INFORMATIQUE THÉORIQUE ET APPLICATIONS

G. BLANC
N. BLEUZEN-GUERNALEC

Algèbres effectives dans la programmation logique avec contraintes

Informatique théorique et applications, tome 26, nº 3 (1992), p. 221-242
<http://www.numdam.org/item?id=ITA_1992__26_3_221_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 26, n° 3, 1992, p. 221 à 242)

ALGÈBRES EFFECTIVES DANS LA PROGRAMMATION LOGIQUE AVEC CONTRAINTES (*)

par G. BLANC ([1]) et N. BLEUZEN-GUERNALEC ([1])

Communiqué par J. E. PIN

Résumé. – *Des structures suffisamment générales pour la résolution de contraintes ont pu être définies pour que reste valide l'essentiel de la sémantique déclarative de la Programmation Logique avec Contraintes (Prolog II, III, CLP($\mathcal{R}$)). En particulier, tous les éléments d'une telle structure doivent être définissables par contraintes.*

La sous-structure constituée des éléments finiment définissables (seuls éléments effectifs procéduralement) ne satisfait toutefois pas exactement les mêmes propriétés déclaratives que la structure elle-même. Nous étudions cette sous-algèbre effective, et obtenons des conditions suffisantes à l'« Invariance de Résolution » des contraintes : une même implémentation permet alors le traitement des contraintes dans le modèle théorique ou dans sa sous-algèbre effective.

Abstract. – *Sufficiently general structures have been defined in such a way that the main points of the declarative semantics of Constraint Logic Programming remain valid (Prolog II, III, CLP($\mathcal{R}$)). Particularly, each element of such a structure must be definable in terms of constraints.*

From a procedural point of view, the only effective terms are the finitely definable ones. The sub-structure of those terms has not yet exactly the same declarative properties as the whole structure. We study that effective sub-algebra and obtain sufficient conditions (concerning constraints) for "Resolution Invariability": the same implementation allows the constraints to be treated in the theoretical model as well as in its effective sub-algebra.

1. INTRODUCTION

La Programmation Logique avec Contraintes trouve son origine dès lors qu'on accepte de modéliser l'implémentation en l'absence de contrôle d'occurrence [Col. 7, Ja.St. 17].

Rappelons que le contrôle d'occurrence [Llo. 18] est le test de l'algorithme d'unification qui donne un échec lorsqu'une variable x doit être unifiée à une

(*) Reçu octobre 1990, révisé en juin 1991.

([1]) Département de Mathématiques, Faculté des Sciences de Luminy, 163, avenue de Luminy, 13288 Marseille Cedex 9.

0988-3754/92/03 221 22/\$4.20/

expression t contenant x. Dès lors que ce test est supprimé, l'unification conduit à accepter des solutions à $x=t$.

Ainsi, par exemple, unifier y à la constante b et x à $f(x, y)$ n'aboutit plus à un échec. Il devient nécessaire d'accepter que le système

$$S \equiv \{ x=f(x, y); y=b \}$$

admet des solutions, donc que $f(f(f(\ldots, b), b), b)$ est un objet à considérer, bien que non descriptible en notation préfixée habituelle. Les structures de données peuvent de la sorte être enrichies [Coup 9].

Les « nouveaux » termes constants ne peuvent évidemment figurer dans un programme que par les sytèmes d'équations (les contraintes) qui les caractérisent.

Il faut souligner que la suppression du contrôle d'occurrence a rendu particulièrement explicite le fait que l'unification n'est qu'une résolution de contraintes dans une certaine structure :

– résolution de systèmes d'égalités dans l'algèbre de Herbrand (unification avec contrôle d'occurrence);

– résolution de systèmes d'égalités dans l'algèbre des arbres finis ou infinis (unification sans contrôle d'occurrence).

Partant de là, pourquoi ne pas s'autoriser l'expressivité d'inéquations, puis d'inégalités entre nombres, puis de tout un traitement de contraintes arithmétiques, ce point de vue finissant par donner naissance aux différents Prolog II, III, CLP($\mathscr{R}$), CLP(X) [Col. 8, Ja.La. 14].

Le modèle algébrique fourni par l'algèbre des termes généraux (arbres quelconques, finis ou infinis) sur un alphabet de symboles fonctionnels [Cour. 10, Ja.St. 17, Ar.Ni. 1] – algèbre complète de Herbrand $\mathbb{H}^{\omega}$ [Llo. 18] – reste l'exemple type le plus intuitif de structure dans laquelle vont se résoudre les contraintes.

Il est évident que les possibilités pratiques ainsi offertes apportent beaucoup à l'efficacité des différentes implémentations [Col. 7, Col. 8, B.B.C. 4, Coup. 9].

La nature différente des contraintes que l'on peut s'autoriser à exprimer, et donc des différentes structures que l'on peut envisager pour la résolution de ces contraintes, oblige dans chaque cas à se demander si le paradigme de la Programmation Logique ne s'en trouve pas altéré [J.L.M. 16]. C'est en effet l'équivalence entre les diverses sémantiques, logique : $P \vdash A$, $P \models A$; algébrique : $P \models_{\mathbb{H}} A$ dans l'algèbre de Herbrand $\mathbb{H}$, $A \in$ plus petit modèle de P; fonctionnel : $A \in$ plus petit point fixe de T_P, $A \in T_P \uparrow \omega$; et procédural :

$A \in$ Succès (P), qui constitue la caractéristique de la programmation déclarative. Plus explicitement, les équivalences classiques [Llo. 18] :

$$\mathscr{T} \cup P \models A \quad \Leftrightarrow \quad A \in T_P \uparrow \omega \quad \Leftrightarrow \quad A \in SS(P)$$
$$\mathscr{T} \cup P^* \models \neg A \quad \Leftrightarrow \quad A \notin T_P \downarrow \omega \quad \Leftrightarrow \quad A \in FF(P),$$

rattachant ces divers aspects, ont pu être obtenues par étapes pour différentes situations de programmation avec contraintes, dans différents travaux [v.Em.Ll. 12] [J.L.M. 15] [Ja.St. 17] [Col. 7].

Dans « Constraint Logic Programming » [Ja.La. 13, Ja.La. 14], J. Jaffar et J. L. Lassez précisent complètement la forme générale d'un *cadre de contraintes* dans lequel ces équivalences (entre autres) restent vérifiées.

En quelques mots : tout élément de la structure $\mathscr{R}$, *espace de résolution* des contraintes, doit être caractérisable par un ensemble (éventuellement infini) de contraintes, c'est par exemple le cas de l'algèbre complète de Herbrand $\mathbb{H}^\omega$ avec l'égalité pour contrainte; de plus $\mathscr{R}$ doit avoir des solutions pour tout ensemble de contraintes dont chaque partie finie est résoluble (*solution-compacité*); et $\mathscr{T}$, axiomatique de $\mathscr{R}$, doit déduire exactement le comportement de $\mathscr{R}$ en regard de la résolution des contraintes finies (*satisfaction-complétude*).

Nous nous proposons dans ce travail de compléter ce bien-fondé de la sémantique déclarative de la Programmation Logique avec contraintes, en étudiant de plus près le rôle particulier de la sous-algèbre $\mathscr{R}_0$ des éléments effectifs dans une algèbre de contraintes $\mathscr{R}$. Plus précisément, $\mathscr{R}_0$ est l'algèbre des éléments définissables par contraintes *finies*, telle que par exemple la sous-algèbre $\mathbb{H}^r$ des arbres rationnels [Mah. 20, Cour. 10] dans le cas d'une algèbre complète de Herbrand $\mathbb{H}^\omega$.

Le rôle privilégié de cette sous-algèbre est évident au sens pratique, puisqu'elle est constituée des seuls éléments traitables procéduralement. L'agent procédural n'a connaissance de, et ne peut manipuler que ces éléments. Les autres éléments, inaccessibles effectivement, que le modèle théorique est pourtant conduit à considérer, ne risquent-ils pas de dénaturer l'implémentation de la résolution de contraintes? Les sémantiques déclaratives dans l'algèbre des arbres rationnels et dans l'algèbre des arbres infinis ne sont pas exactement les mêmes : par exemple, le plus grand point fixe de la fonction conséquences immédiates T_P n'est pas le même dans $\mathbb{H}^r$ ou dans $\mathbb{H}^\omega$ (*cf.* exemple 3.1).

