Autour des procédures de décision pour l'arithmétique linéaire

Un tour bibliographique
Journée ADT Coq Automatisation

Assia Mahboubi

INRIA Saclay – Île-de-France / TypiCal

23/03/2009

Le(s) problème(s)

Les formules linéaires du premier ordre à coefficients entiers avec ordre et égalité:

- comme :

$$7 < 1664$$
$$\forall x \forall y \exists z \quad x \geq y \vee [x < 2z \wedge 2z < y]$$

- mais pas comme :

$$\exists x \exists y \exists z \quad x^5 + y^5 = z^5$$
$$b^2 - 4ac$$

Où l'on interprète les variables dans $\mathbb{N}$, $\mathbb{Z}$, ou $\mathbb{Q}$.

Plus formellement

$$
\begin{array}{lll}
\text{num} & ::= & 0 \mid 1 \\
\text{var} & ::= & x \mid y \mid z \mid \ldots \\
\text{op} & ::= & < \mid > \mid \leq \mid \geq \mid = \\
\text{term} & ::= & \text{num} \mid \text{var} \mid \text{term} + \text{term} \mid - \text{term} \mid \text{num} \times \text{term} \\
\text{formula} & ::= & \text{term op term} \mid \\
 & & \text{formula} \wedge \text{formula} \mid \text{formula} \vee \text{formula} \mid \\
 & & \neg \text{ formula} \mid \exists \text{ var formula} \mid \forall \text{ var formula}
\end{array}
$$

L'astuce sur $\mathbb{N}$, $\mathbb{Z}$

$$
\begin{array}{lll}
\text{num} & ::= & 0 \mid 1 \\
\text{var} & ::= & x \mid y \mid z \mid \ldots \\
\text{op} & ::= & < \mid > \mid \leq \mid \geq \mid = \\
\text{term} & ::= & \text{num} \mid \text{var} \mid \text{term} + \text{term} \mid - \text{term} \mid \text{num} \times \text{term} \\
\text{formula} & ::= & \text{term op term} \mid \\
 & & \text{formula} \wedge \text{formula} \mid \text{formula} \vee \text{formula} \mid \\
 & & \neg \text{ formula} \mid \exists \text{ var formula} \mid \forall \text{ var formula} \\
 & & [\text{numeral} \mid \text{term}]
\end{array}
$$

Les problèmes

- Ce ne sont pas des théorèmes très intéressants.
- Mais à la main, leur solution est fatiguante.
- Ils sont (très) fréquents pour les prouveurs de programmes.
- Les théories sont décidables.

$\Rightarrow$ Cas typique pour recourir à des procédures de décision

La complexité des problèmes

- Les problèmes de mathématiciens (avec quantificateurs) :
  - $T_{\mathbb{Z}}$: Il existe une constante $c > 0$, telle que pour tout n assez grand, il existe une formule de longueur n dont la décision demande au moins $2^{2^{cn}}$ opérations.
  - $T_{\mathbb{Q}}$: Il existe une constante $c > 0$, telle que pour tout n assez grand, il existe une formule de longueur n dont la décision demande au moins 2^{cn} opérations.

  (Fisher Rabin (1977))
- Les problèmes d'informaticiens (sans quantificateurs) :
  - Au moins NP-durs ...
  - Mais tout est dans la partie logique propositionnelle

Les problèmes avec quantificateurs

Par élimination des quantificateurs (formules closes) :

- Si une formule n'a pas de quantificateurs alors c'est facile :

$$9 + 89 < 6 * 2$$

- A partir de P avec n quantificateurs, on calcule Q avec $m < n$ quantificateurs avec $P \Leftrightarrow Q$.

En pratique :

- On normalise (collecte des coefficients . . .).
- On élimine les $\forall$ grâce à la décidabilité des littéraux:

$$\forall x\ P(x) \Leftrightarrow \neg\exists x \neg P(x)$$

- C'est là qu'on a besoin de l'astuce sur $\mathbb{N}$

Arithmétique rationnelle : Fourier (- Motzkin)

- Élimine le $\exists$ le plus profond : $\exists x\ P(x)$
- On se ramène à éliminer sur $\exists x\ a_1 x \wedge \ldots a_{1n} x$
  - par mise en DNF de P
  - par distribution du $\exists$
- Pour une paire de contraintes, on utilise le théorème : $(\exists x\ a \leq cx \wedge dx \leq b) \Leftrightarrow ad \leq cb$
  - $\Rightarrow$ par transitivité
  - $\Leftarrow$ pour $x = \frac{b}{d}$

  ainsi que ses 3 variantes ...
- On généralise sans problème à plus de contraintes.

Et ça marche pour l'arithmétique entière?

Non : maintenant le théorème n'est pas :

$$(\exists x\ a \leq cx \wedge dx \leq b) \Leftrightarrow ad \leq cb$$

Mais seulement :

$$(\exists x\ a \leq cx \wedge dx \leq b) \Rightarrow ad \leq cb$$

Mais ça donne quand même une heuristique pour les $\forall x P(x)$:

- Transformer en $\neg\exists x\ (\neg P(x))$
- Montrer que $\exists x\ (\neg P(x)) \Rightarrow \bot$
- Alors $\forall x P(x) \Leftrightarrow \neg\exists x\ (\neg P(x)) \Leftrightarrow \top$

C'est la tactique omega de Coq

Le test Omega de Pugh (1992)

On passe à la phase 2, dite phase de l'ombre...

$$F := \exists x[\bigwedge_i a_i \leq c_i x \ \wedge \ \bigwedge_j d_j x \leq b_j]$$

- L'ombre réelle de F est : $\bigwedge_{i,j} a_i d_j \leq c_i b_j$
  Elle est équivalente à F dès qu'un $c_i = 1$ ou un $d_j = 1$ (ombre exacte)
- L'ombre foncée de F est : $\bigwedge_{i,j}(c_i - 1)(d_j - 1) \leq b_j c_i - a_i d_j$
  Si elle est vraie, F aussi.

