QASP ou la programmation par ensembles réponses quantifiée

Igor Stéphan

QASP ou la programmation par ensembles réponses quantifiée

Igor Stéphan *

LERIA, Université d'Angers
2, bd Lavoisier, 49045 Angers Cedex 01
igor.stephan@info.univ-angers.fr

Résumé

Nous présentons un nouveau paradigme de programmation logique qui étend la programmation par ensembles réponses propositionnelle en lui ajoutant la possibilité de quantifier universellement certains atomes. Ce nouveau paradigme permet de coder de manière plus compacte des problèmes PSPACE sans avoir recours à la programmation logique par ensembles réponses au premier ordre. Nous montrons que les formules booléennes quantifiées peuvent être très simplement codées en ce nouveau formalisme.

Mots Clef

Programmation par ensembles réponses, quantification, ASP, formules booléennes quantifiées, QBF.

Abstract

We present in this article a new logical programming paradigm which extends propositional ASP by adding quantifications on atoms. This new paradigm allows to encode in a compact way PSPACE problems without resorting to first order ASP. We show that quantified Boolean formulas can be encoded easily with this new formalism.

Keywords

Answer Set Programming, quantification, QBF

1 Introduction

Le terme de programmation par ensembles (de) réponses correspond au paradigme de l'*Answer Set Programming* (ASP), qui est un formalisme approprié pour représenter de nombreux problèmes issus de l'intelligence artificielle dans lesquels l'information disponible est incomplète comme dans le raisonnement non-monotone, la planification, le diagnostic, etc. D'un point de vue général, l'ASP est un cadre couvrant plusieurs sémantiques déclaratives pour différentes sortes de programmes logiques. En ASP, l'information est codée par des règles logiques et les solutions sont obtenues par des ensembles d'information appelés modèles stables. Chaque modèle est un ensemble minimal d'atomes (ou littéraux) contenant des informations sûres (des faits) et les déductions sont obtenues en appliquant certaines des règles par défaut. Ainsi, les conclusions dépendant des informations présentes et absentes, forment un ensemble cohérent d'hypothèses et représentent un point de vue rationnel du monde décrit par les règles. Derrière le terme générique d'ASP, il existe plusieurs variantes syntaxiques et sémantiques, mais dans ce travail nous utiliserons la sémantique originelle des modèles stables [7] pour les programmes logiques normaux.

L'ASP offre à la fois un modèle formel valide et des systèmes opérationnels. C'est aussi un cadre commode pour coder et résoudre des problèmes combinatoires. Plusieurs systèmes opérationnels sont disponibles aujourd'hui pour traiter l'ASP. Smodels [15], Clasp [6] et DLV [12] sont les plus connus d'entre eux. Cependant, ces solveurs travaillent sur des règles propositionnelles après une étape préalable d'instanciation des variables. D'un autre côté, certains solveurs traitent le programme avec variables en travaillant directement sur les règles qui ne sont pas propositionnelles et en limitant l'instanciation [10, 8, 5].

L'utilisation de l'ASP est contrastée du moment où le problème à résoudre est d'une complexité non déterministiquement polynomiale (NP) ou au-delà. Dans le premier cas, le codage du problème est généralement réalisé en fait en ASP propositionnel alors que dans le second cas le codage du problème est généralement réalisé au premier ordre avec une explosion possiblement exponentielle de la taille du problème propositionnel réellement traité *in fine* (sauf pour les solveurs qui travaillent directement au premier ordre). En particulier, les problèmes de la classe de complexité spatiale polynomiale (PSPACE) ne font pas exception à cette règle. Dans le cadre du problème de satisfiabilité d'une formule propositionnelle (SAT) qui est de complexité NP, ce n'est pas le cas, il existe une extension de SAT à la classe PSPACE : les formules booléennes quantifiées (QBF) [14, 3]. Le formalisme QBF est une extension du formalisme de la logique propositionnelle dans laquelle les symboles propositionnels peuvent être non seulement quantifiés existentiellement (comme en logique propositionnelle pour le problème de satisfiabilité) mais aussi universellement. De l'outil théorique [14], il est devenu un domaine de recherche appliqué [3] avec de nombreuses procédures de décision et de nombreuses utilisations en intelligence artificielle pour sa capacité à décrire de manière plus compacte la connaissance (voir [1] pour un florilège).

*Ce travail a bénéficié de l'aide de l'Agence Nationale de la Recherche, projet ASPIQ portant la référence ANR-12-BS02-0003.

Nous nous proposons dans cet article d'étendre l'ASP en lui ajoutant une possibilité de quantification sur les atomes pour pouvoir exprimer simplement et de manière compacte des problèmes dont la complexité se situe dans PSPACE. Nous montrons alors comment les QBF peuvent être codées dans ce nouveau paradigme appelé « programmation par ensembles réponses quantifiée » (ou QASP pour *Quantified Answer Set Programming*).