En d'autres termes, la résolution des contraintes est-elle implémentée dans l'algèbre des arbres rationnels ou dans celle des arbres infinis? On sait heureusement que la résolution est la même dans ce cas [Mah. 20] : les deux algèbres sont logiquement équivalentes. Mais en est-il de même dans tous les cadres de contraintes remplissant les conditions générales proposées dans [Ja.La. 13] ?

Nous vérifions que ce n'est pas le cas (*cf.* partie 5), et nous donnons dans la partie 4 des conditions simples et naturelles à vérifier pratiquement, dans lesquelles on peut démontrer une Invariance de Résolution (IR) (*cf.* prop. 4.1; cor. 4.3), justifiant une implémentation de résolution de contraintes valable aussi bien dans $\mathscr{R}$ que dans $\mathscr{R}_0$.

Dans la partie numéro 2 nous présentons une partie significative des résultats de [Ja.La 13, Ja.La. 14], en introduisant rapidement les concepts généraux à la sémantique déclarative de la Programmation Logique avec Contraintes. Nous renvoyons au travail de J. Jaffar et J. L. Lassez pour plus de précision, d'illustrations détaillées, ou de compléments. Nous nous limitons au minimum de présentation générale : ainsi, par exemple, il serait souhaitable de se situer dans le cadre d'un langage à plusieurs sortes, certaines sortes étant des algèbres formelles de termes, pour correspondre à la réalité des situations implémentées. Ceci ne ferait qu'alourdir le texte et nous choisissons pour la clarté des concepts de présenter une seule sorte de structure $\mathscr{R}$, comme le font très vite J. Jaffar et J. L. Lassez [Ja.La. 13] [Ja.La. 14].

La partie 3 étudie le rôle privilégié de la sous-algèbre des éléments effectifs ainsi que les propriétés nécessaires à leur étude. La partie 4 montre les résultats d'Indépendance de Résolution dans l'une ou l'autre algèbre et donne des conditions suffisantes pour que leurs sémantiques déclaratives coïncident totalement. Enfin la partie 5 construit une situation de Programmation avec Contraintes au sens de [Ja.La. 13] dans laquelle on dispose d'algorithmes de résolutions différents selon que l'on considère les éléments effectifs ou le modèle théorique. Cet exemple rend nécessaire le renforcement des hypothèses d'un cadre de contraintes, comme il est fait dans la partie 4, si l'on souhaite l'Indépendance de Résolution.

2. SÉMANTIQUE DÉCLARATIVE DE LA PROGRAMMATION LOGIQUE AVEC CONTRAINTES

Cette partie présente les principaux concepts et résultats de [Ja.La. 13] [Ja.La. 14], en particulier les aspects logique, algébrique, fonctionnel, procédural, de la programmation logique avec contraintes, dans le cadre général d'une structure quelconque de résolution de contraintes.

Dans toute la suite, V, Σ, Π, désignent respectivement un ensemble de variables, de symboles fonctionnels et de symboles relationnels; $\mathscr{R}$ est une interprétation du langage du premier ordre associé à (Π, Σ).

Plus précisément, $\mathscr{R}$ est la donnée d'un ensemble non vide $|\mathscr{R}|$, base de réalisation et, pour chaque élément f de Σ (respectivement p de Π), d'arité n, d'une application $\mathscr{R}(f) : |\mathscr{R}|^n \to |\mathscr{R}|$ [respectivement d'une relation $\mathscr{R}(p)$, partie de $|\mathscr{R}|^n$]. On considèrera toujours dans Π le symbole d'égalité $=$ avec son interprétation standard.

On désignera par $\tilde{x}$ une suite finie $x_1, \ldots, x_n$ de variables et par $\tilde{t}$ une suite finie $t_1, \ldots, t_m$ de termes du langage associé à (Π, Σ). De même $\tilde{A}$ désignera une suite finie $A_{1,}, \ldots, A_k$.

Si φ est une formule du langage (Π, Σ), $\tilde{\exists}\varphi$ (resp. $\tilde{\forall}\varphi$) désigne la clôture existentielle (resp. universelle) de φ, c'est-à-dire : $\exists x_1 \exists x_2 \ldots \exists x_n \varphi$ (resp. $\forall x_1 \forall x_2 \ldots \forall x_n \varphi$), si $x_1, x_2, \ldots, x_n$ sont les variables libres qui figurent dans φ.

Une *$\mathscr{R}$-assignation* est une fonction $\theta : V \to |\mathscr{R}|$. Si t (respectivement $\tilde{t}$) désigne un terme (une suite finie de termes), on note $\theta t\,(\theta\tilde{t})$ l'élément de $|\mathscr{R}|$ (de $|\mathscr{R}|^n$) correspondant, par l'assignation θ. On dit qu'une formule atomique $A := p(\tilde{t})$ est satisfaite par θ si $\theta\tilde{t} \in \mathscr{R}(p)$; le groupement $p(\theta\tilde{t})$ est aussi noté $A\theta$.

On appelle (Π, Σ)-*contrainte*, ou simplement contrainte lorsqu'il n'y a pas d'ambiguïté, tout ensemble c de formules atomiques. Pour une contrainte finie c_0 (cardinal de c_0 fini), on assimilera le plus souvent c_0 à la formule conjonction de ses éléments.

On dit qu'une $\mathscr{R}$-assignation θ satisfait une contrainte c si chaque élément de c est satisfait par θ; c est dite *résoluble* (dans $\mathscr{R}$) si une telle assignation existe. En particulier, dire qu'une contrainte c_0 finie est résoluble s'exprime aussi par :

$$\mathscr{R} \models \tilde{\exists} c_0.$$

Un *programme logique de contraintes* dans $\mathscr{R}$, soit P, qu'on appellera un *$\mathscr{R}$-programme*, est défini par la donnée d'un alphabet de symboles relationnels Π_P disjoint de Π, et d'un ensemble fini de règles de contraintes, les *clauses* de P, de la forme :

$$A \leftarrow (c \,\square\, B_1, B_2, \ldots, B_n),$$

avec c une (Π, Σ)-contrainte finie (éventuellement vide), $A, B_1, \ldots, B_n$ des atomes du langage (Π_P, Σ) (éventuellement $n=0$).

Suivant l'usage un *but* est de la forme : $G := (c \square B_1, \ldots, B_n)$; il est dit atomique si $n=1$. Pour un but atomique $G = c \square A$, on note :

$$[G] = \{ A\theta \mid \theta : \mathscr{R}\text{-assignation satisfaisant } c \}.$$

Une *variante d'une clause* C est une clause C' qui ne diffère de C que par un éventuel renommage des variables.

A tout $\mathscr{R}$-programme P est associée l'habituelle axiomatique déclarative, qu'on notera aussi P, conjonction des formules $\tilde{\forall}\, \Phi_p$ du langage $(\Pi \cup \Pi_P, \Sigma)$:

$$\Phi_p := \left(\left\{ \begin{array}{c} (\tilde{x} = \tilde{t}_1 \wedge c_1 \wedge \tilde{A}_1) \vee \\ (\tilde{x} = \tilde{t}_2 \wedge c_2 \wedge \tilde{A}_2) \vee \\ \vdots \\ (\tilde{x} = \tilde{t}_m \wedge c_m \wedge \tilde{A}_m) \end{array} \right\} \Rightarrow p(\tilde{x}) \right),$$

correspondant à la collection des clauses de P ayant p en tête :

$$\begin{array}{c} p(\tilde{t}_1) \leftarrow (c_1 \square \tilde{A}_1) \\ p(\tilde{t}_2) \leftarrow (c_2 \square \tilde{A}_2) \\ \vdots \\ p(\tilde{t}_m) \leftarrow (c_m \square \tilde{A}_m) \end{array}$$

écrites sans variable commune entre elles.

On utilisera aussi la complétion de Clark P^* [Cla. 6, Llo. 18, Ja.La. 13], dans laquelle le connecteur d'implication de Φ_p est remplacé par le connecteur d'équivalence $\Leftrightarrow$.

