \mathcal{T} \models c \rightarrow b$ alors pour tout modèle $\mathcal{M}$ de $\mathcal{T}$, $P \models_{\mathcal{M}} c \rightarrow b$. Donc $\models_{\mathcal{M}} c \rightarrow \bigvee_{r \in R_{\mathcal{M}}} r$ où $R_{\mathcal{M}}$ est l'ensemble des contraintes réponses selon le critère $CR(\mathcal{M})$. Si r est dans $R_{\mathcal{M}}$ alors $\models_{\mathcal{M}} \tilde{\exists} r$, donc $non(\mathcal{T} \models \neg r)$, donc r est une contrainte réponse selon $CR(\mathcal{T})$, donc r est dans R. On en déduit $\models_{\mathcal{M}} c \rightarrow \bigvee_{r \in R} r$ pour tout modèle $\mathcal{M}$ de $\mathcal{T}$, et donc $\mathcal{T} \models c \rightarrow \bigvee_{r \in R} r$.

La preuve de 2, qui utilise la compacité de la logique du premier ordre, demande quelques préliminaires.

Théorème 4.10 *(Une des formes du théorème de compacité de la logique du premier ordre)*
Un ensemble de formules E (partie d'un langage du premier ordre $\mathcal{L}(V, \Sigma, \Pi)$) est satisfiable ssi toute partie finie de E est satisfiable.

Preuve :

- Si E est satisfiable, alors toutes ses parties finies le sont aussi.

- Réciproquement, supposons que toutes les parties finies de E soient satisfiables, et que E soit infini et insatisfiable. Si E ne comprend que des formules closes on peut immédiatement déduire une contradiction en utilisant la forme standard du théorème de compacité : tout ensemble de formules inconsistant a une partie finie inconsistante. Si ce n'est pas le cas, soit X l'ensemble des variables libres dans les formules de E. Soit A un ensemble disjoint de V, Σ et Π qui soit en bijection avec X. Considérons le langage $\mathcal{L}' = \mathcal{L}(V, \Sigma \bigcup A, \Pi)$ où les éléments de A sont des constantes, c'est-à-dire des fonctions d'arité 0. Toute interprétation I de $\mathcal{L}$ peut être prolongée en une interprétation I' de $\mathcal{L}'$ ainsi définie : elle a le même domaine D, à tout symbole de fonction f de Σ on associe la même fonction f_I, à tout symbole de prédicat p de Π on associe la même relation p_I ; il ne reste qu'à donner des valeurs dans D pour les constantes de A. Si l'on dispose d'une valuation v sur X (dans l'interprétation I), on peut attribuer les valeurs correspondantes aux constantes de A pour fabriquer I'. Réciproquement, si l'interprétation I' de $\mathcal{L}'$ est donnée, on peut lui associer une interprétation I de $\mathcal{L}$ et une valuation v sur X.

  L'ensemble de formules E de $\mathcal{L}$ a pour correspondant dans $\mathcal{L}'$ l'ensemble E' obtenu en remplaçant dans les formules de E les variables de X par les constantes de A.

  E' est inconsistant car s'il était consistant il existerait une interprétation I' de $\mathcal{L}'$ modèle de E' ; on en déduirait une interprétation I de $\mathcal{L}$ et une valuation v sur X pour lesquelles les formules de E seraient vraies (E serait satisfiable).

En vertu du théorème de compacité (forme standard), E' a une partie finie inconsistante G'. Les formules correspondantes dans $\mathcal{L}$ forment une partie finie insatisfiable G de E, ce qui contredit l'hypothèse.

Corollaire 4.1 *(Satisfiabilité finie)*
Soit $\mathcal{A}$ un ensemble de formules closes et E un ensemble de formules. Alors E est satisfiable dans $\mathcal{A}$ ssi toutes les parties finies de E sont satisfiables dans $\mathcal{A}$.

Preuve :

- Si E est satisfiable dans $\mathcal{A}$ toutes ses parties le sont aussi, en particulier les parties finies.
- Inversement, montrons que si E est insatisfiable dans $\mathcal{A}$ alors E a une partie finie insatisfiable dans $\mathcal{A}$. Si E est insatisfiable dans $\mathcal{A}$, c'est que $\mathcal{A} \bigcup E$ est insatisfiable. Donc $\mathcal{A} \bigcup E$ a une partie finie insatisfiable. Cette partie finie est de la forme $\mathcal{A}' \bigcup E'$ où $\mathcal{A}'$ et E' sont des parties finies de $\mathcal{A}$ et E respectivement. Alors $\mathcal{A} \bigcup E'$ est également insatisfiable, ce qui signifie que E' est insatisfiable dans $\mathcal{A}$.

Corollaire 4.2 *(Couverture finie)*
Soit $\mathcal{A}$ un ensemble de formules closes, et F, $\{F_i\}_{i\in I}$ des formules. Alors $\mathcal{A} \models F \rightarrow \bigvee_{i\in I} F_i$ ssi il existe une partie finie J de I telle que $\mathcal{A} \models F \rightarrow \bigvee_{j\in J} F_j$.

Preuve :

- $\mathcal{A} \models F \rightarrow \bigvee_{j\in J} F_j$ où J est une partie de I implique évidemment $\mathcal{A} \models F \rightarrow \bigvee_{i\in I} F_i$.
- Réciproquement, si $\mathcal{A} \models F \rightarrow \bigvee_{i\in I} F_i$, considérons l'ensemble de formules $\mathcal{A} \bigcup \{F\} \bigcup \{\neg F_i\}_{i\in I}$. Il est insatisfiable car pour toute interprétation M modèle de $\mathcal{A}$ et toute valuation v solution de r, il existe un i tel que $v_{\mathcal{M}}(F_i) = vrai$, donc $v_{\mathcal{M}}(\neg F_i) = faux$. Donc il a une partie finie insatisfiable (si toutes les parties finies étaient satisfiables, l'ensemble entier le serait aussi). Cette partie finie insatisfiable comprend : une partie finie de $\mathcal{A}$, éventuellement F, et une partie finie $\{\neg F_j\}_{j\in J}$ de $\{\neg F_i\}_{i\in I}$. Elle est incluse dans $\mathcal{A} \bigcup \{F\} \bigcup \{\neg F_j\}_{j\in J}$ qui est donc également insatisfiable. Alors dans toute interprétation M modèle de $\mathcal{A}$ et toute valuation v telle que $v_M(F) = vrai$, il existe un j tel que $v_M(\neg F_j) = faux$, donc $v_M(F_j) = vrai$.

Preuve de 2 (théorème 4.9) : se déduit immédiatement de 1 et du corollaire de couverture finie.

La propriété d'indépendance des contraintes négatives définie ci-dessous va permettre d'affiner le dernier théorème en ajoutant la propriété de couverture unique.

Définition 4.5 *Les contraintes ont la propriété d'*indépendance des contraintes négatives *(ICN) si*
$\mathcal{T} \models c \rightarrow \bigvee_{i=1}^{n} c_i \Rightarrow \ \exists j (\mathcal{T} \models c \rightarrow c_j)$.

Remarque 4.7 *La propriété ICN est traditionnellement exprimée de manière équivalente par*

$\mathcal{T} \models \neg\tilde{\exists}(c \wedge \bigwedge_{i=1}^{n} \neg c_i) \Rightarrow \exists j (\mathcal{T} \models \neg\tilde{\exists}(c \wedge \neg c_j))$

Cette propriété ICN n'a pas de raison d'être vérifiée en général. Nous verrons que, dans le cas de la programmation logique, elle est vérifiée pour les contraintes qui représentent les substitutions, ce qui permet d'expliquer le résultat classique : si la substitution θ produit une conséquence logique du programme, alors il existe une substitution réponse plus générale que θ.

Théorème 4.11 *(Couverture unique)*

Si les contraintes ont la propriété d'indépendance des contraintes négatives et si $P, \mathcal{T} \models c \rightarrow b$ alors il existe une contrainte réponse r telle que $\mathcal{T} \models c \rightarrow r$.

Preuve : Résulte immédiatement du corollaire de couverture finie et de la propriété d'indépendance des contraintes négatives.

4.2.3 Cas particulier de la programmation logique

En programmation logique, on sait que si θ est une substitution telle que $P \models b\theta$ alors il existe une substitution réponse σ plus générale que θ. Cela peut s'expliquer par le fait que les contraintes qui représentent les substitutions ont la propriété ICN.

Tout programme logique peut être représenté par un programme contraint en ajoutant à tout corps de règle ou de but la contrainte initiale $vrai$. L'ensemble des symboles de prédicats de contraintes Π_C est réduit à $\emptyset$, ce qui n'empêche pas de disposer des contraintes $vrai$ et $faux$, et de l'égalité. La théorie représentant la satisfiabilité des contraintes est vide puisque l'on manipule un langage du premier ordre avec égalité.

On ne considère que des substitutions idempotentes : du type $\tilde{x} = \tilde{t}$ où $\tilde{x}$ est l'ensemble des variables libres du but question, $\tilde{t}$ un ensemble de termes dans lesquels les $\tilde{x}$ n'ont pas d'occurrences libres. Notons $\tilde{y}$ les variables libres des $\tilde{t}$. La substitution étant idempotente $\tilde{x}$ et $\tilde{y}$ sont disjoints.

La substitution $\theta : \tilde{x} = \tilde{t}$ peut être représentée par la contrainte $\tilde{x} = \tilde{t}$ $(x_1 = t_1 \wedge x_2 = t_2 \wedge \ldots \wedge x_n = t_n)$ que nous noterons c_θ. Alors

$$P \models b\theta \text{ ssi } P \models c_\theta \rightarrow b$$

parce que

$$\models b\theta \leftrightarrow (c_\theta \rightarrow b)$$

$\rightarrow$ Soit $\mathcal{D}$ une pré-interprétation, I une interprétation dans $\mathcal{D}$, v une valuation solution de $b\theta$ et de c_θ. Alors v est telle que $v(\tilde{x}) = v_{\mathcal{D}}(\tilde{t})$. D'autre part, $v_{\mathcal{D}}(b\theta) = v'_{\mathcal{D}}(b)$ où v' est la valuation qui attribue aux variables $\tilde{x}$ les valeurs des $\tilde{t}$ dans v soit $v'(\tilde{x}) = v_{\mathcal{D}}(\tilde{t})$. Alors $v_{\mathcal{D}}(b) = v'_{\mathcal{D}}(b)$ (parce que v et v' coïncident sur $\tilde{x}$) $= v_{\mathcal{D}}(b\theta)$ qui est dans I. Donc $v_{\mathcal{D}}(b)$ est dans I.