2 Préliminaires

Un programme logique normal est un ensemble fini de règles de la forme

$$(c \leftarrow a_1,\ \ldots,\ a_n,\ not\ b_1,\ \ldots,\ not\ b_m.)$$

$n \geqslant 0, m \geqslant 0$ où $c, a_1, \ldots, a_n, b_1, \ldots b_m$ sont des atomes qui constituent l'ensemble $\mathcal{A}$ et $\mathcal{P}$ désigne l'ensemble des programmes logiques normaux. Une telle règle peut se lire « si tous les a_i appartiennent à un ensemble réponse et si aucun des b_j n'y appartient, alors c doit appartenir à cet ensemble réponse ». Le « *not* » étant une négation par défaut, un tel programme peut-être vu comme un sous cas d'une théorie de défauts de Reiter [11]. Pour une règle r, on note $tête(r) = c$ sa tête, $corps^+(r) = \{a_1, \ldots,\ a_n\}$ son corps positif et $corps^-(r) = \{b_1, \ldots,\ b_m\}$ son corps négatif. Le réduit (dit de Gelfond et Lifschitz) d'un programme P par un ensemble d'atomes X est le programme $P^X = \{tête(r) \leftarrow corps^+(r).\ |\ r \in P, corps^-(r) \cap X = \emptyset\}$. Un tel programme est dit défini puisqu'il ne possède pas de négation par défaut et il admet un unique modèle de Herbrand minimal noté $Cn(P)$. Par définition, un modèle stable de P est un ensemble d'atomes $S \subseteq \mathcal{A}$ tel que $S = Cn(P^S)$. Notons qu'un programme peut avoir aucun, un ou plusieurs modèles stables. Par exemple, $\{a.,\ b \leftarrow a, not\ d.,\ c \leftarrow a, not\ b.\}$ possède l'unique modèle stable $\{a, b\}$, $\{a \leftarrow not\ b.\ \ , b \leftarrow not\ a.\}$ possède deux modèles stables $\{a\}$ et $\{b\}$ et $\{a.,\ b \leftarrow a, not\ d.,\ d \leftarrow b.\}$ n'en possède aucun.

3 Programmation par ensembles réponses quantifiée

Nous présentons la syntaxe de la programmation par ensembles réponses quantifiée comme une extension directe de la programmation par ensembles réponses à laquelle sont ajoutés des quantifications sur les atomes.

Définition 3.1 (programme normal quantifié) *Le symbole $\exists$ est utilisé pour la quantification existentielle et $\forall$ pour la quantification universelle (q est utilisé pour noter un quantificateur quelconque). Par convention, des quantificateurs différents lient des atomes différents. Un lieur est une chaîne de caractère $q_1 x_1 \ldots q_n x_n$ avec $x_1, \ldots, x_n$ des atomes distincts et $q_1 \ldots q_n$ des quantificateurs. L'ensemble des lieurs est noté* **LIEUR**. *Un* programme normal quantifié QP *est la donnée d'un lieur Q et d'un programme normal P dont tous les atomes sont présents dans le lieur Q. L'ensemble des programmes normaux quantifiés est noté $\mathcal{QP}$.*

Exemple 3.1 *L'expression suivante est un programme normal quantifié : $\exists e \forall a \exists na \exists b \exists nb \exists s P_1$ avec*

$$\begin{aligned} P_1 = \{\ & e \leftarrow not\ e, not\ s., b \leftarrow not\ nb., \\ & nb \leftarrow not\ b., a \leftarrow not\ na., na \leftarrow not\ a., \\ & s \leftarrow a, nb., s \leftarrow not\ a, not\ nb.\} \end{aligned}$$

Un programme normal quantifié peut être vu comme un jeu à deux joueurs. Le résultat est donc un arbre qui dicte au joueur existentiel comment il doit jouer selon les coups du joueur universel pour gagner de manière certaine. Cet arbre est ici un ensemble de fonctions d'appartenance.

Définition 3.2 (fonction d'appartenance) *Les symboles in et out sont les* marqueurs d'appartenance. *Une fonction partielle ϕ de l'ensemble des atomes dans l'ensemble des fonctions de $\{in, out\}^n$ dans $\{in, out\}$ est une* fonction d'appartenance *pour un lieur si pour tout atome existentiellement quantifié x il existe un unique couple $(x \mapsto \overline{x}) \in \phi$ tel que la fonction $\overline{x}$ a pour arité le nombre d'atomes universellement quantifiés qui précèdent x dans ce lieur. La fonction set associe à un atome x et un marqueur d'appartenance un ensemble d'atomes ainsi : $set_x(in) = \{x\}$ et $set_x(out) = \emptyset$.*

Les fonctions d'appartenance sont à rapprocher des fonctions de Skolem dont elles s'inspirent. La fonction set est étendue naturellement aux nuplets et ensembles ; les atomes en sont omis lorsqu'il n'y a pas d'ambiguïté.

Exemple 3.2 *L'ensemble ϕ_1 ci-dessous est une fonction d'appartenance pour le lieur $\exists e \forall a \exists na \exists b \exists nb \exists s$:*

$$\begin{aligned} &\{\overline{na}_a = \{(in \mapsto out), (out \mapsto in)\}, \\ &\ \overline{b}_a = \{(in \mapsto out), (out \mapsto in)\}, \\ &\ \overline{nb}_a = \{(in \mapsto in), (out \mapsto out)\}, \\ &\ \overline{s}_a = \{(in \mapsto in), (out \mapsto in)\}, \overline{e} = out\} \end{aligned}$$

$\phi_1(in) = \{\overline{na} = out, \overline{b} = out, \overline{nb} = in, \overline{s} = in, \overline{e} = out\}$ *et* $set_a(\phi_1(in)) = \{nb, s\}$.
$\phi_1(out) = \{\overline{na} = in, \overline{b} = in, \overline{nb} = out, \overline{s} = in, \overline{e} = out\}$ *et* $set_a(\phi_1(out)) = \{na, b, s\}$.