Aspect algébrique associé à un programme

La *$\mathscr{R}$-base* associée au $\mathscr{R}$-programme P, analogue à la base de Herbrand associée à un programme logique classique, est définie par :

$$\mathbb{B}_P = \{ p(\theta\tilde{x}) \mid p \in \Pi_P, \theta \text{ une } \mathscr{R}\text{-assignation} \}.$$

On la notera aussi $\mathbb{B}$ s'il n'y a pas d'ambiguïté.

Il est évident et essentiel, comme en programmation logique classique, de noter que toute interprétation du langage Π_p dans $\mathscr{R}$ peut être assimilée à une partie de $\mathbb{B}_P$, munissant ainsi ces interprétations de la structure de treillis des parties de $\mathbb{B}_P$.

On notera $\mathscr{T} \models_{\mathscr{R}} \varphi$ pour exprimer que toute interprétation, extension de $\mathscr{R}$, qui satisfait les formules de $\mathscr{T}$, satisfait aussi φ.

Fonctionnelle d'un programme

On considère la *fonction conséquences immédiates* $T_{\mathscr{R},P}$ de Kowalski, Van Emden (v.Em.Ko. 11, Llo. 18] associée au $\mathscr{R}$-programme P, qu'on notera aussi T_P ou $T_{\mathscr{R}}$ ou T lorsqu'il n'y a pas d'ambiguïté. T est définie sur et à valeurs dans les parties de la $\mathscr{R}$-base $\mathbb{B}$ par :

$D \in T(I)$ si et seulement si existent une clause $C := A \leftarrow (c \,\square\, B_1, \ldots, B_k)$ de P, et une $\mathscr{R}$-assignation θ satisfaisant c, telles que $A\theta = D$ et $B_i\theta \in I$ pour chaque $i \leqq k$.

On dira que $D \in T(I)$ *par la clause* C *et l'assignation* θ. On vérifie le rapport habituel entre la fonction conséquences immédiates et les modèles de P extensions de $\mathscr{R}$:

PROPOSITION 2.1 : *Soient* I *une partie de* $\mathbb{B}$, A *une formule atomique du langage* (Π_P, Σ) *et* θ *une* $\mathscr{R}$-*assignation. Alors :*

1. $T(I) \subset I$ *si et seulement si* I *modèle de* P.
2. $T(I) = I$ *si et seulement si* I *modèle de* P^*.
3. $P \models_{\mathscr{R}} A\theta$ *si et seulement si* $A\ \theta \in lfp\ (T)$.
4. $P^* \models_{\mathscr{R}} \neg A\theta$ *si et seulement si* $A\theta \notin gfp(T)$

où $lfp(T)$ *et* $gfp(T)$ *désignent respectivement le plus petit et le plus grand points fixes de* T.

PROPOSITION 2.2 : *T est continue.*

Aspect procédural d'un programme

Un *pas de* ($\mathscr{R}$, P)-*dérivation* depuis le but
$G := (c \,\square\, q_1(\tilde{t}_1), \ldots, q_n(\tilde{t}_n))$ fournit un but,
$G' := (c' \,\square\, \tilde{B}_1, \ldots, \tilde{B}_n)$ s'il existe n variantes de clauses de P, de la forme :

$$q_i(\tilde{u}_i) \leftarrow (c_i \,\square\, \tilde{B}_i)$$

sans variable commune ni entre elles, ni avec G, de sorte que

$$c' := c \cup c_1 \cup \ldots \cup c_n \cup \{\tilde{t}_1 = \tilde{u}_1, \ldots, \tilde{t}_n = \tilde{u}_n\}$$

soit résoluble dans $\mathscr{R}$.

– un *succès* sur le but $(c \,\square\, A_1, \ldots, A_n)$ est une suite finie de pas de ($\mathscr{R}$, P)-dérivation se terminant par un but $(c' \,\square\,)$: en particulier c' est résoluble.

– Toute suite de pas de ($\mathscr{R}$, P)-dérivation maximale (qui ne peut se prolonger) et qui n'est pas un succès, est un *chemin d'échec fini*.

– $SS(P)$ est l'ensemble des buts atomiques $c \square A$ sur lesquels existe un succès se terminant par $(c' \square\ )$, avec : $[c' \square\ A] = [c \square A]$.

– $FF(P)$ est l'ensemble des buts atomiques pour lesquels toute dérivation est un chemin d'échec fini.

La question importante qui se pose alors est : comment choisir les structures $\mathscr{R}$ dans lesquelles s'exprimeront les contraintes, de manière à conserver les équivalences entre les sémantiques logique, algébrique, fonctionnelle, procédurale [J.L.M. 16]. J. Jaffar et J. L. Lassez ont totalement répondu à cette question en introduisant la notion de *cadre de contraintes* que nous présentons maintenant.

Fixons d'abord quelques notations. Dans une contrainte c peuvent figurer des variables distinctes, éventuellement en quantité infinie. Il sera parfois souhaitable de distinguer plus particulièrement certaines de ces variables $x_1, x_2 \ldots x_n (\tilde{x})$. On considèrera dans ce cas $c(\tilde{x})$ et on dira que $\tilde{d} \in |\mathscr{R}|^n$ est une solution de $c(\tilde{x})$ s'il existe une $\mathscr{R}$-assignation θ satisfaisant c et telle que $\tilde{d} = \theta\, \tilde{x}$.

On notera $\mathrm{Sol}(c(\tilde{x}))$ la partie de $|\mathscr{R}|^n$ constituée par les solutions de $c(\tilde{x})$.

Si $\tilde{d} \in \mathrm{Sol}(c(\tilde{x}))$, on pourra écrire par abus de notation que $c(\tilde{d})$ est résoluble dans $\mathscr{R}$; si de plus c est finie, on pourra même écrire $\mathscr{R} \models \tilde{\exists}\, c(\tilde{d})$, (clôture existentielle de c où les variables x_i ont été assignées en d_i).

DÉFINITION 2.3 : $\mathscr{R}$ est une *structure de contraintes* si :

(SC_1) : tout élément d de $|\mathscr{R}|$ est définissable par une contrainte au sens : il existe une contrainte c et une variable x telle que

$$\mathrm{Sol}(c(x)) = \{d\}$$

(SC_2) : $\mathscr{R}$ est *solution-compacte* au sens : toute contrainte c insoluble possède une partie finie (une sous-contrainte finie) c_0 insoluble.

Le premier exemple non trivial de structure de contraintes est l'algèbre complète de Herbrand $\mathbb{H}^{\omega}$ [Llo 18], algèbre des arbres finis ou infinis sur l'alphabet Σ, avec $\Pi = \{ = \}$, base théorique de Prolog II [Col. 7, Ja.St. 17, Mah. 20].

Une *axiomatique satisfaction complète* de $\mathscr{R}$ est une classe $\mathscr{T}$ de formules du langage (Π, Σ), peut-être infinie, mais au moins récursivement présentable, dont $\mathscr{R}$ soit modèle, et qui soit complète pour la satisfaisabilité, au sens : pour chaque contrainte finie c_0, on a :

$$\text{soit } \mathscr{T} \models \tilde{\exists}\, c_0, \text{ soit } \mathscr{T} \models \neg\, \tilde{\exists}\, c_0.$$

Il résulte évidemment de cette définition qu'une axiomatique satisfaction-complète $\mathcal{T}$ de $\mathcal{R}$ reflète exactement le comportement de $\mathcal{R}$ pour la satisfaisabilité des contraintes finies :

$$\mathcal{T} \models \tilde{\exists} c_0 \quad \text{si et seulement si} \quad \mathcal{R} \models \tilde{\exists} c_0$$
$$\mathcal{T} \models \neg \tilde{\exists} c_0 \quad \text{si et seulement si} \quad \mathcal{R} \models \neg \tilde{\exists} c_0.$$

Ce sera en particulier le cas si $\mathcal{T}$ est une théorie complète et $\mathcal{R}$ modèle de $\mathcal{T}$; de même, si $\mathcal{R}$ est décidable, elle possède une axiomatique satisfaction-complète : c'est ce qui se passe dans la pratique (cas, par exemple, de l'axiomatique complète des arbres infinis [Mah 20]).