$\leftarrow$ Supposons que $c_\theta \rightarrow b$ est vrai. Soit v une valuation. Soit v' la valuation identique à v sauf sur $\tilde{x}$, et telle que $v'(\tilde{x}) = v_{\mathcal{D}}(\tilde{t})$. Alors $v_{\mathcal{D}}(b\theta) = v'_{\mathcal{D}}(b\theta) = v'_{\mathcal{D}}(b)$, et $v'_{\mathcal{D}}(b)$ est dans I car $v'_{\mathcal{D}}(c_\theta) = vrai$. Donc $v_{\mathcal{D}}(b\theta)$ est dans I.

La réponse $\sigma : \tilde{x} = \tilde{t}$ à la question $\leftarrow b$ pour le programme P doit être comprise comme $\exists \tilde{y}(\tilde{x} = \tilde{t})$.

Si $P \models b\sigma$ et si θ est une instance de σ de la forme $\tilde{x} = \tilde{u}$ (où les $\tilde{u}$ comprennent les variables libres $\tilde{z}$), ce qui signifie que les $\tilde{u}$ sont obtenus par substitution à partir des $\tilde{t}$, alors $P \models b\theta$ parce que

$$\models \exists_{-\tilde{x}} c_\theta \rightarrow \exists_{-\tilde{x}} c_\sigma$$

ou encore

$$\models \exists \tilde{z}(\tilde{x} = \tilde{u}) \rightarrow \exists \tilde{y}(\tilde{x} = \tilde{t})$$

Les $\tilde{u}$ sont obtenus par substitution à partir des $\tilde{t}$: un remplacement des $\tilde{y}$ par des termes $\tilde{w}$ dans lesquels les variables libres sont les $\tilde{z}$. Soit v une valuation sur $\tilde{x}$ telle que $\exists \tilde{z}(\tilde{x} = \tilde{u})$; alors il existe une valuation v' prolongeant v à $\tilde{x} \bigcup \tilde{z}$ telle que $v(\tilde{x}) = v'(\tilde{x}) = v'_{\mathcal{D}}(\tilde{u})$. Les $\tilde{u}$ étant obtenus par substitution à partir des $\tilde{t}$, il existe une valuation v'' telle que $v'_{\mathcal{D}}(\tilde{u}) = v''_{\mathcal{D}}(\tilde{t})$ et $v''(\tilde{y}) = v'_{\mathcal{D}}(\tilde{w})$. Alors $v(\tilde{x}) = v'(\tilde{x}) = v''(\tilde{x}) = v''_{\mathcal{D}}(\tilde{t})$, donc $\exists \tilde{y}(\tilde{x} = \tilde{t})$ est prouvée par l'existence de la valuation v''.

Le rapport entre les substitutions σ et θ que nous venons d'étudier se traduit par : σ est plus générale que θ ssi $\models \exists_{-\tilde{x}} c_\theta \rightarrow \exists_{-\tilde{x}} c_\sigma$.

Revenons au problème initial : si $P \models b\theta$ alors il existe une substitution réponse σ plus générale que θ. Ceci s'exprime avec contraintes par :

$$P \models c \rightarrow b \Rightarrow \exists \text{ une réponse } r \text{ telle que } \models c \rightarrow r$$

Nous savons déjà que

$$P \models c \rightarrow b \Rightarrow \exists \text{ un nombre fini de réponses } \{r_i\}_{1 \leq i \leq n} \text{ telles que } \models c \rightarrow \bigvee_{i=1}^{n} r_i$$

Le passage de n à 1 sera obtenu grâce à la propriété d'indépendance des contraintes négatives pour les contraintes représentant les substitutions applicables à b : substitutions idempotentes, les variables libres $\tilde{x}$ sont les mêmes. Supposons donc

$$\models \exists_{-\tilde{x}}\ \tilde{x} = \tilde{u} \rightarrow \bigvee_{i=1}^{n} \exists_{-\tilde{x}}\ \tilde{x} = \tilde{t}_i$$

qui s'écrit encore :

$$\models \exists \tilde{z}\ \tilde{x} = \tilde{u} \rightarrow \bigvee_{i=1}^{n} \exists \tilde{y}_i\ \tilde{x} = \tilde{t}_i \quad (1)$$

(1) est vraie dans toute interprétation, en particulier dans l'univers de Herbrand avec variables :

- le domaine D est l'ensemble des termes avec variables construits sur V et Σ,
- l'interprétation d'un symbole de fonction f d'arité n est l'application de D^n dans D qui associe aux termes t_1, ..., t_n le terme $f(t_1, \ldots, t_n)$.

Considérons la valuation v qui associe aux variables $\tilde{x}$ les valeurs $\tilde{u}$ (qui sont des éléments du domaine D). Alors la valuation v' qui prolonge v à $\tilde{x} \bigcup \tilde{z}$ par l'identité sur $\tilde{z}$ rend vraie $\exists \tilde{z}\ \tilde{x} = \tilde{u}$. Donc la partie droite de (1) : $\bigvee_{i=1}^{n} \exists \tilde{y}_i\ \tilde{x} = \tilde{t}_i$ est vraie dans v.

Donc il existe un j tel que $\exists \tilde{y_j}\ \tilde{x} = \tilde{t_j}$ est vraie dans v. Donc il existe une valuation v'' qui prolonge v à $\tilde{x} \bigcup \tilde{y_j}$ telle que $v(\tilde{x}) = v''(\tilde{t_j})$. D'autre part, $v(\tilde{x}) = \tilde{u}$. Donc $\tilde{u} = v''(\tilde{t_j})$, ce qui exprime que $\tilde{u}$ est une instanciation de $\tilde{t_j}$; la substitution $\tilde{x} = \tilde{u}$ est donc moins générale que la substitution réponse $\tilde{x} = \tilde{t_j}$.

4.2.4 Branches non échec

Les branches non échec ont une caractérisation plus simple que dans le cas général :

Théorème 4.12 *(Branches non échec)*
Les branches non échec des arbres SLD équitables pour B selon $CR(\mathcal{T})$ correspondent aux squelettes complets acceptables (éventuellement infinis) de racine B.

Preuve : Nous savons que les branches non échec correspondent aux squelettes complets S de racine B finiment acceptables. Pour toute partie finie P de $cont(S)$ il existe un sous-squelette fini S' de S tel que P soit incluse dans $cont(S')$; P est donc satisfiable dans $\mathcal{T}$. Le théorème de satisfiabilité finie assure que $cont(S)$ est satisfiable dans $\mathcal{T}$, ce qui veut dire que S est acceptable.

Les conséquences positives de P et du complété P^* sont les mêmes :

Théorème 4.13

- $P, \mathcal{T} \models \tilde{\exists} b$ *ssi* $P^*, \mathcal{T} \models \tilde{\exists} b$,
- $P, \mathcal{T} \models c \rightarrow b$ *ssi* $P^*, \mathcal{T} \models c \rightarrow b$.

Preuve :

- $P, \mathcal{T} \models \tilde{\exists} b \Rightarrow P^*, \mathcal{T} \models \tilde{\exists} b$ car tout modèle de $\mathcal{T}$ et P^* est aussi modèle de $\mathcal{T}$ et P.
- Inversement, si $P^*, \mathcal{T} \models \tilde{\exists} b$, dans tout modèle $\mathcal{D}$ de $\mathcal{T}$ on a $P^* \models_{\mathcal{D}} \tilde{\exists} b$, donc $P \models_{\mathcal{D}} \tilde{\exists} b$; et ceci pour tout modèle $\mathcal{D}$ de $\mathcal{T}$; donc $P, \mathcal{T} \models \tilde{\exists} b$.
- Si $P, \mathcal{T} \models c \rightarrow b$ alors $P^*, \mathcal{T} \models c \rightarrow b$ car tout modèle de P^* est un modèle de P.
- Inversement si $P^*, \mathcal{T} \models c \rightarrow b$, pour tout modèle $\mathcal{D}$ de $\mathcal{T}$, toute valuation v solution de c et tout $\mathcal{D}$-modèle I de P^*, $v_{\mathcal{D}}(b) \in I$. Donc $v_{\mathcal{D}}(b)$ est dans tous les $\mathcal{D}$-modèles de P^*, en particulier le plus petit qui est en même temps le plus petit $\mathcal{D}$-modèle de P ; donc $v_{\mathcal{D}}(b)$ est dans tous les $\mathcal{D}$-modèles de P. Donc $P, \mathcal{T} \models c \rightarrow b$.

Les conséquences négatives de P^* correspondent exactement aux buts d'échec fini :

Théorème 4.14 *Un but $\leftarrow b$ est d'échec fini selon $CR(\mathcal{T})$ ssi $P^*, \mathcal{T} \models \neg b$.*

Preuve : Soit $b = c \square B_1, \ldots, B_n$. Pour montrer que

$$\leftarrow b \text{ d'échec fini } \Leftrightarrow P^*, \mathcal{T} \models \neg b$$

nous allons montrer que les propriétés opposées sont équivalentes :

$$non\ (P^*, \mathcal{T} \models \neg b) \Leftrightarrow \leftarrow b \text{ n'est pas d'échec fini}$$

.

$\Rightarrow$ Si *non* $(P^*, \mathcal{T} \models \neg b)$, il existe une pré-interprétation $\mathcal{D}$ modèle de $\mathcal{T}$, une $\mathcal{D}$-interprétation I modèle de P^* (c'est-à-dire point fixe de $T^P_{\mathcal{D}}$) et une valuation v telles que $v_I(b) = vrai$. Cela signifie :

$v_{\mathcal{D}}(c) = vrai$

$v_{\mathcal{D}}(B_1) \in I$

...

$v_{\mathcal{D}}(B_n) \in I$

Les $v_{\mathcal{D}}(B_i)$ appartiennent à un $\mathcal{D}$-modèle de P^*, donc au plus grand $\mathcal{D}$-modèle de P^* : l'ensemble des racines des $\mathcal{D}\infty$-arbres de preuve. Donc $v_{\mathcal{D}}(B_i)$ est racine d'un $\mathcal{D}\infty$-arbre de preuve A_i. Le n-uplet $< v_{\mathcal{D}}(B_1), \ldots, v_{\mathcal{D}}(B_n) >$ est racine d'une $\mathcal{D}\infty$-forêt de preuve F. Comme pour les $\mathcal{D}$-forêts de preuve finies, il existe un squelette (éventuellement infini) S, de racine $\leftarrow b$, complet, qui correspond à F : tout nœud de F (exception faite de la racine) est un $\mathcal{D}$-atome tête d'une $\mathcal{D}$-règle R, il lui correspond donc une clause C de P et une valuation v telles que $R = v_{\mathcal{D}}(C)$; après renommage des clauses, ces valuations correspondent à une unique valuation v telle que $F = v_{\mathcal{D}}(S)$. En particulier, S est $\mathcal{D}$-soluble. Donc S est acceptable selon $CR(\mathcal{T})$. D'après le théorème "branches non échec", S correspond à une branche non échec dans tout arbre SLD équitable. Donc $\leftarrow b$ n'est pas d'échec fini.