La sémantique de la programmation par ensembles réponses quantifiée est définie alors comme une extension naturelle de celle des programmes logiques normaux.

Définition 3.3 (ensemble réponse quantifié) *Une fonction d'appartenance ϕ est un ensemble réponse quantifié (ou modèle stable quantifié) pour un programme normal quantifié QP dont le nombre d'atomes universellement quantifiés de Q est n si pour tout n-uplet $x \in \{in, out\}^n$,*

$$Cn((P \cup fait(set(x)))^{set(x) \cup set(\phi(x))}) = set(x) \cup set(\phi(x))$$

avec la fonction $fait$ qui associe à un ensemble d'atomes X un ensemble de faits $\{x.\}_{x \in X}$.

Exemple 3.3 *Le programme normal quantifié $\forall a \exists b P_2$ avec $P_2 = \{b \leftarrow a., b \leftarrow not\ a.\}$ admet pour unique ensemble réponse quantifié la fonction d'appartenance $\phi_2 = \{\overline{b} = \{(in \mapsto in), (out \mapsto in)\}$. En effet :*

– $set_a(in) = \{a\}$, $\phi_2(in) = \{\overline{b} = in\}$, $set_a(\phi_2(in)) = \{b\}$, $P_2 \cup \{fait(\{a\})\} = \{b \leftarrow a., b \leftarrow not\ a., a.\}$ *et* $Cn(\{b \leftarrow a., b \leftarrow not\ a., a.\}^{\{a\}\cup\{b\}}) = \{a, b\}$;
– $set_a(out) = \emptyset$, $\phi_2(out) = \{\overline{b} = in\}$, $set_a(\phi_2(out)) = \{b\}$, $P_2 \cup \{fait(\emptyset)\} = \{b \leftarrow a., b \leftarrow not\ a.\}$ *et* $Cn(\{b \leftarrow a., b \leftarrow not\ a.\}^{\emptyset\cup\{b\}}) = \{b\}$

Le programme normal quantifié de l'exemple 3.1 $\exists e \forall a \exists na \exists b \exists nb \exists s P_1$ *admet pour unique ensemble réponse quantifié la fonction d'appartenance de l'exemple 3.2. En effet :*

– $set(in) = \{a\}$, $set(\phi_1(in)) = \{nb, s\}$,

$$(P_1 \cup \{a.\})^{set(in)\cup set(\phi_1(in))} = (P_1 \cup \{a.\})^{\{a,nb,s\}} = \{a., nb., s \leftarrow a, nb.\}$$

$$Cn((P_1 \cup \{a.\})^{set(in)\cup set(\phi_1(in))}) = set(in) \cup set(\phi_1(in))$$

– $set(out) = \emptyset$, $set(\phi_1(out)) = \{na, b, s\}$,

$$P_1^{set(out)\cup set(\phi_1(out))} = P_1^{\{na,b,s\}} = \{na., b., s., s \leftarrow a, nb.\}$$

$$Cn(P_1^{set(out)\cup set(\phi_1(out))}) = set(out) \cup set(\phi_1(out))$$

Le programme normal quantifié (dont on a inversé les quantificateurs du précédent programme) $\exists e \exists b \exists nb \forall a \exists na \exists s P_1$ *n'admet pas d'ensemble réponse quantifié.*

Le théorème qui suit exprime qu'un programme normal quantifié ne contenant que des quantificateurs existentiels est en fait un programme par ensembles réponses classique.

Théoreme 3.1 *Une fonction d'appartenance* ϕ *est un ensemble réponse quantifié pour un programme normal quantifié ne contenant que des quantificateurs existentiels si et seulement si* $set(\phi)$ *est un modèle stable du programme sans les quantificateurs.*

Le théorème qui suit démontre l'appartenance du problème de décision pour les programmes par ensembles quantifiés à la classe de complexité PSPACE-complet.

Théoreme 3.2 *Le problème de décision de l'existence d'un ensemble réponse quantifié pour un programme normal quantifié est un problème dont la classe de complexité est PSPACE-complet.*

L'appartenance à PSPACE du problème de décision de l'existence d'un ensemble réponse quantifié pour un programme normal quantifié est immédiate de part la nature de la définition 3.3. Le caractère complet est une conséquence du théorème 4.1 de la section suivante qui montre comment tout problème de décision sur la validité d'une QBF peut être ramené à un problème de décision sur l'existence d'un ensemble réponse quantifié pour un programme normal quantifié.