DÉFINITION 2.4 : On appelle *cadre de contraintes* un couple $(\mathcal{R}, \mathcal{T})$ dans lequel $\mathcal{R}$ est une structure de contraintes et $\mathcal{T}$ une axiomatique satisfaction-complète de $\mathcal{R}$.

Les résultats généraux de sémantique déclarative des programmations avec contraintes ci-dessous sont obtenus dans [Ja.La. 13] [Ja.La. 14].

THÉORÈME 2.5 (J. Jaffar, J. L. Lassez) : *$(\mathcal{R}, \mathcal{T})$ désignant un cadre de contraintes, pour chaque $\mathcal{R}$-programme P, on a les équivalences suivantes :*

– *d'une part, pour tout* $D \in \mathbb{B}$:

$$(a_1)\ D \in T \uparrow \omega \textit{ si et seulement si } D \in [SS(P)],$$
$$(a_2)\ D \notin T \downarrow \omega \textit{ si et seulement si } D \in [FF(P)],$$

où $[X] = \bigcup_{x \in X} [x]$ (cf. *notation p.* 4)

– *d'autre part, pour tout but atomique G :*

$$(b_1)\ \mathcal{T} \cup P \models G \textit{ si et seulement si } G \in SS(P)$$
$$(b_2)\ \mathcal{T} \cup P^* \models \neg G \textit{ si et seulement si } G \in FF(P).$$

Signalons que ces résultats sont démontrés sous une hypothèse plus faible que la solution-compacité de la structure $\mathcal{R}$:

On dit que $\mathcal{R}$ est *solution-compacte faible* [Ja.La. 14] si pour toute contrainte finie c_0 existe une famille de contraintes finies $(c_j)_{j \in J}$ vérifiant :

$$\overline{\mathrm{Sol}(c_0(\tilde{x}))} = \bigcup_{j \in J} \mathrm{Sol}(c_j(\tilde{x})).$$

On vérifie aisément qu'une structure de contraintes solution-compacte est solution-compacte faible, la réciproque n'étant pas vraie ainsi que le montre l'exemple de l'algèbre de Herbrand $\mathbb{H}$ avec $\Pi = \{ = \}$, ou celui de l'algèbre complète de Herbrand $\mathbb{H}^{\omega}$ avec $\Pi = \{ =, \neq \}$.

Les cadres de contraintes (faibles) utilisés par les différents Prolog et CLP($\mathscr{R}$) sont :

$\mathbb{H}$, algèbre de Herbrand sur Σ, et $\Pi = \{ = \}$

$\mathbb{H}^{\omega}$, algèbre complète de Herbrand sur Σ, et $\Pi = \{ =, \neq \}$

$\mathbb{R}$, corps des réels, $\Sigma = \{ +, 0, 1, \ldots \}$, et $\Pi = \{ =, \neq, \leqq, \ldots \}$

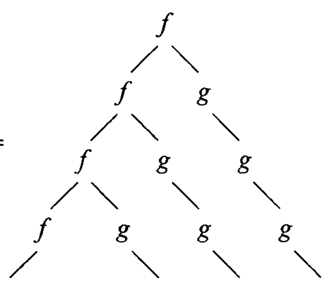
$\mathbb{R}$, corps des réels, $\Sigma = \{ +, *, 0, 1, \ldots \}$, et $\Pi = \{ =, \neq, \leqq, \ldots \}$.

Pour les précisions et exemples, on peut se reporter à [Ja.La. 13].

3. SOUS-ALGÈBRE EFFECTIVE

Le traitement procédural est évidemment limité aux contraintes finies. En particulier, les données elles-mêmes (élément de $|\mathscr{R}|$) ne peuvent être manipulées et traitées que sous une description finie. Par exemple, Prolog II traite les arbres rationnels, même si son modèle théorique peut se placer dans les arbres infinis, la constante

$r :=$



se décrivant par la contrainte finie :

$$c(x) := \{ x = f(x, y), y = g(y) \}.$$

— Remarquons que le modèle théorique dans lequel s'obtiennent les principaux résultats de sémantique déclarative [v.Em.Ll. 12, J.L.M. 15, Ja.St. 17, Col. 7] peut aussi bien être l'algèbre des arbres infinis $\mathbb{H}^{\omega}$, que l'algèbre des arbres rationnels $\mathbb{H}^{r}$, les deux étant logiquement équivalentes [Mah. 20].

– Toutefois, les résultats de nature déclarative n'entrent pas tous dans le cadre de cette remarque. Ainsi, considérons le programme P suivant :

Exemple 3.1 :

$$P : \begin{cases} C(o) \leftarrow S(x) \\ S(o.y) \leftarrow R(o.y) \\ R(x.s(x).y) \leftarrow R(s(x).y) \end{cases}$$

Les plus grands points fixes $gfp(T)$ des fonctions conséquences immédiates T, relativement à l'algèbre des arbres rationnels et à l'algèbre des arbres infinis, sont distincts. Si $\mathbb{B}^r$ et $\mathbb{B}$ désignent les bases de Herbrand relatives à $\mathbb{H}^r$ et $\mathbb{H}^\omega$, et si T^r et T sont les fonctions conséquences immédiates associées à P correspondantes, il est aisé de constater que l'on a :

$$C(o) \notin gfp(T^r) \quad \text{d'une part et}$$

$$C\,o) \in gfp(T) \cap \mathbb{B}^r \quad \text{d'autre part,}$$

la signification en sémantique algébrique étant que le complété de Clark P^* de P déduit $\neg\, C(o)$ dans $\mathbb{H}^r$ alors qu'il ne le déduit pas dans $\mathbb{H}^\omega$:

$$P^* \models_{\mathbb{H}^r} \neg\, C(o) \quad \text{alors que} \quad P^* \not\models_{\mathbb{H}^\omega} \neg\, C(o).$$

Du point de vue d'une éventuelle implémentation de résolution des contraintes dans $\mathbb{H}^r$ ou dans $\mathbb{H}^\omega$, cette différence n'a pas grande importance, car elle reste toute théorique. En effet, la réalité procédurale ne conduit à envisager que les itérations finies des deux fonctions conséquences immédiates, plus précisément, dans l'exemple précédent, les $T \downarrow \omega$. Et ici il y a coïncidence, au sens : $C(o) \in T^r \downarrow \omega$ et $C(o) \in (T \downarrow \omega) \cap \mathbb{B}^r$. (Rappelons que l'on a toujours $T \downarrow \omega = gfp(T)$ dans l'algèbre complète de Herbrand $\mathbb{H}^\omega$ [Llo 18]).

En d'autres termes, une implémentation de résolution dans $\mathbb{H}^r$ ou dans $\mathbb{H}^\omega$ conduit à la « même réponse », un calcul infini dans le cas envisagé.

En résumé, d'un point de vue pratique, on est amené à considérer :

– d'une part, les éléments réellement traitables, ceux de $\mathbb{H}^r$, et les littéraux correspondants, ceux de $\mathbb{B}^r$;

– d'autre part, les résultats finîment obtenus, succès (les littéraux de $T \uparrow \omega$) ou échecs finis (ceux de $\overline{T \downarrow \omega}$).

Nous savons qu'il est équivalent d'envisager la résolution des contraintes dans le cadre théorique $\mathbb{H}^\omega$ ou dans le cadre effectif $\mathbb{H}^r$, lorsque le but proposé et le calcul sont finis [Mah. 20].

On peut dégager les relations essentielles garantissant l'Indépendance de Résolution sous la forme suivante :

$$(IR)\begin{cases} T^r\uparrow\omega=(T\uparrow\omega)\cap\mathbb{B}^r \\ T^r\downarrow\omega=(T\downarrow\omega)\cap\mathbb{B}^r \end{cases}$$

La question qui se pose intuitivement, dans une structure de contraintes quelconque $\mathcal{R}$, peut être formulée ainsi : pour un but dans lequel les paramètres (les constantes) sont finiment décrits, la résolution des contraintes dans la structure $\mathcal{R}$ ou dans sa partie « effective » $\mathcal{R}_0$ conduit-elle au même résultat ?

Nous allons maintenant préciser cette question. Dans toute la suite, $\mathcal{R}$ est une structure de contraintes.