$\Leftarrow$ Si $\leftarrow b$ n'est pas d'échec fini, il a un arbre équitable qui a une branche non échec. D'après le théorème "branches non échec", cette branche correspond à un squelette S (éventuellement infini) complet acceptable selon $CR(\mathcal{T})$. Donc il existe un modèle $\mathcal{D}$ de $\mathcal{T}$ et une valuation v dans $\mathcal{D}$ solution de $cont(S)$. Alors $v_{\mathcal{D}}(S)$ est une $\mathcal{D}\infty$-forêt de preuve de racine $< B_1, \ldots, B_n >$, ce qui montre que chaque $v_{\mathcal{D}}(B_i)$ est dans le plus grand $\mathcal{D}$-modèle de P^*. Donc b est vrai dans le modèle $\mathcal{D}$ de $\mathcal{T}$, le plus grand $\mathcal{D}$-modèle de P^* et la valuation v. Donc *non* $(P^*, \mathcal{T} \models \neg b)$.

4.3 Réponses selon un critère CR en relation avec une théorie $\mathcal{T}$

Nous supposons ici que le critère de rejet CR ne coïncide pas nécessairement avec $CR(\mathcal{T})$ pour une certaine théorie $\mathcal{T}$ mais est en relation avec celui-ci : les réponses selon CR englobent les réponses selon $CR(\mathcal{T})$.

C'est la situation dans laquelle se trouvent l'algorithme "pratique" A utilisé comme critère de rejet (les réponses OUI et $PEUT\text{-}ETRE$ sont considérées comme une acceptation), et la théorie $th(A)$.

Définition 4.6 *(Relation entre CR et $\mathcal{T}$)*
CR est en relation avec *$\mathcal{T}$ si pour toute collection de contraintes E, E rejetée par $CR \Rightarrow E$ rejetée par $CR(\mathcal{T})$.*

Il est clair qu'alors toute réponse selon $CR(\mathcal{T})$ est une réponse selon CR.

Théorème 4.15 *Si CR est en relation avec $\mathcal{T}$,*
$P, \mathcal{T} \models \tilde{\exists} b \Rightarrow \leftarrow b$ admet une réponse (selon CR).

Preuve : Si $P, \mathcal{T} \models \tilde{\exists} b$ alors $\leftarrow b$ admet une réponse selon $CR(\mathcal{T})$, qui est également une réponse selon CR.

Théorème 4.16 *Si r est une contrainte réponse pour le but $\leftarrow b$ alors $P, \mathcal{T} \models r \rightarrow b$.*

Preuve : Ce n'est qu'un cas particulier de $P \models r \rightarrow b$.

Théorème 4.17 *Si $P, \mathcal{T} \models c \rightarrow b$ alors*

1. *$\mathcal{T} \models c \rightarrow \bigvee_{r \in R} r$ où R est l'ensemble des contraintes réponses (selon CR) pour $\leftarrow b$,*
2. *il existe un ensemble fini R' de contraintes réponses (selon CR) tel que $\mathcal{T} \models c \rightarrow \bigvee_{r' \in R'} r'$,*
3. *et si les contraintes ont la propriété ICN, il existe une contrainte réponse r telle que $\mathcal{T} \models c \rightarrow r$.*

Preuve :

1. On sait que $\mathcal{T} \models c \rightarrow \bigvee_{r \in R_{\mathcal{T}}} r$ où $R_{\mathcal{T}}$ est l'ensemble des contraintes réponses pour $\leftarrow b$ selon $CR(\mathcal{T})$. Et les réponses selon $CR(\mathcal{T})$ font partie des réponses selon CR.
2. La couverture finie est obtenue avec des contraintes réponses selon $CR(\mathcal{T})$, qui sont des contraintes réponses selon CR.
3. Dans le cas ICN, la couverture unique est assurée par une contrainte réponse selon $CR(\mathcal{T})$ qui est aussi une contrainte réponse selon CR.

Théorème 4.18 *Si $\leftarrow b$ est d'échec fini alors $P^*, \mathcal{T} \models \neg b$.*

Preuve : Si $\leftarrow b$ est d'échec fini selon CR, il est a fortiori d'échec fini selon $CR(\mathcal{T})$, donc $P^*, \mathcal{T} \models \neg b$.

5 Atomes contraints

Les questions atomiques jouent un rôle primordial parce que :

- on peut toujours se ramener à une question atomique en construisant une nouvelle clause qui a pour corps le but non atomique initial,

- les atomes apparaissent explicitement dans les dérivations à partir d'un but donné : à tout nœud N autre que la racine dans un squelette S on associe l'atome noté $atome(N)$.

Nous considérons ici des *atomes contraints* qui sont des couples atome-contrainte. Les $\mathcal{D}$-atomes, qui servent à étiqueter les arbres de preuve, comprennent des éléments du domaine de la pré-interprétation qui ne sont donc pas dans le langage du programme. Les atomes contraints permettent de s'affranchir de toute pré-interprétation et de rester dans le langage du programme. Ils permettent également de caractériser un programme comme un ensemble de faits : l'ensemble des atomes contraints déduits de P est un programme équivalent à P en ce sens qu'il donne les mêmes contraintes réponses.

Les atomes contraints sont envisagés dans [12, 13, 10].

Définition 5.1 *(Atome contraint)*
Un atome contraint est un couple $< p(\tilde{x}), c >$ *où*

- $p(\tilde{x})$ *est un atome portant sur des variables :* $p \in \Pi_P$, $\tilde{x}$ *est un ensemble de variables distinctes,*

- c *est une contrainte ne comprenant pas d'autres variables libres que* $\tilde{x}$.

Nous le noterons $p(\tilde{x}) \leftarrow c$, *comme un fait.*

Remarque 5.1 *La restriction à des atomes ne comprenant que des variables (distinctes) n'est pas très contraignante car tout atome* $p(\tilde{t})$ *comprenant des termes peut se représenter par* $p(\tilde{x}) \leftarrow \tilde{x} = \tilde{t}$.

5.1 Instances, ensemble succès

Définition 5.2 *($\mathcal{D}$-instances)*
Les $\mathcal{D}$-instances d'un atome contraint dans une pré-interprétation $\mathcal{D}$ constituent l'ensemble de $\mathcal{D}$-atomes :

$$[p(\tilde{x}) \leftarrow c]_{\mathcal{D}} = \{p(v(\tilde{x})) \mid v \text{ valuation de } \tilde{x} \text{ dans } \mathcal{D} \text{ solution de } c\}$$

Cette définition se généralise à un ensemble d'atomes contraints X :

$$[X]_{\mathcal{D}} = \bigcup_{A \in X} [A]_{\mathcal{D}}$$

Remarque 5.2 *Si c est insatisfiable dans $\mathcal{D}$ alors* $[p(\tilde{x}) \leftarrow c]_{\mathcal{D}} = \emptyset$.

Définition 5.3 *(Ensemble succès)*
L'ensemble succès associé à un programme P et un critère de rejet CR est

$$SS_{CR}(P) = \{p(\tilde{x}) \leftarrow r \mid \models r \leftrightarrow \text{une contrainte réponse selon } CR \text{ pour} \leftarrow vrai \square p(\tilde{x})\}$$

L'ensemble succès et ses intances dans un domaine $\mathcal{D}$ sont définis dans [12, 13] et [10].
Notre définition d'ensemble succès est analogue à celle de [12, 13] et à SS_3 de [10].

Définition 5.4 *(Relation entre CR et $\mathcal{D}$)*
Le critère de rejet CR est en relation avec $\mathcal{D}$ *si toute collection de contraintes rejetée par CR est insatisfiable dans $\mathcal{D}$.*

Si le critère de rejet CR est en relation avec la pré-interprétation $\mathcal{D}$, alors les $\mathcal{D}$-instances de l'ensemble succès coïncident avec le plus petit $\mathcal{D}$-modèle de P.

Théorème 5.1 *Si CR est en relation avec $\mathcal{D}$ alors $[SS_{CR}(P)]_{\mathcal{D}} = M^P_{\mathcal{D}}$.*

Preuve :

1. Tout élément de $[SS_{CR}(P)]_{\mathcal{D}}$ est de la forme $p(v(\tilde{x}))$ où $p(\tilde{x}) \leftarrow r$ est dans $SS_{CR}(P)$ et v est une solution de r. Donc r est une contrainte réponse pour $p(\tilde{x})$; il existe un squelette fini S complet de racine $\leftarrow vrai \square p(\tilde{x})$ dont la contrainte associée est r. v est une solution de r, donc $v_{\mathcal{D}}(S)$ est un $\mathcal{D}$-arbre de preuve (une forêt de preuve réduite à un seul arbre) de racine $v_{\mathcal{D}}(vrai \square p(\tilde{x})) = p(v(\tilde{x}))$. Ce $\mathcal{D}$-atome $p(v(\tilde{x}))$ est donc dans $Rac^P_{\mathcal{D}} = M^P_{\mathcal{D}}$.

2. Puisque CR est en relation avec $\mathcal{D}$, $SS_{CR}(P)$ contient $SS_{CR(\mathcal{D})}(P)$, donc $[SS_{CR}(P)]_{\mathcal{D}}$ contient $[SS_{CR(\mathcal{D})}(P)]_{\mathcal{D}}]$. Il suffit alors, pour achever la preuve du théorème, de montrer que ce dernier ensemble contient $M^P_{\mathcal{D}}$. Soit a un élément de $M^P_{\mathcal{D}} = Rac^P_{\mathcal{D}}$. a est de la forme $p(\tilde{d})$ et est racine d'un $\mathcal{D}$-arbre de preuve A. Il existe un squelette fini complet S et une valuation v tels que $A = v_{\mathcal{D}}(S)$. La racine de S est étiquetée par une clause $H \leftarrow c \square B_1, \ldots, B_n$ telle que $v_{\mathcal{D}}(H) = a = p(\tilde{d})$. Donc $H = p(\tilde{t})$ et $v_{\mathcal{D}}(\tilde{t}) = \tilde{d}$. Considérons l'arbre S' construit de la manière suivante :

   - sa racine est étiquetée par $\leftarrow vrai \square p(\tilde{x})$,

   - il a un seul sous-arbre : le squelette S.