4 QBF en QASP

Déterminer si un programme possède ou non un modèle stable est un problème NP-complet et donc le lien avec le problème NP-complet canonique a été introduit dans [9] dans le cas où l'on s'intéresse à une formule propositionnelle donnée sous forme normale conjonctive (i.e. une conjonction de disjonctions de littéraux). Nous rappelons ici l'approche proposée. Soit Σ un ensemble de clauses. Le processus consiste à fabriquer un programme $\Pi_{SATFNC\to ASP}(\Sigma)$ contenant les règles $na \leftarrow not\ a.$ et $a \leftarrow not\ na.$ pour tout atome a apparaissant dans Σ. Pour toute clause dans Σ, on crée un nouvel atome c et pour tout littéral l de cette clause, on ajoute dans $\Pi_{SATFNC\to ASP}(\Sigma)$ la règle $c \leftarrow a.$ si l est un atome a et la règle $c \leftarrow na.$ si l est la négation d'un atome a. Enfin, on ajoute à $\Pi_{SATFNC\to ASP}(\Sigma)$ la règle $\leftarrow not\ c.$ [1]. De cette manière Σ possède un modèle propositionnel si et seulement si $\Pi_{SATFNC\to ASP}(\Sigma)$ possède un modèle stable. On remarque aisément que les premières paires de règles de $\Pi_{SATFNC\to ASP}(\Sigma)$ permettent de générer toutes les interprétations possibles pour Σ, les règles dont la tête est c permettent d'inférer c si l'interprétation satisfait la clause correspondante et les contraintes finales écartent des ensembles de réponses possibles tous ceux qui ne contiennent pas c. C'est-à-dire tous ceux qui ne correspondent pas à des modèles propositionnels de Σ.

Nous présentons d'abord des préliminaires nécessaires sur les formules booléennes quantifiées puis nous en décrivons notre traduction vers les programmes par ensembles réponses quantifiés.

4.1 Formules propositionnelles et booléennes quantifiées

L'ensemble des valeurs booléennes $vrai$ et $faux$ est noté $\mathcal{B}$. L'ensemble des symboles propositionnels est noté $\mathcal{SP}$. Les constantes $\top$ et $\bot$ sont des symboles tels que $\top$ est toujours interprété à $vrai$ et $\bot$ est toujours interprété à $faux$. Le symbole $\wedge$ est utilisé pour la conjonction, $\vee$ pour la disjonction, $\neg$ pour la négation, $\rightarrow$ pour l'implication et $\leftrightarrow$ pour l'équivalence. L'ensemble des opérateurs $\{\wedge, \vee, \rightarrow, \leftrightarrow\}$ est noté $\mathcal{O}$. L'ensemble **PROP** des formules propositionnelles est défini inductivement ainsi : tout symbole propositionnel (ainsi que $\top$ et $\bot$) est élément de **PROP** ; si F est élément de **PROP** alors $\neg F$ est élément de **PROP** ; si F et G sont éléments de **PROP** et $\circ$ est élément de $\mathcal{O}$ alors $(F \circ G)$ est élément de **PROP**. La définition des notions de « quantificateur » et de « lieur » rejoint celles des QBF en remplaçant simplement l'atome par le symbole propositionnel. L'ensemble des lieurs QBF est aussi noté **LIEUR** pas abus de langage. Une QBF QF est formée d'un lieur Q et d'une formule propositionnelle F (parfois appelée « matrice »). Soient Σ et σ deux formules telles que σ soit une sous formule de Σ, la fonction o : **PROP** $\times$ **PROP** $\rightarrow \{0,1\}^*$ est telle que $o(\sigma, \Sigma)$

1. Une telle règle sans tête est appelée contrainte et est un raccourci pour la règle $e \leftarrow not\ c, not\ e.$ où e est un nouveau symbole.

calcule l'occurrence de σ dans Σ ainsi : $o(\Sigma, \Sigma) = \epsilon$; $o(\sigma, \Sigma) = 0o(\sigma, \Sigma_0)$ si $\Sigma = \neg\Sigma_0$; $o(\sigma, \Sigma) = 0o(\sigma, \Sigma_0)$ si $\Sigma = (\Sigma_0 \circ \Sigma_1), \circ \in \mathcal{O}$ et σ est une sous formule de Σ_0 ; $o(\sigma, \Sigma) = 1o(\sigma, \Sigma_1)$ si $\Sigma = (\Sigma_0 \circ \Sigma_1), \circ \in \mathcal{O}$ et σ est une sous formule de Σ_1 [2].

La sémantique des symboles booléens est définie de manière habituelle. Une valuation est une fonction de l'ensemble des symboles propositionnels dans $\mathcal{B}$; elle satisfait une formule si appliquée à celle-ci elle vaut $vrai$ sinon elle la falsifie. Un modèle (i.e. une valuation qui satisfait la formule) est décrit par un ensemble de littéraux ; par exemple, pour la valuation ν qui est telle que $\nu(x) = vrai$, $\nu(y) = faux$ et $\nu(z) = vrai$ qui satisfait la formule $((x \vee y) \leftrightarrow z)$, ce modèle est noté $\{x, \neg y, z\}$. La sémantique des quantificateurs est la suivante : pour tout symbole propositionnel y et QBF F,