DÉFINITION 3.2 : Un élément de $|\mathcal{R}|$ est dit *effectif* si une contrainte finie suffit à la définir (*cf.* déf. 2.3).

PROPOSITION. DÉFINITION 3.3 : L'ensemble des éléments effectifs de $|\mathcal{R}|$ constitue une sous-algèbre $\mathcal{R}_0$ de $\mathcal{R}$.

La vérification est immédiate. On appellera *sous-algèbre effective* de $\mathcal{R}$ cette sous-algèbre $\mathcal{R}_0$.

Exemples 3.4 (Support de Prolog, II, III, CLP ($\mathcal{R}$))

$$\mathcal{R}=\langle\mathbb{H};\, a, f, g;\, =\rangle,\qquad \mathcal{R}_0=\mathcal{R}$$

$$\mathcal{R}=\langle\mathbb{H}^\omega;\, a, f, g;\, =, \neq\rangle,\qquad \mathcal{R}_0=\langle\mathbb{H}^r;\, a, f, g;\, =, \neq\rangle$$

$$\mathcal{R}=\langle\mathbb{R};\, 0, 1, \ldots, +;\, =, \leqq, \ldots\rangle,\qquad \mathcal{R}_0=\langle\mathbb{Q};\, 0, 1, \ldots, +;\, =, \leqq\ldots\rangle$$

$$\begin{cases} \mathcal{R}=\langle\mathbb{R};\, 0, 1, \ldots, +, *;\, =, \leqq, \ldots\rangle, \\ \mathcal{R}_0=\langle\mathbb{A};\, 0, 1, \ldots, +, *;\, =, \leqq\ldots\rangle \end{cases}$$

où $\mathbb{H}$, $\mathbb{H}^r$, $\mathbb{H}^\omega$ désignent des algèbres de Herbrand classique, d'arbres rationnels, d'arbres infinis, $\mathbb{R}$, $\mathbb{Q}$, $\mathbb{A}$ les corps usuels des nombres réels, rationnels, algébriques. En général $\mathcal{R}_0$ n'est pas une structure de contraintes.

Les notions de $\mathcal{R}$ et de $\mathcal{R}_0$-programmes coïncident évidemment (c'est plus précisément de (Π, Σ)-programme qu'il faudrait parler).

Soit P un tel programme; nous noterons respectivement $\mathbb{B}_0$ et T_0 la $\mathcal{R}_0$-base et la $\mathcal{R}_0$-fonction conséquences immédiates associées à P. Nous supposerons toujours que les clauses C de P sont sous la forme :

$$p(\tilde{x})\leftarrow(c\,\square\, q_1(\tilde{y}_1), q_2(\tilde{y}_2), \ldots, q_k(\tilde{y}_k)),$$

les suites de variables $\tilde{x}, \tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_k$ n'ayant aucune variable commune deux à deux : il s'agit d'une simple commodité d'écriture, ne diminuant en rien la généralité puisque le symbole d'égalité fait partie des contraintes.

Rappelons que si I est une partie de $\mathbb{B}_0$, dire que $p(\tilde{d}) \in T_0(I)$ par la clause C de P et l'assignation θ signifie

1. θ, à valeurs dans $|\mathscr{R}_0|$, satisfait c,
2. C, de la forme précédente, est telle que $\theta\tilde{x}=\tilde{d}$ et $q_i(\theta\tilde{y}_i) \in I$ pour $i \leq k$.

Dans la suite, on notera $\mathrm{Sol}^p(c(\tilde{x}))$, où p appartient à Π_P et c désigne une contrainte finie, la partie de $\mathbb{B}$ constituée par les $p(\tilde{d})$ pour $\tilde{d} \in \mathrm{Sol}(c(\tilde{x}))$. Les $\mathrm{Sol}^p(c(\tilde{x}))$ seront appelées *parties contraintes* de $\mathbb{B}$.

DÉFINITION 3.5 : *Une partie I de $\mathbb{B}$ est dite* effective *si, pour chaque élément $G=p(\tilde{d})$ de I, existe une contrainte finie $c_0(\tilde{x})$ satisfaisant :*

$$G \in \mathrm{Sol}^p(c_0(\tilde{x})) \subset I,$$

c'est-à-dire : $c_0(\tilde{d})$ est résoluble (dans $\mathscr{R}$) et, si $c_0(\tilde{e})$ est résoluble (dans $\mathscr{R}$), $p(\tilde{e}) \in I$.

Remarques :

– Les parties effectives de $\mathbb{B}$ sont les parties finiment recouvrables introduites par J. Jaffar et P. Stuckey dans [Ja.St. 17].

– L'hypothèse de solution-compacité faible se traduit par : le complémentaire de toute partie contrainte de $\mathbb{B}$ est effectif.

– Toute partie de $\mathbb{B}_0$ est effective puisque, si $G=p(\tilde{d})$ est élément de $\mathbb{B}_0$, il existe par définition une contrainte finie $c_0(\tilde{x})$ telle que $\{d\}=\mathrm{Sol}(c_0(\tilde{x}))$.

– $\mathscr{R}_0$ est une structure solution-compacte faible.

– Les parties effectives sont des ouverts pour la topologie de $\mathbb{B}$ engendrée par les parties contraintes $\mathrm{Sol}^p(c(\tilde{x}))$.

La proposition suivante, utilisée dans le chapitre 4, montre en particulier que ce sont exactement les ouverts de cette topologie.

PROPOSITION 3.6 : *Les parties effectives de $\mathbb{B}$ sont stables par intersection finie, union quelconque et l'opération T de conséquences immédiates.*

Preuve : Les stabilités par intersection finie et réunion se vérifient aisément.

Soit I partie effective de $\mathbb{B}$ et considérons : $p(\tilde{d}) \in T(I)$ par la clause C et l'assignation θ. I étant effective, à chaque $q_i(\theta\tilde{y}_i)$ on peut associer une contrainte finie c_i telle que $q_i(\theta\tilde{y}_i) \in \mathrm{Sol}^{q_i}(c_i) \subset I$. On impose que les c_i n'aient aucune variable commune entre elles et avec c, et on constate que l'on a :

$$p(\tilde{d}) \in \mathrm{Sol}^p(c \wedge c_1 \wedge \ldots c_k) \subset T(I).$$

COROLLAIRE 3.7 : *Les parties $T \uparrow n$, $T \downarrow n$, $T_0 \uparrow n$, $T_0 \downarrow n$ sont effectives quel que soit l'entier n.*

Remarques :

– Les parties effectives ne sont pas stables par l'opération de complémentaire. On le constate dans $\mathbb{H}^\omega$ avec la contrainte d'égalité, pour une partie réduite à un arbre non rationnel. Cette partie n'est pas effective, alors que son complémentaire l'est.

– Les parties effectives sont exactement les réunions de parties contraintes.

Signalons un autre concept utile habituellement dans la sémantique déclarative de Programmation Logique avec Contraintes, la notion de parties *finiment présentables :* réunions finies de parties contraintes. Par exemple, $T \uparrow n$, $T \downarrow n$, $T_0 \uparrow n$, $T_0 \downarrow n$ sont finiment présentables [Ja.St. 17, Ja.La. 13]. Une propriété essentielle de ces parties tient à la proposition suivante :

PROPOSITION 3.8 : *Si I est finiment présentable, I et $\bar{I}$ sont effectifs.*

Preuve : Il suffit de montrer que, si :

$$p(\tilde{d}) \notin \bigcup_{1 \leq i \leq n} \mathrm{Sol}^p(c_i(\tilde{z}_i)),$$

existe $c_0(\tilde{x})$ finie telle que :

$$p(\tilde{d}) \in \mathrm{Sol}^p(c_0(\tilde{x})) \subset \overline{\bigcup_{1 \leq i \leq n} \mathrm{Sol}^p(c_i(\tilde{z}_i))}.$$

Considérons $c(\tilde{x})$ contrainte définissant $\tilde{d}$. Par hypothèse, chaque $c(\tilde{x}) \cup c_i(\tilde{z}_i)$ est insoluble, donc, par solution-compacité, existe une sous-contrainte finie c_0 de c telle que chaque $c_0(\tilde{x}) \cup c_i(\tilde{z}_i)$ soit insoluble. c_0 convient.