   S' est un squelette de racine $vrai \square p(\tilde{x})$. Sa collection de contraintes est $cont(S')$ $= cont(S) \bigcup \{vrai,\ p(\tilde{x}) = H\}$. Le symbole de prédicat de H étant p, $p(\tilde{x}) = H$ se résume à $\tilde{x} = \tilde{t}$. Dans $cont(S')$ les variables $\tilde{x}$ sont distinctes de toutes les variables apparaissant dans $cont(S)$. Donc la solution v de S se prolonge en une solution v' de S' en donnant aux variables $\tilde{x}$ les valeurs $v_{\mathcal{D}}(\tilde{t})$: $v'(\tilde{x}) = v_{\mathcal{D}}(\tilde{t})$. S' est donc un squelette $\mathcal{D}$-soluble de racine $vrai \square p(\tilde{x})$. Soit r sa contrainte réponse ; $p(\tilde{x}) \leftarrow r$ est dans $SS_{CR(\mathcal{D})}(P)$ et la valuation v', solution de r, est telle que $v'_{\mathcal{D}}(p(\tilde{x})) = p(v'(\tilde{x})) = p(v_{\mathcal{D}}(\tilde{t})) = p(\tilde{d}) = a$. Ceci prouve que a est dans $[SS_{CR(\mathcal{D})}(P)]_{\mathcal{D}}$.

5.2 Programme normalisé

Définition 5.5 *(Clause normalisée)*

Une clause de programme est dite normalisée *si elle est de la forme*

$$p(\tilde{x}) \leftarrow c \ \square \ B_1, \ldots, B_n$$

À une clause de programme quelconque $p(\tilde{t}) \leftarrow c \ \square \ B_1, \ldots, B_n$ *on associe une clause normalisée équivalente* $p(\tilde{x}) \leftarrow \tilde{x} = \tilde{t} \wedge c \ \square \ B_1, \ldots, B_n$ *où* $\tilde{x}$ *est un ensemble de variables distinctes non libres dans la clause initiale.*

L'équivalence entre la clause initiale et la clause normalisée a été étudiée au paragraphe 2.5 à propos du complété.

Définition 5.6 *(Programme normalisé)*
Étant donné un programme P*, un programme normalisé* P' *est obtenu en remplaçant chaque clause de* P *par une clause normalisée équivalente.*

Les programmes normalisés équivalents à P sont identiques au nom près des variables nouvelles choisies dans les clauses. Nous les appellerons donc "le programme normalisé" équivalent à P.

Exemple 5.1 *Le programme Fib :*
$fib(0,1) \leftarrow vrai$
$fib(1,1) \leftarrow vrai$
$fib(N, A+B) \leftarrow N > 1 \ \square \ fib(N-2, A), fib(N-1, B)$
est équivalent au programme normalisé Fib' *:*
$fib(X,Y) \leftarrow X = 0 \wedge Y = 1$
$fib(X,Y) \leftarrow X = 1 \wedge Y = 1$
$fib(X,Y) \leftarrow X = N \wedge Y = A + B \wedge N > 1 \ \square \ fib(N-2, A), fib(N-1, B)$

Étant donné un but B, les réponses pour B dans P et dans P' se correspondent : si S est un squelette réponse pour B dans P il lui correspond un squelette normalisé S' obtenu en remplaçant les clauses par des clauses normalisées équivalentes. Les contraintes associées $ca(S)$ et $ca(S')$ sont équivalentes (au sens $\models c \leftrightarrow c'$) car la clause $p(\tilde{t}) \leftarrow c \ \square \ B_1, \ldots, B_n$ étiquetant un nœud N de S où $A = atome(N)$, produit dans $cont(S)$ les contraintes
c,
$p(\tilde{t}) = A$.
La clause normalisée $p(\tilde{x}) \leftarrow \tilde{x} = \tilde{t} \wedge c \ \square \ B_1, \ldots, B_n$ produit dans $cont(S')$
c,
$\tilde{x} = \tilde{t}$,
$p(\tilde{x}) = A$.
La partie $p(\tilde{t}) = A$ de $ca(S)$ est équivalente à la partie $\exists \tilde{x}(\tilde{x} = \tilde{t} \wedge p(\tilde{x}) = A)$ de $ca(S')$.

On appelle *réponse normalisée* un squelette réponse dans le programme normalisé ; la contrainte associée est dite *contrainte réponse normalisée.*

5.3 Règles d'induction sur les atomes contraints

Nous étudions ici en termes de systèmes inductifs le problème évoqué par [13] et [10] : l'ensemble succès peut être obtenu comme point fixe d'un opérateur sur les ensembles d'atomes contraints.

On déduit du programme P et du critère de rejet CR un ensemble de règles d'induction sur les atomes contraints. Nous avons vu que tout programme a un équivalent normalisé. Nous supposons désormais que le programme P est normalisé.

Définition 5.7 *(Règles associées à une clause et à un programme)*
Étant donné une clause (normalisée) $C : p(\tilde{x}) \leftarrow c \square p_1(\tilde{t_1}), \ldots, p_n(\tilde{t_n})$,
le système de règles associé à C *est l'ensemble* $R(C)$ *des règles*
$(p(\tilde{x}) \leftarrow r) \leftarrow (p_1(\tilde{x_1}) \leftarrow r_1), \ldots, (p_n(\tilde{x_n}) \leftarrow r_n)$
vérifiant :

- *les variables* $\tilde{x_1}, \ldots, \tilde{x_n}$ *sont toutes distinctes et distinctes de* $\tilde{x}$,
- $\models r \leftrightarrow \exists_{-\tilde{x}}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge r_i))$,
- r *n'est pas rejetée par* CR.

Le système de règles $R(P)$ *associé au programme* P *est l'union des* $R(C)$ *sur les clauses* C *de* P.

Exemple 5.2 *Le programme normalisé* Fib' :
$fib(X, Y) \leftarrow X = 0 \wedge Y = 1$
$fib(X, Y) \leftarrow X = 1 \wedge Y = 1$
$fib(X, Y) \leftarrow X = N \wedge Y = A + B \wedge N > 1 \square fib(N - 2, A), fib(N - 1, B)$
admet pour système de règles selon le critère de rejet $CR(\mathcal{N})$:
$(fib(X, Y) \leftarrow X = 0 \wedge Y = 1) \leftarrow$
$(fib(X, Y) \leftarrow X = 1 \wedge Y = 1) \leftarrow$
et toutes les
$(fib(X, Y) \leftarrow r) \leftarrow (fib(X_1, Y_1) \leftarrow r_1), (fib(X_2, Y_2) \leftarrow r_2)$
où r_1, r_2, r *sont des contraintes telles que* r *est satisfiable dans* $\mathcal{N}$ *et*
$\models r \leftrightarrow \exists N \exists A \exists B \exists X_1 \exists Y_1 \exists X_2 \exists Y_2 (X = N \wedge Y = A + B \wedge N > 1 \wedge X_1 = N - 2 \wedge Y_1 = A \wedge r_1 \wedge X_2 = N - 1 \wedge Y_2 = B \wedge r_2)$

On déduit du système de règles $R(P)$

- l'opérateur de conséquence immédiate T_{CR}^P,
- son plus petit point fixe $pppf(T_{CR}^P)$,
- des arbres de preuve dits *arbres de preuve contraints* dont les nœuds sont étiquetés par des atomes contraints,
- l'ensemble des racines d'arbres de preuves contraints Rac_{CR}^P.

D'après ce qui a été montré sur les systèmes inductifs,

$$pppf(T_{CR}^P) = Rac_{CR}^P = T_{CR}^P \uparrow \omega$$

5.4 Ensemble succès et arbres de preuve contraints

Nous allons montrer que l'ensemble $pppf(T^P_{CR}) = Rac^P_{CR} = T^P_{CR} \uparrow \omega$ déduit du système inductif coïncide avec $SS_{CR}(P)$.

Commençons par étudier les clôtures existentielles dans les contraintes, et la correspondance entre les squelettes réponses pour les buts $\leftarrow vrai \square p(\tilde{x})$ et les arbres de preuve contraints.

Lemme 5.1 *(Clôtures existentielles)*
Soit f et g deux formules de $\mathcal{L}(V, \Sigma, \Pi_C)$ telles que les variables libres de g comprennent celles de f ; soit $\tilde{y}$ l'ensemble des variables libres dans g mais pas dans f.

1. $\models f \leftrightarrow \exists \tilde{y}\ f$
2. $\models (f \wedge \exists \tilde{y}\ g) \leftrightarrow \exists \tilde{y}(f \wedge g)$

Preuve : Pour montrer que deux formules a et b sont équivalentes au sens $\models a \leftrightarrow b$, il faut prouver que pour toute interprétation $\mathcal{D}$ de $\mathcal{L}(V, \Sigma, \Pi_C)$ et toute valuation v, $v_{\mathcal{D}}(a) = v_{\mathcal{D}}(b)$, ou encore que $v_{\mathcal{D}}(a) = vrai$ ssi $v_{\mathcal{D}}(b) = vrai$.