Le test Omega de Pugh, complet

- Phase 1 : si on a que des $\forall$, on essaie avec Fourier
- Phase 2 :
  - On calcule l'ombre réelle et on teste son exactitude,
  - sinon, on essaie de montrer que F est vraie en calculant son ombre foncée.
- Phase 3: le théorème complet parle aussi des échardes de F, une disjonction qui traite le cas général, dite phase 3.

Pugh et Norrish prétendent que les phases 1 et 2 couvrent la plupart des cas rencontrés dans leur contextes respectifs.

État des lieux de l'implantation en Coq

- Une tactique Fourier en Ocaml (L. Pottier)
- Une tactique (R)Omega en Ocaml (P. Crégut, P. Letouzey), qui ne fait essentiellement que la phase 1, c'est à dire Fourier
- Pas de réflexion (totale)
- Pas de modularité sur la représentation des nombres

Comment faire mieux ?

- Abstraire les nombres (travail de E. Makarov ?)
- Reprendre à zéro l'implémentation de la décision de Presburger
- Une autre approche algorithmique?

Élimination des quantificateurs de Cooper (1972)

Pré-processing de $F := \exists x\ P(x)$:

- ne garder les $\neg$ qu'autour des atomes de divisibilité
- réduire les opérateurs à $<$ et $n|$.
- calculer le ppcm c des coefficients de x dans P et ramener tous ces coefficients à c
- changer de variable : $\exists x P(cx) \Leftrightarrow \exists y P(y) \wedge c|y$

On a plus que des atomes de la forme :

- $c < x$ ou $x < c$: position de x sur la droite des entiers
- $n|x$: contraintes sur les diviseurs de x

Pourquoi F peut être vraie? - 1er cas

Parce qu'il y a des solutions aussi petites qu'on veut.

- On regarde la "limite" des atomes de P :
  - $x < a$ sera $\top$ si x est assez petit
  - $a < x$ sera $\bot$ si x est assez petit
  - $n|x + a$ sera inchangé
- $P_{-\infty}$ est cette formule, qui n'a que des atome de divisibilité.
- Soit p le ppcm des constantes de $P_{-\infty}$, il suffit de tester $P_{-\infty}$ sur $1 \ldots p$.

Pourquoi F peut être vraie? - 2ème cas

Parce qu'il y a une plus petite solution x_0.

- Alors forcément il y a un atome $c < x$ qui est vrai pour x_0 mais pas pour les x plus petits.
- Soit $C := \{c \mid c < x \text{ est un atome de P}\}$
- Il suffit de tester les $P(c + j)$ où $c \in C$ et $j = 1 \dots p$

La formule finale est donc :

$$\exists x\ P(x) \Leftrightarrow \bigvee_{j=1\dots p} P_{-\infty}(j) \vee \bigvee_{j=1\dots p} \bigvee_{c \in C} P(c + j)$$

Avantages de l'approche Cooper

- Assez facile
- Plus besoin de DNF (méthode du point intérieur)
- Moins d'explosition dans la taille de la formule sans quantificateur mais comparaison des performances difficile (Norrish 2003)
- Analogue pour $\mathbb{Q}$ (Ferrante -Rackoff (1975) et Voos - Weispfennig (1993)), mais sans partage de code
- Procédé d'élimination des quantificateurs plus facile à factoriser (Isabelle/HOL - Nipkov (2008))

Complexité

- Méthode de Cooper : Il existe une constante p telle que Cooper demande $2^{2^{2^{pn}}}$ opérations pour décider une formule de taille n.
- Méthode de Ferrante et Rackhoff : Il existe une constante p telle que F-R demande $2^{2^{pn}}$ opérations pour décider une formule de taille n.

L'approche SMT

Beaucoup des problèmes concrets de la preuve de programme ont peu de structure logique :

- Pas de quantificateurs (fragment existentiel)
- Une structure logique purement propositionnelle

$\Rightarrow$ Utilisation de Solveurs Modulo Theorie (SMT) :

- Un SAT solveur
- Une procédure de décision sur la théorie sans quantificateurs
- Une coopération entre le SAT solveur et la procédure de décision

L'approche SMT pour la preuve formelle?

- Stéphane a construit une architecture SMT en Coq.
- Les systèmes automatiques, même hors DeCert (Z3,...) s'orientent vers la production de certificats.
- On peut même retrouver de l'élimination des quantificateurs efficace (D. Monniaux 2008)

$\Rightarrow$ On a besoin de procédures efficaces pour ces théories sans quantificateurs, en Coq

Des algorithmes complètement différents

Basés sur des techniques de programmation linéaire et d'optimisation :

- Simplexe (Kantorovitch (1939) - Dantzig (1947))
- Coupes (ex. de Gomory) pour le cas entier

Preuves de correction :

- Algèbre linéaire (Bibliothèques ssreflect)
- Arguments de convexité (Besson (?), Pasca (2008))

Conclusion

- Coq est sous-équipé en procédures de décision arithmétiques.
- Omega ne reflète pas les possibilités actuelles du système.
- Heureusement on a Micromega mais on dépend de la précision de l'oracle.
- Quelle priorité pour l'automatisation ?
  - approche complète pour la preuve mathématique (Micromega?)?
  - approche sans quantificateurs pour la preuve de programmes?
- Dans tous les cas, séparer la logique du calcul sur la théorie.
- Quelle possibilités pour le partage du code?
  - Polynômes?
  - Nombres?
  - Élimination des quantificateurs?
  - Lien avec les procédures non-linéaires?