La réciproque n'est pas vraie, comme on le constate pour $\mathbb{H}^\omega$ avec la contrainte d'égalité, et $I = \{ f^n(a) \}_{n \in \omega}$.

L'importance de la propriété de présentation finie dans les démonstrations peut se pressentir en remarquant que l'hypothèse de solution-compacité faible évoquée dans la partie 2, suffisante à l'obtention des équivalences entre les diverses sémantiques, n'est rien d'autre que :

– le complémentaire de toute partie finiment présentable est effectif.

4. RÉSOLUTION DANS LE MODÈLE EFFECTIF

Lorsque les conditions d'Indépendance de Résolution, qui s'écrivent dans le cas général :

$$\text{(IR)} \qquad \begin{cases} T_0 \uparrow \omega = (T \uparrow \omega) \cap \mathbb{B}_0 \\ T_0 \downarrow \omega = (T \downarrow \omega) \cap \mathbb{B}_0 \end{cases}$$

ne sont pas satisfaites, il est nécessaire de choisir entre deux implémentations, l'une de résolution dans l'algèbre effective $\mathscr{R}_0$, l'autre de résolution dans l'algèbre théorique $\mathscr{R}$. Les paramètres, données effectives, étant eux de toute façon dans $\mathscr{R}_0$, il peut en effet exister un algorithme de résolution dans $\mathscr{R}$, par exemple si $\mathscr{R}$ est une structure décidable, comme les structures utilisées en pratique et citées en exemples (*cf.* 3.4).

Nous construisons dans la partie 5 un exemple pour lequel les deux algorithmes possibles de résolution ne conduisent pas au même résultat.

Dans cette partie, nous donnons une condition suffisante (et naturelle) pour que les condition (IR) soient remplies (corollaire 4.3). Cette condition, d'expression suffisamment simple, est vérifiée par les modèles utilisés pratiquement; il devient ainsi évident que ces modèles satisfont (IR). Nous donnons aussi une condition nécessaire et suffisante (corollaire 4.4). Dans toute la suite, $(\mathscr{R}, \mathscr{T})$ est un cadre de contraintes (*cf.* déf. 2.4) et P un $\mathscr{R}$-programme.

PROPOSITION 4.1 : *Si l'algèbre effective $\mathscr{R}_0$ est modèle de $\mathscr{T}$, on a, pour toute partie effective I de $\mathbb{B}$:*

$$T_0 (I \cap \mathbb{B}_0) = T(I) \cap \mathbb{B}_0.$$

Preuve : Il suffit de montrer que, si $p(\tilde{d}) \in T(I)$ par

$$C : p(\tilde{x}) \leftarrow (c \,\square\, q_1(\tilde{y}_1), \ldots, q_k(\tilde{y}_k))$$

et θ, avec $\tilde{d} \in |\mathscr{R}_0|^n$, alors $p(\tilde{d}) \in T_0(I \cap \mathbb{B}_0)$.

Choisissons $c_0(\tilde{x})$ contrainte finie caractérisant $\tilde{d}$ et, pour chaque $i = 1, \ldots, k$, $c_i(\tilde{y}_i)$ contrainte finie vérifiant

$$q_i(\theta \tilde{y}_i) \in \mathrm{Sol}^{q_i}(c_i(\tilde{y}_i)) \subset I,$$

de sorte que les autres variables apparaissant dans $c_0(\tilde{x})$, $c_1(\tilde{y}_1)$, $\ldots$, $c_k(\tilde{y}_k)$ soient toutes nouvelles, distinctes entre elles et distinctes des variables de c.

Sur ces nouvelles variables, θ peut être modifié en une solution de $\{c_0, c, c_1, \ldots, c_k\}$. Ainsi :

$$\mathscr{R} \models \tilde{\exists}(c_0 \wedge c \wedge c_1 \wedge \ldots \wedge c_k)$$

d'où :

$$\mathscr{T} \models \tilde{\exists}(c_0 \wedge c \wedge c_1 \wedge \ldots \wedge c_k)$$

et enfin, si $\mathscr{R}_0$ est modèle de $\mathscr{T}$:

$$\mathscr{R}_0 \models \tilde{\exists}(c_0 \wedge c \wedge c_1 \wedge \ldots \wedge c_k).$$

On dispose donc d'une assignation θ' à valeurs dans $|\mathscr{R}_0|$, avec

- $\theta' \tilde{x} = \tilde{d}$, puisque θ' satisfait c_0 qui caractérise $\tilde{d}$,
- θ' satisfait c,
- pour chaque $i = 1, \ldots, k, q_i(\theta' \tilde{y}_i) \in \mathrm{Sol}^{q_i}(c_i(\tilde{y}_i)) \subset I$, d'où le résultat.

PROPOSITION 4.2 : *Si $\mathscr{R}_0$ est modèle de $\mathscr{T}$ on a , pour chaque entier n :*

$$T_0 \uparrow n = (T \uparrow n) \cap \mathbb{B}_0$$
$$T_0 \downarrow n = (T \downarrow n) \cap \mathbb{B}_0$$

Immédiat par induction, compte rendu du résultat précédent et du corollaire 3.6.

COROLLAIRE 4.3 : *Si $\mathscr{R}_0$ est modèle de $\mathscr{T}$, les conditions d'Indépendance de Résolution sont remplies.*

Il s'agit d'une simple vérification.

La condition utilisée est que $\mathscr{R}_0$ soit (aussi) modèle de $\mathscr{T}$. Il est clair que $\mathbb{H}^r$ est modèle de l'axiomatique de $\mathbb{H}^\omega$ [Mah. 20]. De même, il est connu que le corps des nombres algébriques est modèle premier de la théorie des corps réels clos, axiomatique complète de $\langle \mathbb{R}; +, * \rangle$ [Ja.St. 17] [Mac. 19]. Dans ces cas, les résolutions dans le modèle effectif et dans le modèle théorique ne présentent pas de différence.

COROLLAIRE 4.4 : *Une condition nécessaire et suffisante pour que les conditions d'Indépendance de Résolution soient remplies est que $\mathscr{R}_0$ soit satisfaction-équivalent à $\mathscr{R}$, c'est-à-dire, pour toute contrainte finie c_0 :*

$$\mathscr{R} \models \tilde{\exists} c_0 \quad \textit{si et seulement si} \quad \mathscr{R}_0 \models \tilde{\exists} c_0.$$

La condition est évidemment nécessaire, en considérant un programme aussi simple que : $A \leftarrow (c_0 \square\ )$, si $\mathscr{R}_0$ n'est pas satisfaction-équivalent à $\mathscr{R}$. Elle est aussi suffisante : la démonstration de la proposition 4.1 n'a utilisé que cette propriété.

Nous pensons que la condition « $\mathscr{R}_0$ modèle de $\mathscr{T}$ », *a priori* plus forte que la dernière condition, est néanmoins plus intéressante à utiliser.

Nous verrons dans la dernière partie que les propriétés de cadre de contraintes proposées dans [Ja.St. 17] ne permettent pas de déduire (IR).

On peut se demander si le renforcement de la notion de cadre de contraintes par l'hypothèse « $\mathscr{R}_0$ modèle de $\mathscr{T}$ », qui force $\mathscr{R}_0$ à être un sous-cadre de contraintes faible de $(\mathscr{R}, \mathscr{T})$, ne permet pas d'obtenir plus sur la correspondance des résultats déclaratifs dans les deux modèles, « au-delà des dérivations finies ».

Il est par exemple clair que, si $\mathscr{R}_0$ est modèle de $\mathscr{T}$, l'égalité : $lfp(T_0) = lfp(T) \cap \mathbb{B}_0$ est vraie. En effet, par continuité de T et de T_0, on a $lfp(T_0) = T_0 \uparrow \omega$ et $lfp(T) = T \uparrow \omega$, d'où le résultat lorsque $\mathscr{R}_0$ est modèle de $\mathscr{T}$. Remarquons que sans cette hypothèse (donc dans le cas général d'un cadre de contraintes), le résultat n'est pas vrai (*cf.* exemple 5.5 de la dernière partie).