1. Soit $\tilde{x}$ l'ensemble des variables libres de f ; $\tilde{x}$ et $\tilde{y}$ sont disjoints. Soit $\mathcal{D}$ une interprétation de $\mathcal{L}(V, \Sigma, \Pi_C)$ et v une valuation dans $\mathcal{D}$. La valeur $v_{\mathcal{D}}(f)$ ne dépend que de la partie de v portant sur $\tilde{x}$; supposons donc que v soit restreinte à $\tilde{x}$.
   - Si $v_{\mathcal{D}}(f) = vrai$, tout prolongement v' de v à $\tilde{x} \bigcup \tilde{y}$ est tel que $v'_{\mathcal{D}}(f) = v_{\mathcal{D}}(f) = vrai$. On déduit qu'il existe un prolongement v' tel que $v'_{\mathcal{D}}(f) = vrai$ et donc que $v_{\mathcal{D}}(\exists \tilde{y}\ f) = vrai$.
   - Inversement, si $v_{\mathcal{D}}(\exists \tilde{y}\ f) = vrai$, il existe un prolongement v' de v à $\tilde{x} \bigcup \tilde{y}$ tel que $v'_{\mathcal{D}}(f) = vrai$; or $v'_{\mathcal{D}}(f) = v_{\mathcal{D}}(f)$; donc $v_{\mathcal{D}}(f) = vrai$.
2. Soit $\mathcal{D}$ une interprétation de $\mathcal{L}(V, \Sigma, \Pi_C)$ et v une valuation restreinte aux variables libres $\tilde{x}$ de f.
   - Si $v_{\mathcal{D}}(f \wedge \exists \tilde{y}\ g) = vrai$, $v_{\mathcal{D}}(f) = vrai$ et $v_{\mathcal{D}}(\exists \tilde{y}\ g) = vrai$, donc il existe un prolongement v' de v à $\tilde{x} \bigcup \tilde{y}$ tel que $v'_{\mathcal{D}}(g) = vrai$. Or $v'_{\mathcal{D}}(f) = v_{\mathcal{D}}(f) = vrai$. Donc $v'_{\mathcal{D}}(f \wedge g) = vrai$; donc $v_{\mathcal{D}}(\exists \tilde{y}(f \wedge g)) = vrai$.
   - Inversement, si $v_{\mathcal{D}}(\exists \tilde{y}(f \wedge g)) = vrai$, il existe un prolongement v' de v à $\tilde{x} \bigcup \tilde{y}$ tel que $v'_{\mathcal{D}}(f \wedge g) = vrai$. Alors $v'_{\mathcal{D}}(f) = vrai$ et $v'_{\mathcal{D}}(g) = vrai$. Or $v_{\mathcal{D}}(f) = v'_{\mathcal{D}}(f)$; donc $v_{\mathcal{D}}(f) = vrai$. D'autre part $v'_{\mathcal{D}}(g) = vrai$ signifie : $v_{\mathcal{D}}(\exists \tilde{y}\ g) = vrai$. Donc $v_{\mathcal{D}}(f \wedge \exists \tilde{y}\ g) = vrai$.

Définition 5.8 *(Squelette primitif)*
On appelle squelette primitif *un squelette fini complet dont tous les nœuds, y compris la racine, sont étiquetés par des classes d'équivalence de clauses (par variantes) du programme P.*

Rappelons que nous supposons que P est normalisé ; les clauses intervenant dans un squelette primitif sont donc normalisées.

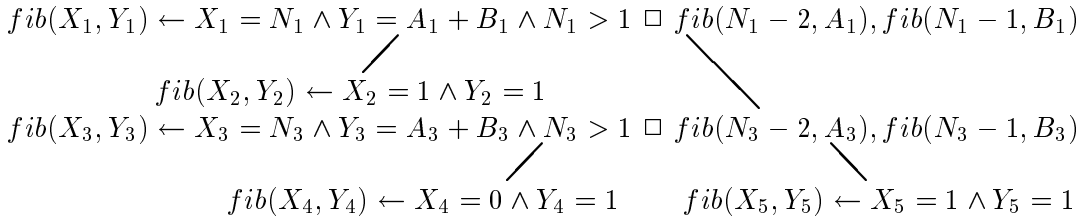

Figure 25 : Squelette S_{13}

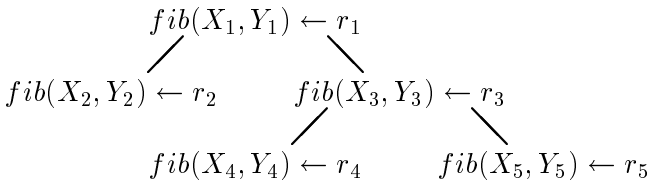

Figure 26 : Arbre contraint AC_1

Définition 5.9 *(Arbre contraint associé à un squelette primitif)*

Un arbre contraint est un arbre étiqueté par des atomes contraints. Les arbres de preuve contraints sont des arbres contraints. Étant donné un squelette primitif S et une fonction de choix de variantes de clauses (qui sert à construire $cont(S)$), l'arbre contraint associé *à $\mathcal{C}(S)$ est l'arbre A comportant à l'étiquetage près les mêmes nœuds que S, et pour tout nœud N de S, soit $p(\tilde{x})$ la tête de la variante de clause étiquetant N, soit S_N le squelette extrait de S ayant N pour racine, alors le nœud N' correspondant dans A est étiqueté par l'atome contraint $p(\tilde{x}) \leftarrow \exists_{-\tilde{x}} \mathcal{C}(S_N)$.*

Rappelons que pour un squelette fini S, $\mathcal{C}(S)$ est la conjonction des contraintes de $cont(S)$.

Exemple 5.3 *Pour le programme normalisé Fib' (exemple 5.1) le squelette S_{13} de la figure 25 a pour arbre contraint associé l'arbre AC_1 de la figure 26, où*

$r1 = \exists N_1 \exists A_1 \exists B_1 \exists X_2 \exists Y_2 \exists X_3 \exists Y_3 \exists N_3 \exists A_3 \exists B_3 \exists X_4 \exists Y_4 \exists X_5 \exists Y_5 (X_1 = N_1 \wedge Y_1 = A_1 + B_1 \wedge N_1 > 1 \wedge X_2 = N_1 - 2 \wedge Y_2 = A_1 \wedge X_2 = 1 \wedge Y_2 = 1 \wedge X_3 = N_1 - 1 \wedge Y_3 = B_1 \wedge X_3 = N_3 \wedge Y_3 = A_3 + B_3 \wedge N_3 > 1 \wedge X_4 = N_3 - 2 \wedge Y_4 = A_3 \wedge X_4 = 0 \wedge Y_4 = 1 \wedge X_5 = N_3 - 1 \wedge Y_5 = B_3 \wedge X_5 = 1 \wedge Y_5 = 1)$,

$r_2 = X_2 = 1 \wedge Y_2 = 1$,

$r_3 = \exists N_3 \exists A_3 \exists B_3 \exists X_4 \exists Y_4 \exists X_5 \exists Y_5 (X_3 = N_3 \wedge Y_3 = A_3 + B_3 \wedge N_3 > 1 \wedge X_4 = N_3 - 2 \wedge Y_4 = A_3 \wedge X_4 = 0 \wedge Y_4 = 1 \wedge X_5 = N_3 - 1 \wedge Y_5 = B_3 \wedge X_5 = 1 \wedge Y_5 = 1)$,

$r_4 = X_4 = 0 \wedge Y_4 = 1$,

$r_5 = X_5 = 1 \wedge Y_5 = 1$.

Lemme 5.2 *(Arbre de preuve contraint associé à un squelette primitif)*

Si S est un squelette primitif acceptable pour CR alors l'arbre contraint associé $CA(S)$ est un arbre de preuve contraint.

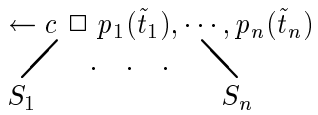

Figure 27 : Squelette primitif S

Preuve : La preuve se fait par récurrence sur la hauteur du squelette primitif acceptable S.

1. Si S est de hauteur 1, il est réduit à un seul nœud étiqueté par un fait : $p(\tilde{x}) \leftarrow c$ auquel correspond la règle d'induction $(p(\tilde{x}) \leftarrow \exists_{-\tilde{x}} c) \leftarrow$. Alors $CA(S)$ est réduit à un nœud unique étiqueté par l'atome contraint $p(\tilde{x}) \leftarrow \exists_{-\tilde{x}} c$ qui est bien prouvé par la règle ci-dessus ; $\exists_{-\tilde{x}} c$ n'est pas rejetée par CR car c'est $\exists_{-\tilde{x}} \mathcal{C}(S)$ et S est acceptable.

2. Supposons que pour tout squelette primitif acceptable de hauteur au plus h l'arbre contraint associé soit un arbre de preuve contraint ; soit S un squelette primitif acceptable de hauteur $h+1$. S a la forme indiquée figure 27 où les S_i sont des squelettes primitifs acceptables de hauteur au plus h. Donc chaque $AC(S_i)$ est un arbre de preuve contraint. La racine de S_i est étiquetée par $q_i(\tilde{x_i}) \leftarrow \exists_{-\tilde{x_i}} \mathcal{C}(S_i)$. S étant acceptable, p_i et q_i sont le même symbole de prédicat ; donc la racine de S_i est étiquetée par $p_i(\tilde{x_i}) \leftarrow \exists_{-\tilde{x_i}} \mathcal{C}(S_i)$. La racine de $AC(S)$ est étiquetée par $p(\tilde{x}) \leftarrow \exists_{-\tilde{x}} \mathcal{C}(S)$ que l'on peut encore écrire $p(\tilde{x}) \leftarrow r$ où $r = \exists_{-\tilde{x}}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge \mathcal{C}(S_i)))$. À la clause étiquetant la racine de S sont associées les règles d'induction $(p(\tilde{x}) \leftarrow r_0) \leftarrow (p_1(\tilde{x_1}) \leftarrow r_1), \ldots, (p_n(\tilde{x_n}) \leftarrow r_n)$ où r_0 est non rejetée et équivalente à $\exists_{-\tilde{x}}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge r_i))$. À condition que la contrainte r' soit non rejetée, on peut déduire des racines des $AC(S_i)$ l'atome contraint $p(\tilde{x}) \leftarrow r'$ où $r' = \exists_{-\tilde{x}}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge \exists_{-\tilde{x_i}} \mathcal{C}(S_i)))$.

   Il reste à prouver que r' est équivalente à r (qui intervient dans la racine de $AC(S)$) ; on en déduira, d'une part que r' n'est pas rejetée (parce que r ne l'est pas), d'autre part que $AC(S)$ est un arbre de preuve contraint. Soit $\tilde{y_i}$ l'ensemble des variables libres autres que $\tilde{x_i}$ dans S_i ; soit $\tilde{y}$ l'ensemble des variables libres autres que $\tilde{x}$ dans la clause racine de S. Si les variantes de clauses ont été correctement choisies en vue de la construction de $cont(S)$, les ensembles $\tilde{x}$, $\tilde{y}$, $\tilde{x_i}$, $\tilde{y_i}$ sont deux à deux disjoints. r et r' se récrivent :

   $r = \exists\tilde{y}\exists\tilde{x_1}, \ldots, \exists\tilde{x_n}\exists\tilde{y_1}, \ldots, \exists\tilde{y_n}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge \mathcal{C}(S_i)))$

   $r' = \exists\tilde{y}\exists\tilde{x_1}, \ldots, \exists\tilde{x_n}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge \exists\tilde{y_i}\mathcal{C}(S_i)))$

   D'après le lemme sur les clôtures existentielles, r' est équivalente aux contraintes suivantes, dont la dernière est r :

   $\exists\tilde{y}\exists\tilde{x_1}, \ldots, \exists\tilde{x_n}(c \wedge \bigwedge_{i=1}^{n} \exists\tilde{y_i}(\tilde{x_i} = \tilde{t_i} \wedge \mathcal{C}(S_i)))$

   $\exists\tilde{y}\exists\tilde{x_1}, \ldots, \exists\tilde{x_n}(c \wedge \bigwedge_{i=1}^{n} \exists\tilde{y_1}, \ldots, \exists\tilde{y_n}(\tilde{x_i} = \tilde{t_i} \wedge \mathcal{C}(S_i)))$

   $\exists\tilde{y}\exists\tilde{x_1}, \ldots, \exists\tilde{x_n}\exists\tilde{y_1}, \ldots, \exists\tilde{y_n}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge \mathcal{C}(S_i)))$

Définition 5.10 *(Équivalence entre arbres de preuve contraints)*
Deux arbres de preuve contraints A_1 et A_2 sont équivalents, ce qui est noté $A_1 \cong A_2$, si leurs nœuds sont en bijection et si les nœuds qui se correspondent sont étiquetés par des atomes contraints équivalents : $p(\tilde{x}) \leftarrow c$ et $p(\tilde{x}) \leftarrow c'$ avec $\models c \leftrightarrow c'$.