$$\exists y F = (F[y \leftarrow \top] \vee F[y \leftarrow \bot])$$

et

$$\forall y F = (F[y \leftarrow \top] \wedge F[y \leftarrow \bot]).$$

Une QBF F est valide si $F \equiv \top$. Si y est un symbole propositionnel quantifié existentiellement précédé par les symboles propositionnels quantifiés universellement $x_1, \ldots, x_n$, nous notons $\hat{y}_{x_1,\ldots,x_n}$ sa fonction de Skolem de $\mathcal{B}^n$ dans $\mathcal{B}$. Un modèle pour une QBF F est un ensemble de fonctions de Skolem qui satisfont la formule. Par exemple, la QBF $\exists y \exists x \forall z((x \vee y) \leftrightarrow z)$ n'est pas valide tandis que $\forall z \exists y \exists x((x \vee y) \leftrightarrow z)$ l'est avec pour ensemble possible de fonctions de Skolem $\{\hat{y}_z, \hat{x}_z\}$ [3] tel que $\hat{y}_z(vrai) = vrai$, $\hat{y}_z(faux) = faux$, $\hat{x}_z(vrai) = faux$ et $\hat{x}_z(faux) = faux$. Les fonctions de Skolem sont parfois représentées par des politiques [4] ou des stratégies [2] qui les explicitent en terme d'arbre. Un modèle (booléen) pour une formule booléenne non quantifiée correspond exactement au modèle (QBF) de sa clôture existentielle ; par exemple pour la QBF $\exists y \exists x \exists z((x \vee y) \leftrightarrow z)$, les fonctions de Skolem $\hat{x} = vrai$, $\hat{y} = faux$ et $\hat{z} = vrai$ correspondent au modèle booléen $\nu(x) = vrai$, $\nu(y) = faux$ et $\nu(z) = vrai$ pour la formule propositionnelle $((x \vee y) \leftrightarrow z)$. Une QBF est valide si et seulement s'il existe un ensemble de fonctions de Skolem qui la satisfait.

Enfin, rappelons que le problème (SAT) consistant à décider si une formule booléenne est satisfiable ou non est le problème canonique de la classe NP-complet. De son côté, le problème (QBF) consistant à décider si une formule booléenne quantifiée est valide ou non est le problème PSPACE-complet canonique.

4.2 Traduction d'une QBF en un programme normal quantifié

Dans [13] est présentée une traduction des QBF vers les ASP au premier ordre. Nous présentons ci-dessous la fonction qui en est issue et qui traduit une formule propositionnelle en règles :

2. Comme de coutume, ϵo sera simplement noté o.

3. Pour tout $b \in \mathcal{B}$, la valuation $\nu(z) = b$, $\nu(x) = \hat{x}_z(b)$, $\nu(y) = \hat{y}_z(b)$ est un modèle de $((x \vee y) \leftrightarrow z)$.

Soient Σ, Σ_0 et Σ_1 trois formules propositionnelles, x un symbole propositionnel et o une occurrence, les fonctions $\Pi_{P\to R}$: **PROP** $\to \mathcal{P}$ et $\pi_{P\to R}$: **PROP** $\times \{0,1\}^* \to \mathcal{P}$ sont définies par :

$$\Pi_{P\to R}(\Sigma) = \pi_{P\to R}(\Sigma, \epsilon)$$

si $\Sigma = x$ alors $\pi_{P\to R}(\Sigma, o) = \{s_o \leftarrow x.\}$

si $\Sigma = \neg\Sigma_0$ alors $\pi_{P\to R}(\Sigma, o) = \{s_o \leftarrow not\ s_{o0}.\} \cup \pi_{P\to R}(\Sigma_0, o0)$

si $\Sigma = (\Sigma_0 \wedge \Sigma_1)$ alors $\pi_{P\to R}(\Sigma, o) = \left\{ \begin{array}{l} s_o \leftarrow s_{o0}, s_{o1}. \end{array} \right\} \cup \pi_{P\to R}(\Sigma_0, o0) \cup \pi_{P\to R}(\Sigma_1, o1)$

si $\Sigma = (\Sigma_0 \vee \Sigma_1)$ alors $\pi_{P\to R}(\Sigma, o) = \left\{ \begin{array}{l} s_o \leftarrow s_{o0}., \\ s_o \leftarrow s_{o1}. \end{array} \right\} \cup \pi_{P\to R}(\Sigma_0, o0) \cup \pi_{P\to R}(\Sigma_1, o1)$

si $\Sigma = (\Sigma_0 \to \Sigma_1)$ alors $\pi_{P\to R}(\Sigma, o) = \left\{ \begin{array}{l} s_o \leftarrow not\ s_{o0}., \\ s_o \leftarrow s_{o1}. \end{array} \right\} \cup \pi_{P\to R}(\Sigma_0, o0) \cup \pi_{P\to R}(\Sigma_1, o1)$

si $\Sigma = (\Sigma_0 \leftrightarrow \Sigma_1)$ alors $\pi_{P\to R}(\Sigma, o) = \left\{ \begin{array}{l} s_o \leftarrow s_{o0}, s_{o1}., \\ s_o \leftarrow not\ s_{o0}, not\ s_{o1}. \end{array} \right\} \cup \pi_{P\to R}(\Sigma_0, o0) \cup \pi_{P\to R}(\Sigma_1, o1)$

Dans [13] les atomes représentant des symboles propositionnels existentiellement quantifiés ainsi que les atomes intermédiaires sont transformés en symbole de prédicat dont les arguments sont les combinaisons sur les symboles propositionnels universellement quantifiés.