Par contre, l'hypothèse $\mathscr{R}_0$ modèle de $\mathscr{T}$ ne suffit pas à établir l'égalité $gfp(T_0) = gfp(T) \cap \mathbb{B}_0$, ainsi que le montre l'exemple 3.1.

On peut toutefois énoncer des conditions suffisantes au renforcement de la correspondance déclarative concernant $\mathscr{R}$ et $\mathscr{R}_0$.

LEMME 4.5 : *Si $\mathscr{R}_0$ est modèle de $\mathscr{T}$ et si I est une partie effective de $\mathbb{B}$, alors : si I est point fixe de T, $I \cap \mathbb{B}_0$ est point fixe de T_0.*

C'est un corollaire immédiat de la proposition 4.1.

PROPOSITION 4.6 : *Si $\mathscr{R}_0$ est modèle de $\mathscr{T}$ et si $gfp(T)$ est effectif, on a :*

$$gfp(T_0) = gfp(T) \cap \mathbb{B}_0.$$

En termes de dérivation, les atomes qui ne sont pas dans $gfp(T_0)$ sont alors exactement ceux en échec fini par instantiations « closes », à valeurs dans $\mathscr{R}$ à chaque pas (« ground finite failure set » [Ja.La. 14]).

Preuve : par le lemme 4.5, $gfp(T) \cap \mathbb{B}_0$ est un point fixe de T_0 donc inclus dans $gfp(T_0)$. L'inclusion inverse est tout aussi immédiate, puisqu'un point fixe de T_0 est inclus dans $gfp(T)$.

Notons que, dans le lemme 4.5 et dans la proposition 4.6, on ne peut éviter aucune des deux hypothèses :

– effectivité : *voir* exemple 3.1, avec $I = gfp(T) \cap \mathbb{B}^r$ (bien qu'on ait $\mathscr{R}_0$ modèle de $\mathscr{T}$).

– $\mathscr{R}_0$ modèle de $\mathscr{T}$: *voir* exemple 5.5 (où on aura effectivité).

5. UN CAS DE NON-INVARIANCE POUR LA RÉSOLUTION

Dans cette partie, nous construisons un cadre de contraintes $(\mathscr{R}, \mathscr{T})$ particulier pour lequel implémenter la résolution de contraintes dans $\mathscr{R}$ ou dans $\mathscr{R}_0$ ne conduirait pas au même résultat.

Il suffit pour cela qu'existe une contrainte *finie* c_0, résoluble dans $\mathscr{R}$ et non dans $\mathscr{R}_0$; plus explicitement c_0 ne possèdera de solutions que parmi les éléments définissables seulement par contraintes *infinies*.

Les conditions imposées à un cadre de contraintes sont assez fortes. Rappelons-les :

– (SC) : $\mathscr{R}$ est une structure de contraintes, soit :

(SC_1) : tout élément de $|\mathscr{R}|$ est définissable par contrainte.

(SC_2) : $\mathscr{R}$ est solution-compact.

– (ASC) : $\mathscr{T}$ est une axiomatique récursivement présentable et satisfaction-complète de $\mathscr{R}$.

La structure de contraintes que nous considèrerons sera *décidable* (c'est-à-dire $\mathscr{R} \models \Phi$ décidable pour les formules closes Φ du langage). La classe $\mathscr{T}$ des formules vraies dans $\mathscr{R}$ constituera évidemment ainsi une axiomatique de $\mathscr{R}$ satisfaisant (ASC).

Nous nous plaçons dans une algèbre $\mathfrak{U}^\omega$ de *mots infinis* sur un alphabet fini $\mathfrak{U} = \{a, b, c, \ldots\}$, ces algèbres étant bien étudiées en Théorie des Langages [Ni.Pe. 21, Ber. 2]. D'autre part, un tel modèle est suffisamment intuitif à manier : il s'agit d'un cas particulier d'arbres infinis $\mathbb{H}^\omega$ dans lequel $\mathbb{H}^r$ est l'ensemble *PER* des mots infinis ultimement périodiques sur l'alphabet $\mathfrak{U}$.

Plus précisément, on choisit $\Sigma = \{f_a(\ ) \mid a \in \mathfrak{U}\}$, f_a étant interprété par la concaténation à gauche par a : $f_a(s) = a.s$.

On note que, l'égalité faisant toujours partie des contraintes, (SC_1) est vérifié de fait. Dans cette situation :

$$\begin{cases} |\mathscr{R}| = \mathbb{H}^{\omega} = \mathfrak{U}^{\omega} \\ |\mathscr{R}_0| \supseteqq \mathbb{H}^{r} = PER. \end{cases}$$

On introduit enfin un prédicat unaire $K(\ )$ dans les contraintes : $\Pi = \{ =, K(\ ) \}$. On écrit encore K pour désigner la partie de $\mathfrak{U}^{\omega}$ interprétant ce symbole de prédicat : c'est le choix de cette partie K qui va conditionner l'exemple. Tout d'abord :

$$\begin{cases} K \neq \varnothing \quad \text{si et seulement si } \mathscr{R} \models \exists x\, K(x) \\ K \cap |\mathscr{R}_0| = \varnothing \quad \text{si et seulement si } \mathscr{R}_0 \not\models \exists x\, K(x), \end{cases}$$

une difficulté à contourner étant que $\mathscr{R}_0$ est déterminé par le choix de K.

Une partie K telle que $\varnothing \neq K \subset \overline{PER}$ conviendra si on a :

(*a*) $|\mathscr{R}_0| = PER$,

(*b*) $\mathscr{R}$ solution-compact,

(*c*) $\mathscr{R}$ décidable.

L'intérêt du modèle choisi est que les propriétés (*a*) et (*b*) y sont à peu près équivalentes à : K compact et sans point isolé pour la topologie d'espace ultramétrique usuelle de $\mathfrak{U}^{\omega}$ [Ni.Pe 21, Cour. 10].

Plus précisément, on obtient les caractérisations suivantes, où $\mathfrak{U}^*$ est l'ensemble des mots finis sur l'alphabet $\mathfrak{U}$:

PROPOSITION 5.1 : *$PER \subseteqq |\mathscr{R}_0|$ si et seulement si il existe $u_1, \ldots, u_n$ dans $\mathfrak{U}^*$ pour lesquels un, et un seul, élément s de $\mathfrak{U}^{\omega}$ vérifie :*

$$u_i . s \in K \qquad \textit{pour} \quad i = 1, \ldots, n.$$

PROPOSITION 5.2 : *$\mathscr{R}$ est solution-compact si et seulement si chaque mot s de $\bar{K}$ possède un préfixe $u \in \mathfrak{U}^*$ qui ne soit préfixe d'aucun élément de K.*

Pour la preuve de ces deux propositions, nous renvoyons à [Bl.Blg. 5].

Dans chacune, la démonstration de l'une des implications est facile (condition suffisante pour 5.1, nécessaire pour 5.2).

L'autre implication se démontre en utilisant la propriété de confluence de la relation définie sur $v(\mathscr{E})$ [$\mathscr{E}$ système consistant d'équations, $v(\mathscr{E})$ l'ensemble des variables figurant dans $\mathscr{E}$] par :

« $x \mapsto y$ si et seulement si existe $u \in \mathfrak{U}^*$ tel que l'équation $x = u . y$ s'obtienne par simplification finie de $\mathscr{E}$ ».

Remarques 5.3 :

– La condition de la proposition 5.2 traduit exactement la compacité de K dans l'espace ultramétrique $\mathfrak{U}^\omega$.

– Dans le cas $n=1$, la condition de la proposition 5.1 traduit l'existence d'un point isolé de K.