Lemme 5.3 *(Squelette primitif associé à un arbre de preuve contraint)*
Pour tout arbre de preuve contraint A il existe un squelette primitif acceptable S tel que $A \cong CA(S)$.

Preuve : Étant donné un arbre de preuve contraint A on construit le squelette S de la manière suivante :
- il a les mêmes nœuds que A aux étiquettes près,
- soit N un nœud de A, il est étiqueté par $p(\tilde{x}) \leftarrow r$; il a n fils N_1, ..., N_n ; N_i est étiqueté par $p_i(\tilde{x_i}) \leftarrow r_i$; il existe une règle de $R(P)$ $(p(\tilde{x}) \leftarrow r') \leftarrow (p_1(\tilde{x_1}) \leftarrow r_1), \ldots, (p_n(\tilde{x_n}) \leftarrow r_n)$ et $\models r \leftrightarrow r'$. Cette règle est issue d'au moins une clause normalisée de la forme : $C = p(\tilde{x}) \leftarrow c \square p_1(\tilde{t_1}), \ldots, p_n(\tilde{t_n})$ et $\models r \leftrightarrow r' \leftrightarrow \exists_{-\tilde{x}}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge r_i))$ et r n'est pas rejetée. Le nœud correspondant N' dans S est étiqueté par l'une des clauses C. S est un squelette primitif car il respecte la règle de concordance entre l'arité d'un nœud et l'arité de la clause qui l'étiquette.

Nous allons montrer par récurrence sur la hauteur de A que l'étiquette $p(\tilde{x}) \leftarrow r$ de la racine de A est telle que $\models r \leftrightarrow \exists_{-\tilde{x}}\mathcal{C}(S)$.

1. Si A est de hauteur 1, il est réduit à un nœud unique étiqueté par $p(\tilde{x}) \leftarrow r$ et $(p(\tilde{x}) \leftarrow r) \leftarrow$ est une règle de $R(P)$. La clause C qui étiquette l'unique nœud de S permet d'engendrer la règle ci-dessus. Donc C est de la forme $p(\tilde{x}) \leftarrow c$ et $\models r \leftrightarrow \exists_{-\tilde{x}}c$. Or $\mathcal{C}(S) = c$. Donc $\models r \leftrightarrow \exists_{-\tilde{x}}\mathcal{C}(S)$.

2. Si la propriété est vraie pour tout arbre de preuve contraint de hauteur au plus h, soit A de hauteur $h+1$. A et S sont représentés figures 28 et 29. Chaque A_i a pour racine $p(\tilde{x_i}) \leftarrow r_i$ et (par hypothèse de récurrence) $\models r_i \leftrightarrow \exists_{-\tilde{x_i}}\mathcal{C}(S_i)$. La clause C engendre la règle $p(\tilde{x}) \leftarrow (p_1(\tilde{x_1}) \leftarrow r_1), \ldots, (p_n(\tilde{x_n}) \leftarrow r_n)$ donc C est de la forme $p(\tilde{x}) \leftarrow c \square p_1(\tilde{t_1}), \ldots, p_n(\tilde{t_n})$ et $\models r \leftrightarrow \exists_{-\tilde{x}}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge r_i))$ donc $\exists_{-\tilde{x}}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge \exists_{-\tilde{x_i}}\mathcal{C}(S_i)))$. Nous avons montré dans le lemme précédent (arbre de preuve contraint associé à un squelette primitif) que cette dernière contrainte est équivalente à $\exists_{-\tilde{x}}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge \mathcal{C}(S_i)))$ qui n'est autre que $\exists_{-\tilde{x}}\mathcal{C}(S)$. Nous en déduisons que

   - S est acceptable parce que $\exists_{-\tilde{x}}\mathcal{C}(S)$ est équivalente à la contrainte r de la racine $p(\tilde{x}) \leftarrow r$ de A,

   - $CA(S) \cong A$.

Théorème 5.2 $SS_{CR}(P) = Rac^P_{CR} = pppf(T^P_{CR}) = T^P_{CR} \uparrow \omega$.

Preuve : Les deux dernières égalités sont déjà prouvées comme conséquences des propriétés des systèmes inductifs. On prouve la première égalité en montrant que $SS_{CR}(P)$ est inclus dans Rac^P_{CR}, puis que Rac^P_{CR} est inclus dans $SS_{CR}(P)$.

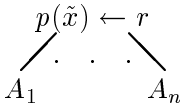

Figure 28 : Arbre de preuve contraint A

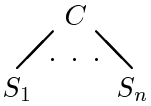

Figure 29 : Squelette S

1. Soit $p(\tilde{x}) \leftarrow r$ un atome contraint élément de $SS_{CR}(P)$. r est une contrainte réponse pour le but $\leftarrow vrai \square p(\tilde{x})$, donc r est équivalente à la contrainte associée à un squelette fini acceptable S représenté figure 30 où S' est un squelette primitif acceptable. Puisque S est acceptable, la racine de S' est étiquetée par une clause (normalisée) qui a p pour symbole de prédicat de tête : $p(\tilde{y}) \leftarrow c \square B_1, \ldots, B_n$. D'après le lemme "arbre de preuve contraint associé à un squelette primitif" il existe un arbre de preuve contraint A', déduit de S' et de racine $p(\tilde{y}) \leftarrow r'$ où r' est telle que $\models r' \leftrightarrow \exists_{-\tilde{y}} \mathcal{C}(S')$. Nous noterons plutôt $r'(\tilde{y})$ cette contrainte pour rappeler qu'elle dépend des variables libres $\tilde{y}$. On a

   $\models r \leftrightarrow \exists_{-\tilde{x}} \mathcal{C}(S)$

   donc

   $\models r \leftrightarrow \exists_{-\tilde{x}} (vrai \wedge \tilde{x} = \tilde{y} \wedge \mathcal{C}(S'))$

   $\models r \leftrightarrow \exists_{-\tilde{x}} (\tilde{x} = \tilde{y} \wedge \mathcal{C}(S'))$

   $\models r \leftrightarrow \exists_{-\tilde{x}} (\tilde{x} = \tilde{y} \wedge \exists_{-\tilde{y}} \mathcal{C}(S'))$ (d'après le lemme "clôtures existentielles")

   $\models r \leftrightarrow \exists \tilde{y} (\tilde{x} = \tilde{y} \wedge r'(\tilde{y}))$

   L'existence de variables $\tilde{y}$ telles que $\tilde{x} = \tilde{y} \wedge r'(\tilde{y})$ est évidemment équivalente à $r'(\tilde{x})$ obtenue en remplaçant $\tilde{y}$ par $\tilde{x}$ dans r' ; le remplacement de $\tilde{y}$ par $\tilde{x}$ en racine de S' correspond au remplacement de $p(\tilde{y}) \leftarrow r'(\tilde{y})$ par $p(\tilde{x}) \leftarrow r'(\tilde{x})$ en racine de A'. Ce dernier atome contraint est dans Rac^P_{CR} qui est défini à une équivalence près sur les contraintes ; or $\models r \leftrightarrow r'(\tilde{x})$; donc $p(\tilde{x}) \leftarrow r$ est dans Rac^P_{CR}.

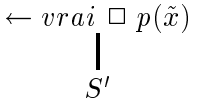

Figure 30 : Squelette fini acceptable S

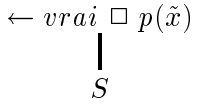

Figure 31 : Squelette S'

2. Si $p(\tilde{x}) \leftarrow r(\tilde{x})$ est dans Rac^P_{CR}, il est racine d'un arbre de preuve contraint A. D'après le lemme "squelette primitif associé à un arbre de preuve contraint", il existe un squelette primitif acceptable S tel que $A \cong CA(S)$. On a $\models r(\tilde{x}) \leftrightarrow \exists_{-\tilde{x}}\mathcal{C}(S)$.

   Considérons le squelette S' de la figure 31.

   Le calcul de sa contrainte réponse provoque le remplacement de $p(\tilde{x}) \leftarrow \ldots$ à la racine de S par $p(\tilde{y}) \leftarrow \ldots$ ce qui donne un nouveau squelette S_1. Alors $\mathcal{C}(S') = \exists_{-\tilde{x}}(vrai \wedge \tilde{x} = \tilde{y} \wedge \mathcal{C}(S_1))$.

   D'après le lemme "clôtures existentielles", on a $\models \mathcal{C}(S') \leftrightarrow \exists_{-\tilde{x}}(\tilde{x} = \tilde{y} \wedge \exists_{-\tilde{y}}\mathcal{C}(S_1))$,

   donc $\models \mathcal{C}(S') \leftrightarrow \exists_{-\tilde{x}}(\tilde{x} = \tilde{y} \wedge r(\tilde{y}))$

   et enfin $\models \mathcal{C}(S') \leftrightarrow r(\tilde{x}))$

   ce qui prouve que $p(\tilde{x}) \leftarrow r(\tilde{x})$ est dans $SS_{CR}(P)$.

[13] et [10] annoncent à peu près le même résultat sans le prouver.

[10] n'envisage pas de fermeture existentielle sur la contrainte réponse, alors que cela présente l'avantage d'abstraire par rapport aux variables auxiliaires introduites dans la dérivation conduisant à la réponse. L'égalité $SS_3(P, \mathcal{R}) = Fix_3(P, \mathcal{R})$ semble donc vérifiée. Par contre, le théorème 5.9 qui affirme que $P \models_{\mathcal{R}} c \,\square\, p(\tilde{x})$ ssi $p(\tilde{x}) \leftarrow c \in SS_2(P, \mathcal{R})$ lorsque c est finie nous semble inexact : pour le programme de la remarque 4.1, $vrai \,\square\, p(X)$ est conséquence logique de P dans $\mathcal{N}$ sans que $p(X) \leftarrow vrai$ soit dans $SS_2(P, \mathcal{N})$.