Cette fonction $\Pi_{P\to R}$ est utilisée dans la définition ci-dessous qui traduit une QBF en un programme par ensembles réponses quantifiés. L'idée essentielle et apport de cette traduction est de coder les symboles propositionnels quantifiés existentiellement en deux atomes quantifiés existentiellement mais de coder les symboles propositionnels quantifiés universellement en deux atomes l'un quantifié universellement et l'autre existentiellement, dépendant fonctionnellement du premier.

Définition 4.1 *Soit la fonction* $\Pi_{Q\to Q}$: **LIEUR** $\to$ **LIEUR** *qui traduit un lieur QBF en un lieur QASP.*

$$\Pi_{Q\to Q}(Q) = \exists bug \pi_{Q\to Q}(Q)$$
$$\pi_{Q\to Q}(\exists x Q) = \exists x \exists nx \pi_{Q\to Q}(Q)$$
$$\pi_{Q\to Q}(\forall x Q) = \forall x \exists nx \pi_{Q\to Q}(Q)$$

Soit la fonction $\Pi_{Q\to R}$: **LIEUR** $\to \mathcal{P}$ *qui traduit un lieur QBF en une paire de règles qui lie les littéraux positif et négatif d'un même atome quantifié (q un quantificateur quelconque).*

$$\Pi_{Q\to R}(Q) = \{bug \leftarrow not\ s_\epsilon, not\ bug.\} \cup \pi_{Q\to R}(Q)$$
$$\pi_{Q\to R}(qxQ) = \{x \leftarrow not\ nx., nx \leftarrow not\ x.\} \cup \pi_{Q\to R}(Q)$$

*Soit la fonction $\pi_{QBF\to QASP}$: **QBF** $\to \mathcal{QP}$ qui traduit une QBF en un programme normal quantifié.*

$$\pi_{QBF\to QASP}(QM) = \Pi_{Q\to Q}(Q)(\Pi_{Q\to R}(Q) \cup \Pi_{P\to R}(M))$$

La définition suivante spécifie un ensemble de fonctions pour traduire les ensembles réponses quantifiés en modèle QBF et, pour partie, réciproquement.

Définition 4.2 *La fonction de traduction $\pi : \mathcal{B} \to \{in, out\}$ associe à chaque valeur booléenne un marqueur d'appartenance à un ensemble réponse.*

$$\pi(vrai) = in, \pi(faux) = out$$

$$\pi^{-1}(in) = vrai, \pi^{-1}(out) = faux$$

La fonction π et sa réciproque sont étendues aux fonctions : $\pi : (\mathcal{B}^n \to \mathcal{B}) \to (\{in, out\}^n \to \{in, out\})$ et $\pi^{-1} : (\{in, out\}^n \to \{in, out\}) \to (\mathcal{B}^n \to \mathcal{B})$.

$$\begin{array}{l}\pi(sk) = \\ \quad \{((\pi(b_1), \ldots, \pi(b_n)) \mapsto \pi(b)) \mid \\ \quad\ ((b_1, \ldots, b_n) \mapsto b) \in sk, b_1, \ldots, b_n, b \in \mathcal{B}\} \\ \pi^{-1}(\phi) = \\ \quad \{((\pi^{-1}(t_1), \ldots, \pi^{-1}(t_n)) \mapsto \pi^{-1}(t)) \mid \\ \quad\ ((t_1, \ldots, t_n) \mapsto t) \in \phi, t_1, \ldots, t_n, t \in \{in, out\}.\}\end{array}$$

Enfin, la fonction $\pi : (\mathcal{B}^n \to \mathcal{B})^* \to (\{in, out\}^n \to \{in, out\})^*$ traduit un ensemble de fonctions de Skolem QBF en un ensemble de fonctions d'appartenance :

$$\pi(m) = \{\pi(\hat{x}) \mid \hat{x} \in m\}$$

*La « réciproque » $\pi^{-1} : (\mathcal{B}^n \to \mathcal{B})^* \times$ **LIEUR** $\to (\{in, out\}^n \to \{in, out\})^*$ nécessite la présence d'un lieur en paramètre pour sélectionner les atomes existentiellement quantifés :*

$$\pi^{-1}(\phi, Q) = \{\pi^{-1}(\overline{x}) \mid \text{"}\exists x\text{"} \in Q, \overline{x} \in \phi\}$$