De bonnes propriétés pour que K, non vide, convienne, semblent ainsi réalisées par :

$$(1) \qquad K \cap PER = \varnothing$$

$$(2) \quad \forall u_1 \ldots u_n \in \mathfrak{U}^* \, \forall s \in \mathfrak{U}^\omega \{ \bigwedge_{i \leqq n} (u_i . s \in K) \to \exists s' \in \mathfrak{U}^\omega (s' \neq s, \bigwedge_{i \leqq n} (u_i . s' \in K)) \}$$

$$(3) \qquad \forall s \in \bar{K} \, \exists u \in \text{Préfixe}(s) \, \forall s' \in \mathfrak{U}^\omega \{ u \in \text{Préfixe}(s') \to s' \in \bar{K} \},$$

ce qui écarte la possibilité d'une partie K dégénérée :

K fini ne satisferait pas (2)

$\bar{K} \cap \overline{PER}$ fini ne satisferait pas (3),

et fait apparaître l'ensemble des mots infinis sans carré sur $\mathfrak{U}$ comme un « bon candidat » pour K. Manifestement, cet ensemble satisfait les conditions (1) et (3). De plus, J. Berstel signale que la condition (2) doit aussi être vérifiée par cet ensemble. Le théorème de Shelton-Soni le montre pour $n=1$ [Ber. 3].

D'autres ensembles de mots infinis sans répétition sur $\mathfrak{U}$ sont de « bons candidats » du même type, par exemple les mots sans chevauchement [Re.Sa 22].

Dans l'un ou l'autre de ces cas, il faudrait vérifier que la condition (2) est satisfaite.

On peut éviter cette vérification en aménageant l'exemple simple suivant :

Choisissons un mot infini sans carré α, constructible, sur $\mathfrak{U}_0 = \{a, b, c\}$, par exemple le mot d'Aršon [Ber. 2].

Soit $\mathfrak{U} = \{a, b, c, d\}$ et K la partie de $\mathfrak{U}^\omega$ constituée des mots sans carré qui donnent α par effacement de la lettre « d ». Les conditions (1), (2) et (3) précédentes sont alors trivialement remplies.

Pour terminer, il reste à vérifier que la structure $\mathcal{R}$ ainsi construite est décidable. On le montre en interprétant la structure $\mathcal{R}$ dans une structure $\hat{\mathcal{R}}$, de langage $\hat{\mathfrak{L}}$, pour laquelle on peut montrer l'élimination de quantifications. $\hat{\mathfrak{L}}$, non égalitaire, utilise une infinité de prédicats $C^{u,v}(\ ,\ )$, $K_v^u(\ )$, $P^{u,v}(\ )$ dans lesquels u, v désignent des mots (finis) sur l'alphabet $\mathfrak{U}$, s'interprétant par :

$$C^{u,v}(x, y) := \exists z (x = u.z \wedge y = v.z)$$
$$K_v^u(x) := \exists z (x = u.z \wedge K(v.z))$$
$$P^{u,v}(x) := \exists z (x = uz \wedge z = vz)$$

PROPOSITION 5.4 : *La structure $\hat{\mathscr{R}}$ permet l'élimination de quantificateurs.*

Cette proposition provient d'un résultat plus général de décidabilité dans les algèbres de mots infinis [Bl.Blg. 5].

Une analyse plus fine des conditions suffisantes à construire un contre-modèle permet de ne pas faire appel aux mots sans carré. Nous renvoyons à [Bl.Blg. 5], pour plus de détails sur ce point.

Dans un tel cadre de contraintes, on écrit très facilement des $\mathscr{R}$-programmes qui ne peuvent que poser des problèmes de sémantique déclarative :

Exemple 5.5 : $P := p(x) \leftarrow (K(y) \square\ )$ vérifie :

$$\begin{cases} T\uparrow\omega = lfp(T) = gfp(T) = T\downarrow\omega = \mathbb{B} \\ T_0\uparrow\omega = lfp(T_0) = gfp(T_0) = T_0\downarrow\omega = \varnothing. \end{cases}$$

En conclusion, « $\mathscr{R}_0$ modèle de $\mathscr{T}$ » semble une condition facilement utilisable et réalisée dans la pratique.

Nous pensons qu'il serait naturel de l'intégrer à la définition même de cadre de contraintes.

Évidemment, cette condition n'est pas nécessaire à l'Invariance de Résolution, mais nous doutons de voir apparaître l' « usage implémenté » d'une structure de contraintes ne la satisfaisant pas.

REMERCIEMENTS

Nous tenons à remercier le rapporteur, dont les observations nous ont été particulièrement utiles.

BIBLIOGRAPHIE

[Ar.Ni. 1] A. ARNOLD et M. NIVAT, The metric space of infinite trees. Algebraic and topological properties, *Annales Societatis Mathematicae Polonae*, Series IV : *Fund. Inform.*, 1980, *III*, 4, p. 445-476.

[Ber. 2] J. BERSTEL, Mots sans carré et morphismes itérés, *Discrete Math.*, 1980, *29*, p. 235-244.

[Ber. 3] J. BERSTEL, Some recent results on squarefree words, Conférence au « Symposium for Theoretical Aspect of Computer Science », (STACS'84), Paris, avril 1984.

[B.B.C. 4] G. BLANC, N. BLEUZEN-GUERNALEC et S. COUPET, Kleene Functor and Logic Programming, Laboratoire de Mathématiques de Marseille, 1989.

[Bl.Blg. 5] G. BLANC et N. BLEUZEN-GUERNALEC, Topologie et Décidabilité dans les Algèbres de mots infinis, Rapport technique, Dept. de Math., Marseille-Luminy, 1990.

[Cla. 6] K. L. CLARK, Negation as Failure, in Logic and Database, H. GALLAIRE et J. MINKER éd., Plenum Press, New York, 1978, p. 293-322.

[Col. 7] A. COLMERAUER, Prolog and Infinite Trees, *in* Logic Programming, K. L. CLARK et S. A. TARNLUND éd., Academic Press, New York, 1982, p. 231-251.

[Col. 8] A. COLMERAUER, Une introduction à Prolog III, Note du G.I.A., Université d'Aix-Marseille-II, 1989.

[Coup. 9] S. COUPET-GRIMAL, Deux arguments pour les arbres infinis en Prolog, Thèse de Doctorat de l'Université d'Aix-Marseille-II, 1988.

[Cour. 10] B. COURCELLE, Fundamental properties of infinite Trees, *Theoret. Comput. Sci.*, 1983, *25*, (2), p. 95-169.

[v.Em.Ko. 11] M. H. VAN EMDEN et R. A. KOWALSKI, The semantics of Predicate Logic as a programming Language, *J. A.C.M.*, 1976, *23*, 4, p. 733-742.

[v.Em.Ll. 12] M. H. VAN EMDEN et W. LLOYD, A logical reconstruction of Prolog II, Proc. 2nd Conference on Logic Programming, Uppsala, Sweden, 1984, p. 35-40.

[Ja.La. 13] J. JAFFAR et J. L. LASSEZ, Constraint Logic Programming, Technical Report, Dept. of Comp. Sci., Monash University, juin 1986.

[Ja.La. 14] J. JAFFAR et J. L. LASSEZ, Constraint Logic Programming, *in* Proc. Conf. on Principles of Programming Languages, 1987.

[J.L.M. 15] J. JAFFAR, J. L. LASSEZ et M. J. MAHER, A logical foundation for Prolog II, technical report n 45, Monash University, 1985.

[J.L.M. 16] J. JAFFAR, J. L. LASSEZ et M. J. MAHER, Some issues and trends in the semantics of Logic Programming, Proc. 3rd International Conference on Logic Programming LNCS 225, Springer-Verlag, 1986.

[Ja.St. 17] J. JAFFAR et P. STUCKEY, Semantics of infinite tree logic programming, *Theoret. Comput. Sci.*, 1986, *46*, p. 141-158.

[Llo. 18] J. W. LLOYD, Foundation of Logic Programming, Symbolic Computation, Springer-Verlag, 1984.

[Mac. 19] A. MACINTYRE, Model Completeness, *in* Handbook of Mathematical Logic, J. BARWISE éd., North-Holland, 1978.

[Mah. 20] M. J. MAHER, Complete axiomatization of the algebras of finite, rational and infinite trees, Draft of paper in 3rd Logic in Computer Science Conference, Edinburgh, 1988.

[Ni.Pe. 21] M. NIVAT et D. PERRIN, Automata on Infinite Words, École de Printemps d'Informatique Théorique, Le Mont-Dore, mai 1984, *Lecture Notes in Comput. Sci.*, *192*.

[Re.Sa. 22] A. RESTIVO et S. SALEMI, Overlap free words on two symbols, *Lecture Notes in Comput. Sci.*, 1984, *192*, p. 198-206.