5.5 Atomes contraints et conséquences logiques

La présentation des liens entre l'ensemble succès et les atomes contraints conséquences logiques du programme (dans certaines circonstances) est inspirée du chapitre "Preliminaries on Constraint Logic Programming" de [9].

Étant donné une théorie $\mathcal{T}$ on considère l'ensemble des atomes contraints conséquences logiques de $\mathcal{T}$ et du programme P : $\mathcal{L}_{\mathcal{T}}(P) = \{p(\tilde{x}) \leftarrow c \mid P, \mathcal{T} \models c \rightarrow p(\tilde{x})\}$.

Théorème 5.3 *$SS_{CR}(P)$ est inclus dans $\mathcal{L}_{\mathcal{T}}(P)$.*

Preuve : Si $p(\tilde{x}) \leftarrow c$ est dans $SS_{CR}(P)$, c est une contrainte réponse pour $\leftarrow vrai \,\square\, p(\tilde{x})$ dans le programme P. Indépendamment du critère CR on sait qu'alors $P \models c \rightarrow vrai \,\square\, p(\tilde{x})$, donc $P \models c \rightarrow p(\tilde{x})$, donc $P, \mathcal{T} \models c \rightarrow p(\tilde{x})$ pour toute théorie $\mathcal{T}$. On en déduit que $p(\tilde{x}) \leftarrow c$ est dans $\mathcal{L}_{\mathcal{T}}(P)$.

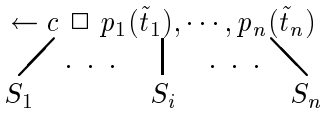

Figure 32 : Squelette réponse S

Définition 5.11 *(Couverture finie)*

Étant donné deux ensembles de contraintes A et B, A est finiment couvert par B *selon la théorie $\mathcal{T}$ si, pour toute contrainte $a \in A$ il existe une partie finie F de B telle que $\mathcal{T} \models a \rightarrow \bigvee_{b \in F} b$.*

Cette définition s'étend à deux ensembles d'atomes contraints : A est finiment couvert par B si pour tout atome contraint $p(\tilde{x}) \leftarrow c$ de A il existe dans B un nombre fini d'atomes contraints $\{p(\tilde{x}) \leftarrow r_i\}_{i=1,\ldots,n}$, tels que $\mathcal{T} \models c \rightarrow \bigvee_{i=1}^{n} r_i$.

Théorème 5.4 *Si le critère de rejet CR est en relation avec la théorie $\mathcal{T}$ alors $\mathcal{L}_{\mathcal{T}}(P)$ est finiment couvert par $SS_{CR}(P)$ selon $\mathcal{T}$.*

Preuve : Si $p(\tilde{x}) \leftarrow c$ est dans $\mathcal{L}_{\mathcal{T}}(P)$ alors $P, \mathcal{T} \models c \rightarrow p(\tilde{x})$; le troisième théorème de 4.3 (réponses selon CR en relation avec $\mathcal{T}$) assure qu'il existe un ensemble fini F de contraintes réponses selon CR tel que $\mathcal{T} \models c \rightarrow \bigvee_{r \in F} r$. L'ensemble d'atomes contraints $\{p(\tilde{x}) \leftarrow r \mid r \in F\}$ est une partie finie de $SS_{CR}(P)$ qui couvre $p(\tilde{x}) \leftarrow c$.

5.6 Équivalence entre le programme et l'ensemble succès

Des équivalences entre programmes contraints sont étudiées dans [10] et [3].

L'ensemble d'atomes contraints $SS_{CR}(P)$ peut être considéré comme un ensemble de faits, donc comme un programme dont toutes les clauses sont des faits. Pour abréger, notons P^+ ce programme. Nous allons montrer que P^+ est équivalent à P : pour tout but $\leftarrow b$, P et P^+ donnent les mêmes contraintes réponses.

Théorème 5.5 *Les deux programmes P et $SS_{CR}(P)$ donnent les mêmes contraintes réponses selon le critère CR.*

Preuve :

1. Soit r une contrainte réponse pour le but $B = \leftarrow c \,\square\, p_1(\tilde{t_1}), \ldots, p_n(\tilde{t_n})$ dans le programme P. Soit $\tilde{y}$ l'ensemble des variables libres de B. r est la contrainte associée au squelette réponse S de la figure 32. Le programme P est supposé normalisé, donc la racine de S_i est étiquetée par une clause $p_i(\tilde{x_i}) \leftarrow \ldots$. S_i est un squelette primitif acceptable (car partie de S acceptable). À S_i est donc associé un arbre de preuve contraint A_i de racine $p(\tilde{x_i}) \leftarrow r_i$ tel que $\models r_i \leftrightarrow \exists_{-\tilde{x_i}} \mathcal{C}(S_i)$. Or $r = \exists_{-\tilde{y}}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge \mathcal{C}(S_i)))$, donc
   $\models r \leftrightarrow \exists_{-\tilde{y}}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge \exists_{-\tilde{x_i}} \mathcal{C}(S_i)))$ (lemme "clôtures existentielles")
   $\models r \leftrightarrow \exists_{-\tilde{y}}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge r_i))$

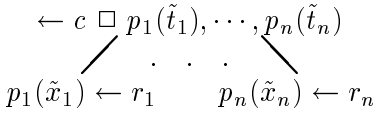

Figure 33 : Squelette S^+

Considérons le squelette S^+ de la figure 33. C'est un squelette pour le programme P^+, et sa contrainte associée est $\exists_{-\tilde{y}}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge r_i))$ qui est équivalente à r. Donc S^+ est acceptable, et r est une contrainte réponse dans P^+.

2. Soit r une contrainte réponse pour B dans P^+. Il existe un squelette S^+ dans P^+ représenté figure 33, et $r = \exists_{-\tilde{y}}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge r_i))$. Chaque $p_i(\tilde{x_i}) \leftarrow r_i$ est élément de $P^+ = SS_{CR}^P = Rac_{CR}^P$, donc est racine d'un arbre de preuve contraint A_i auquel correspond un squelette primitif acceptable S_i de racine $p(\tilde{x_i}) \leftarrow \ldots$ et $\models r_i \leftrightarrow \exists_{-\tilde{x_i}} \mathcal{C}(S_i)$.

   On considère le squelette S représenté figure 32. Alors $ca(S) = \exists_{-\tilde{y}}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge \mathcal{C}(S_i)))$ et

   $\models ca(S) \leftrightarrow \exists_{-\tilde{y}}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge \exists_{-\tilde{x_i}} \mathcal{C}(S_i)))$

   $\models ca(S) \leftrightarrow \exists_{-\tilde{y}}(c \wedge \bigwedge_{i=1}^{n}(\tilde{x_i} = \tilde{t_i} \wedge r_i))$

   $\models ca(S) \leftrightarrow r$

   ce qui prouve que r est, à une équivalence près, une contrainte réponse dans P.

5.7 Ensemble d'échec fini

L'ensemble d'échec fini est constitué d'atomes contraints qui sont d'échec fini selon le critère de rejet $CR(\mathcal{D})$.

On trouve sa définition, et ses rapports avec l'opérateur de conséquence immédiate, dans [12, 13, 15, 16].

Définition 5.12 *(Ensemble d'échec fini)*
L'ensemble d'échec fini associé au programme P et à la pré-interprétation $\mathcal{D}$ est :
$FF_{\mathcal{D}}(P) = \{p(\tilde{x}) \leftarrow c \mid \ \leftarrow c \square p(\tilde{x})$ *est d'échec fini selon* $CR(\mathcal{D})\}$.

Définition 5.13 *(Itération descendante)*
$T_{\mathcal{D}}^P \downarrow \omega$ *est la partie de la* $\mathcal{D}$*-base définie inductivement par :*
$T_{\mathcal{D}}^P \downarrow 0 = \mathcal{D}$*-base,*
$T_{\mathcal{D}}^P \downarrow (n+1) = T_{\mathcal{D}}^P(T_{\mathcal{D}}^P \downarrow n)$,
$T_{\mathcal{D}}^P \downarrow \omega \ = \ \bigcap_{n \in IN} T_{\mathcal{D}}^P \downarrow n$.

Définition 5.14 *(semi-$\mathcal{D}$-arbre de preuve)*
Un semi-$\mathcal{D}$-arbre de preuve est un arbre fini étiqueté par des $\mathcal{D}$-atomes tel que, pour tout nœud interne, s'il est étiqueté e et ses fils $e_1, \ldots, e_n$, alors $e \leftarrow e_1, \ldots, e_n$ est une règle de $R_{\mathcal{D}}(P)$.

Un semi-$\mathcal{D}$-arbre de preuve ressemble beaucoup à un $\mathcal{D}$-arbre de preuve, la seule différence étant que les feuilles du $\mathcal{D}$-arbre de preuve sont des têtes de faits, alors que celles du semi-$\mathcal{D}$-arbre de preuve sont des $\mathcal{D}$-atomes quelconques. Lorsque la feuille correspond à un fait, on dit qu'elle est *prouvée*, et que la branche correspondante est *terminée*.

Définition 5.15 *(Profondeur)*
Un semi-$\mathcal{D}$-arbre de preuve est de profondeur n si toutes ses branches non terminées sont de longueur n (en nombre d'arcs).

Définition 5.16 *(Développement d'un squelette partiel)*
Un squelette partiel est n-développé si toutes ses feuilles indéfinies sont à la profondeur $n+1$ (en nombre d'arcs).

Remarque 5.3 *Un squelette complet n'ayant aucune feuille indéfinie, il est n-développé pour tout n.*

Remarque 5.4 *(Semi-$\mathcal{D}$-arbres de preuve et squelettes partiels)*
Les semi-$\mathcal{D}$-arbres de preuve et les squelettes partiels solubles de racine atomique se correspondent en ce sens que :
- À tout semi-$\mathcal{D}$-arbre de preuve A correspond au moins un squelette partiel, primitif et soluble (comprenant des clauses engendrant les règles utilisées dans A). Ce squelette primitif peut être complété par un but atomique $\leftarrow vrai \ \Box \ p(\tilde{x})$ pour donner un squelette partiel S. Si A est de profondeur n alors S est n-développé.
- Pour tout squelette partiel soluble S il existe une valuation v solution de $cont(S)$. A tout nœud N de S autre que la racine est associé un atome noté $atome(N)$, même si N est indéfini. L'arbre étiqueté par les $v_{\mathcal{D}}(atome(N))$ est un semi-$\mathcal{D}$-arbre de preuve A. Si S est n-développé, alors A est de profondeur n.