Exemple 4.1 *Soit la QBF $F = \forall a \exists b \forall c \exists d M$ avec $M = ((c \vee b) \wedge (b \to ((c \to d) \wedge (c \vee (a \leftrightarrow \neg d)))))$*
Le lieur de la QBF est traduit en un lieur de programme normal quantifié

$$\begin{array}{l}\Pi_{Q\to Q}(\forall a \exists b \forall c \exists d) = \\ \quad \exists bug \forall a \exists na \exists b \exists nb \forall c \exists nc \exists d \exists nd \\ \quad \exists s_\epsilon \exists s_0 \exists s_1 \exists s_{11} \exists s_{110} \exists s_{111} \exists s_{1111}\end{array}$$

et en l'ensemble des règles

$$\begin{array}{ll}\Pi_{Q\to R}(\forall a \exists b \forall c \exists d) \\ = & \{b \leftarrow not\ nb., nb \leftarrow not\ b., \\ & \quad a \leftarrow not\ na., na \leftarrow not\ a., \\ & \quad c \leftarrow not\ nc., nc \leftarrow not\ c., \\ & \quad d \leftarrow not\ nd., nd \leftarrow not\ d.\}\end{array}$$

La formule propositionnelle est traduite en l'ensemble des règles[4]

$$\begin{array}{ll}\Pi_{P\to R}(M) \\ = & \{bug \leftarrow not\ bug, not\ s_\epsilon., \quad s_\epsilon \leftarrow s_0, s_1., \\ & \quad s_0 \leftarrow c., \quad s_0 \leftarrow b., \\ & \quad s_1 \leftarrow b., \quad s_1 \leftarrow s_{11}., \\ & \quad s_{11} \leftarrow s_{110}, s_{111}., \\ & \quad s_{110} \leftarrow not\ c., \quad s_{110} \leftarrow d., \\ & \quad s_{111} \leftarrow c., \quad s_{111} \leftarrow s_{1111}., \\ & \quad s_{1111} \leftarrow a, nd., \quad s_{1111} \leftarrow not\ a, not\ nd.\}\end{array}$$

Ainsi la QBF est traduite en le programme normal quantifié

$$\begin{array}{ll}\pi_{QBF\to QASP}(\forall a \exists b \forall c \exists d\ M) \\ = & \Pi_{Q\to Q}(\forall a \exists b \forall c \exists d) P\end{array}$$

avec

$$P = \Pi_{Q\to R}(\forall a \exists b \forall c \exists d) \cup \Pi_{P\to R}(M)$$

L'unique ensemble réponse quantifié ϕ de ce programme normal quantifié est

$$\begin{array}{l}\phi = \\ \quad \{\overline{na}_a, \overline{b}_a, \overline{nb}_a, \overline{nc}_{ac}, \overline{d}_{ac}, \overline{nd}_{ac}, \overline{s_\epsilon}_{ac}, \overline{s_0}_{ac}, \overline{s_1}_{ac}, \\ \quad\ \overline{s_{11}}_{ac}, \overline{s_{110}}_{ac}, \overline{s_{111}}_{ac}, \overline{s_{1111}}_{ac}\}\end{array}$$

avec :

$$\begin{array}{l}\overline{na}_a = \{(in \mapsto out), (out \mapsto in)\} \\ \overline{b}_a = \{(in \mapsto in), (out \mapsto in)\} \\ \overline{nb}_a = \{(in \mapsto out), (out \mapsto out)\} \\ \overline{nc}_{ac} = \{((in, in) \mapsto out), ((in, out) \mapsto in), \\ \quad ((out, in) \mapsto out), ((out, out) \mapsto in)\} \\ \overline{d}_{ac} = \{((in, in) \mapsto in), ((in, out) \mapsto out), \\ \quad ((out, in) \mapsto in), ((out, out) \mapsto in)\} \\ \overline{nd}_{ac} = \{((in, in) \mapsto out), ((in, out) \mapsto in), \\ \quad ((out, in) \mapsto out), ((out, out) \mapsto out)\} \\ \overline{s_\epsilon}_{ac} = \{((in, in) \mapsto in), ((in, out) \mapsto in), \\ \quad ((out, in) \mapsto in), ((out, out) \mapsto in)\}\end{array}$$

Pour tout s_o avec $o \in \{s_0, s_1, s_{11}, s_{110}, s_{111}\}$,

$$\begin{array}{ll}\overline{s_o}_{ac} & = \\ & \{((in, in) \mapsto in), ((out, out) \mapsto in), \\ & \ ((in, out) \mapsto in), ((out, in) \mapsto in)\} \\ \overline{s_{1111}}_{ac} & = \\ & \{((in, in) \mapsto out), ((in, out) \mapsto in), \\ & \ ((out, in) \mapsto in), ((out, out) \mapsto in)\}\end{array}$$

En effet, par exemple,

$$\begin{array}{l}\phi(in, in) = \\ \quad \{\overline{na}_a(in, in) = out, \overline{b}_a(in, in) = in, \\ \quad\ \overline{nb}_a(in, in) = out, \overline{nc}_{ac}(in, in) = out, \\ \quad\ \overline{d}_{ac}(in, in) = in, \overline{nd}_{ac}(in, in) = out, \\ \quad\ \overline{s_\epsilon}_{ac}(in, in) = in, \overline{s_0}_{ac}(in, in) = in, \\ \quad\ \overline{s_1}_{ac}(in, in) = in, \overline{s_{11}}_{ac}(in, in) = in, \\ \quad\ \overline{s_{110}}_{ac}(in, in) = in, \overline{s_{111}}_{ac}(in, in) = in, \\ \quad\ \overline{s_{1111}}_{ac}(in, in) = out\}\end{array}$$