Définition 5.17 *(Solution compacité)*
Le langage des contraintes est solution compact *relativement à la pré-interprétation $\mathcal{D}$ si pour toute contrainte c il existe un ensemble de contraintes $\{c_i\}_{i\in I}$ tel que $\models_{\mathcal{D}} \neg c \leftrightarrow \bigvee_{i\in I} c_i$.*

On remarquera que les variables libres des contraintes c_i sont les mêmes que celles de c.

Cette notion est définie dans [12, 15, 16, 13]. Nous n'en utilisons que la partie SC_2 conformément à [15, 16, 13].

Lemme 5.4 *(Squelettes n-développés et échec fini)*
Un but B est d'échec fini selon un critère de rejet CR si et seulement si il existe un entier positif n tel que B ne soit racine d'aucun squelette partiel acceptable n-développé.

Preuve : Nous allons montrer que B n'est pas d'échec fini ssi il n'y a pas de limitation sur la profondeur des squelettes partiels acceptables de racine B dont les branches non terminées ont même longueur.

1. Si B n'est pas d'échec fini, il admet une dérivation non échec équitable. Celle-ci calcule un squelette S complet de racine B, finiment acceptable. S est complet donc n-développé pour tout n.

2. S'il n'y pas de limitation sur la profondeur des squelettes partiels acceptables de racine B dont les branches non terminées ont même longueur, pour tout n il existe un squelette partiel n-développé S_n, de racine B, acceptable.
Considérons l'arbre $\mathcal{A}$ des squelettes partiels acceptables de racine B dont les branches non terminées ont même longueur. La racine de $\mathcal{A}$ est étiquetée par $S_0 = sq(B)$. Un nœud étiqueté par S' est fils d'un nœud étiqueté par S si S' s'obtient en prolongeant d'un cran toutes les branches non terminées de S (S' est un prolongement de S). $\mathcal{A}$ est à branchement fini car il n'existe qu'un nombre fini de possibilités de prolonger d'un cran toutes les branches non terminées d'un squelette. Pour tout n, $\mathcal{A}$ comprend une branche de longueur n : celle qui joint S_0 à S_n. D'après le lemme de Köenig, $\mathcal{A}$ comprend une branche infinie : une suite infinie $T_0 = S_0, T_1, \ldots, T_i, T_{i+1} \ldots$ de squelettes partiels acceptables de racine B tels que T_i est i-développé et T_{i+1} est un prolongement de T_i. On en déduit une dérivation infinie passant par les T_i. Cette dérivation est équitable car toute feuille indéfinie de T_i a été choisie lorsqu'est atteint T_{i+1}. Donc B n'est pas d'échec fini.

Théorème 5.6 *Si le langage des contraintes est solution compact relativement à $\mathcal{D}$, alors $[FF_{\mathcal{D}}(P)]_{\mathcal{D}} = \overline{T^P_{\mathcal{D}} \downarrow \omega}$.*

Preuve :

1. Pour prouver que $[FF_{\mathcal{D}}(P)]_{\mathcal{D}} \subseteq \overline{T^P_{\mathcal{D}} \downarrow \omega}$, il faut montrer que si $\leftarrow c \,\square\, p(\tilde{x})$ est d'échec fini et si v est solution de c, alors $p(v(\tilde{x})) \notin T^P_{\mathcal{D}} \downarrow \omega$. Soit $\tilde{d} = v(\tilde{x})$. Supposons que $p(\tilde{d}) \in T^P_{\mathcal{D}} \downarrow \omega$ et que la valuation notée $\tilde{x} := \tilde{d}$ dans laquelle $\tilde{x}$ a la valeur $\tilde{d}$ est solution de c. Alors, pour tout n il existe un semi-$\mathcal{D}$-arbre de preuve A_n de profondeur n et de racine $p(\tilde{d})$. À cet A_n correspond un squelette primitif soluble auquel on peut ajouter une racine étiquetée par $\leftarrow c \,\square\, p(\tilde{x})$ pour donner un squelette partiel S_n n-développé. S_n est soluble car on peut prolonger la valuation v solution du squelette primitif en une valuation dans laquelle $\tilde{x} = \tilde{d}$. D'après le lemme ci-dessus, $\leftarrow c \,\square\, p(\tilde{x})$ n'est pas d'échec fini.

2. La preuve de l'inclusion inverse $\overline{T^P_{\mathcal{D}} \downarrow \omega} \subseteq [FF_{\mathcal{D}}(P)]_{\mathcal{D}}$ utilise la solution compacité. Montrons que si $p(\tilde{d}) \notin T^P_{\mathcal{D}} \downarrow \omega$ alors $p(\tilde{d}) \in [FF_{\mathcal{D}}(P)]_{\mathcal{D}}$. Il existe un n tel que $p(\tilde{d}) \notin T^P_{\mathcal{D}} \downarrow n$, donc $p(\tilde{d})$ n'est racine d'aucun semi-$\mathcal{D}$-arbre de preuve de profondeur n. Considérons l'ensemble des squelettes partiels solubles n-développés de racine $\leftarrow vrai \,\square\, p(\tilde{x})$. C'est un ensemble fini : $\{S_i\}_{i=1,\ldots,m}$. À chaque S_i correspond une contrainte $c_i = ca(S_i)$ dans laquelle les variables autres que $\tilde{x}$ sont quantifiées existentiellement. c_i est satisfiable puisque S_i est soluble. Mais c_i est fausse dans la valuation $\tilde{x} := \tilde{d}$, sinon $v_{\mathcal{D}}(S_i)$ serait un semi-$\mathcal{D}$-arbre de preuve de racine $p(\tilde{d})$ et de profondeur n.

   - Si $m = 0$, alors $\leftarrow vrai \,\square\, p(\tilde{x})$ n'est racine d'aucun squelette partiel soluble n-développé. D'après le lemme ci-dessus, $\leftarrow vrai \,\square\, p(\tilde{x})$ est d'échec fini selon $CR(\mathcal{D})$ et $p(\tilde{d})$ en résulte par la valuation $\tilde{x} := \tilde{d}$.
   - Si $m \neq 0$, le langage des contraintes étant solution compact relativement à $\mathcal{D}$, pour chaque c_i il existe un ensemble de contraintes $\{c_i^j\}_{j \in J}$ tel que $\models_{\mathcal{D}} \neg c_i \leftrightarrow \bigvee_j c_i^j$. c_i est fausse dans la valuation $\tilde{x} := \tilde{d}$, donc il existe un j tel que c_i^j soit vraie dans $\tilde{x} := \tilde{d}$. Choisissons pour chaque c_i une telle

contrainte c_i^j que nous noterons c_i'. c_i' est vraie dans $\tilde{x} := \tilde{d}$, et $c_i \wedge c_i'$ est insatisfiable dans $\mathcal{D}$ car $\models_{\mathcal{D}} c_i' \rightarrow \neg c_i$. Considérons la contrainte $c = \bigwedge_{i=1}^{m} c_i'$, et le but $\leftarrow c \square p(\tilde{x})$. c est vraie dans $\tilde{x} := \tilde{d}$. Tout squelette S de racine $\leftarrow c \square p(\tilde{x})$ n-développé correspond à un squelette T de racine $\leftarrow vrai \square p(\tilde{x})$ dans lequel on a remplacé la contrainte $vrai$ par c. Si T est insoluble, S l'est aussi. Si T est soluble, il fait partie de l'ensemble $\{S_i\}_{i=1,\ldots,m}$. La contrainte associée à S est $c \wedge c_i$, qui comprend $c_i' \wedge c_i$. Elle est donc insatisfiable. Tout S est donc insoluble. D'après le lemme ci-dessus, $\leftarrow c \square p(\tilde{x})$ est d'échec fini. $p(\tilde{d})$ en résulte par la valuation $\tilde{x} := \tilde{d}$ qui est solution de c.

Références

[1] P. Aczel. An Introduction to Inductive Definition, in Handbook of Mathematical Logic, pp 739-782, (J. Barwise ed.). *North Holland*, 1977.

[2] K. Apt. Introduction to Logic Programming, in Handbook of Theoretical Computer Science, volume B, chapter 10, pp 491-574 (J. van Leeuwen ed.). *Elsevier*, 1990.

[3] A. Bossi, M. Gabrielli, G. Levi, M. Martelli, The s-semantics approach : theory and applications, in Journal of Logic Programming 19-20, pp 149-197, 1994

[4] P. Codognet, Programmation Logique avec Contraintes : une introduction, à paraître dans *TSI*, 1995.

[5] J. Cohen, Constraint Logic Programming Languages, *CACM*, 33, pages 52-68, juillet 1990.

[6] A. Colmerauer, An Introduction to Prolog III, *CACM*, 33, pages 69-90, juillet 1990.

[7] R. Cori, D. Lascar, Logique Mathématique (cours et exercices), *Masson*, 1993.

[8] P. Deransart, J. Maluszinski. A Grammatical View of Logic Programming. *MIT Press*, 1993.

[9] F. Fages, Constructive negation by pruning, Rapport LIENS 94-14, septembre 1994

[10] M. Gabrielli, G. Levi, Modeling answer constraints in constraint logic Programming, *ICLP'91*, pp 238-252, (V. J. Saraswat, K. Ueda ed.) *MIT Press*, 1991.

[11] N. Heintze, N. Jaffar, S. Michaylov, P. Stuckey, R. Yap, The CLP(R) programmer's manual version 1.2, *IBM T. J. Watson Research Center.*

[12] J. Jaffar, J. L. Lassez, Constraint Logic Programming, Technical Report 86/73, Department of Computer Science, Monash University, 1986 (et 14th ACM Symposium on Principles of Programming Languages, Munich Janvier 1987).

[13] J. Jaffar, M. Maher, Constraint Logic Programming : a survey, in Journal of Logic Programming, Vol. 19-20, pages 503-581, 1994.

[14] J. W. Lloyd, Foundations of Logic Programming, *Springer Verlag, 2nd ed.*, 1987.

[15] M.J. Maher, A CLP View of Logic Programming, Proc. Conf. on Algebraic and Logic Programming, *LNCS 632*, pages 364-383, 1992.

[16] M.J. Maher, A Logic Programming view of CLP, *ICLP'93* , pages 737-753, *Warren ed.*, 1993.