4. Les règles du type $s_o \leftarrow x.$ avec s_o un atome intermédiaire et x un atome issu d'un symbole propositionnel sont omises et les atomes s_o substitués par les atomes x dans le reste des règles.

et

$$set(\phi(in,in)) = \{b, d, s_\epsilon, s_0, s_1, s_{11}, s_{110}, s_{111}\}$$

et $(set((in,in)) = \{a,c\})$

$$Cn((P \cup \{a., c.\})^{\{a,c\} \cup set(\phi(in,in))}) = \{a,c\} \cup set(\phi(in,in))$$

Enfin, on extrait aisément l'unique modèle QBF $\pi^{-1}(\phi, \forall a \exists b \forall c \exists d) = \{\hat{b}_a, \hat{d}_{ac}\}$ *avec*

$$\hat{b}_a = \pi^{-1}(\bar{b}_a) = \{(vrai \mapsto vrai), (faux \mapsto vrai))\}$$
$$\hat{d}_{ac} = \pi^{-1}(\bar{d}_{ac}) = \{((vrai, vrai) \mapsto vrai), ((vrai, faux) \mapsto faux), (((faux, vrai) \mapsto vrai), ((faux, faux) \mapsto vrai)\}$$

Le théorème suivant assure la correction et complétude de la traduction.

Théoreme 4.1 (correction et complétude) *Soit une QBF* QF.
Si m *est un modèle QBF de* QF *alors il existe un ensemble* $\mathcal{F}$ *de fonctions* $\{in, out\}^n \rightarrow \{in, out\}$ *telles que* $\pi(m) \cup \mathcal{F}$ *est un ensemble réponse quantifié de* $\pi_{QBF \rightarrow QASP}(QF)$.
Si ϕ *est un ensemble réponse quantifié de* $\pi_{QBF \rightarrow QASP}(QF)$ *alors* $\pi^{-1}(\phi, Q)$ *est un modèle QBF de* QF.

5 Conclusion

Nous avons présenté un nouveau paradigme de programmation logique qui étend la programmation par ensembles réponses propositionnelle en lui ajoutant la possibilité de quantifier universellement ou existentiellement les atomes. Nous avons codé notre traduction en Prolog et le programme est librement disponible [5]. La description de la sémantique et le programme Prolog laisse présager une extension aisée des procédures de décision orientées atome. Nous envisageons d'explorer l'algorithmique pour des procédures de décision orientées règle. Nous envisageons aussi de faire une plus large étude des applications de la programmation par ensembles réponses pour voir celles qui peuvent obtenir un codage plus compact grâce à l'introduction des quantificateurs. Nous nous intéresserons particulièrement aux jeux à deux jours à horizon fini où la description sous la forme de règles ASP des règles du jeu semble prometteuse.

Références

[1] M. Benedetti and H. Mangassarian. Experience and Perspectives in QBF-Based Formal Verification. *Journal on Satisfiability, Boolean Modeling and Computation*, 5 :133–191, 2008.

[2] L. Bordeaux. Boolean and interval propagation for quantified constraints. In *Proceedings of the 1st International Workshop on Quantification in Constraint Programming (QCP'05)*, 2005.

[3] M. Cadoli, M. Schaerf, A. Giovanardi, and M. Giovanardi. An Algorithm to Evaluate Quantified Boolean Formulae and Its Experimental Evaluation. *Journal of Automated Reasoning*, 28(2) :101–142, 2002.

[4] S. Coste-Marquis, H. Fargier, J. Lang, D. Le Berre, and P. Marquis. Representing Policies for Quantified Boolean Formulae. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, pages 286–296, 2006.

[5] M. Dao-Tran, T. Eiter, M. Fink, G. Weidinger, and A. Weinzierl. OMiGA : An Open Minded Grounding On-The-Fly Answer Set Solver. In *JELIA*, pages 480–483, 2012.

[6] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. Conflict-Driven Answer Set Solving. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 386–392, 2007.

[7] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the 5th International Conference on Logic Programming*, pages 1070–1080, 1988.

[8] C. Lefèvre and P. Nicolas. A First Order Forward Chaining Approach for Answer Set Computing. In *Proceedings of the 10th Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09)*, pages 196–208, 2009.

[9] I. Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3-4) :241–273, 1999.

[10] A. Dal Palù, A. Dovier, E. Pontelli, and G. Rossi. Gasp : Answer set programming with lazy grounding. *Journal Fundamenta Informaticae*, 96(3) :297–322, 2009.

[11] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2) :81–132, 1980.

[12] F. Ricca and N. Leone. Disjunctive logic programming with types and objects : The DLV$^+$ system. *Journal of Applied Logic*, 5(3) :545–573, 2007.

[13] I. Stéphan, B. Da Mota, and P. Nicolas. From (Quantified) Boolean Formulas to Answer Set Programming. *Journal of logic and computation*, 19(4) :565–590, 2009.

[14] L.J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3 :1–22, 1977.

[15] T. Syrjänen and I. Niemelä. The Smodels System. In *LPNMR*, pages 434–438, 2001.

5. http://www.info.univ-angers.fr/pub/stephan/Research/Download.html