Une procédure de décision pour une combinaison propositionnelle d'égalités et d'inégalités

Jean-Christophe Filliâtre
LRI, Orsay

En collaboration avec S. Owre, H. Rueß et N. Shankar
SRI International, Menlo Park, USA

LACL, 12 novembre 2001

Plan

1. Survol d'ICS
2. Une combinaison de techniques
3. Quelques mots sur l'implantation

1

Survol d'ICS

ICS (*Integrated Canonizers and Solvers*)

ICS est une procédure de décision pour une théorie

- du premier ordre
  - symboles de fonctions non interprétés
  - symboles de fonctions interprétés dans des théories (arithmétique, n-uplets, tableaux, ensembles, etc.)
- sans quantificateur
- égalité / diségalité
- inégalité linéaire (sur les rationnels)
- logique propositionnelle

Exemples

ICS peut montrer

1. $f^3(x) = f(x) \vdash f^5(x) = f(x)$
2. $\vdash 2x + 3y = 13 \ \wedge \ 3x = 2y \ \Rightarrow \ x = 2$
3. $f(f(x) - f(y)) \neq f(z), \ y \leq x, \ y \geq x + z, \ z \geq 0 \vdash \mathsf{false}$
4. $a + 2 = b \vdash f(A[a := 3][b - 2]) = f(b - a + 1)$
5. $\vdash ((p \Rightarrow q) \Rightarrow p) \Rightarrow p$

Interface “ask/tell”

Interface conçue pour une utilisation incrémentale

```
type state
val empty_state : state

val process : state → prop → ⊥+state

val check : state → prop → bool
```

2

Une combinaison de techniques

Une combinaison de techniques

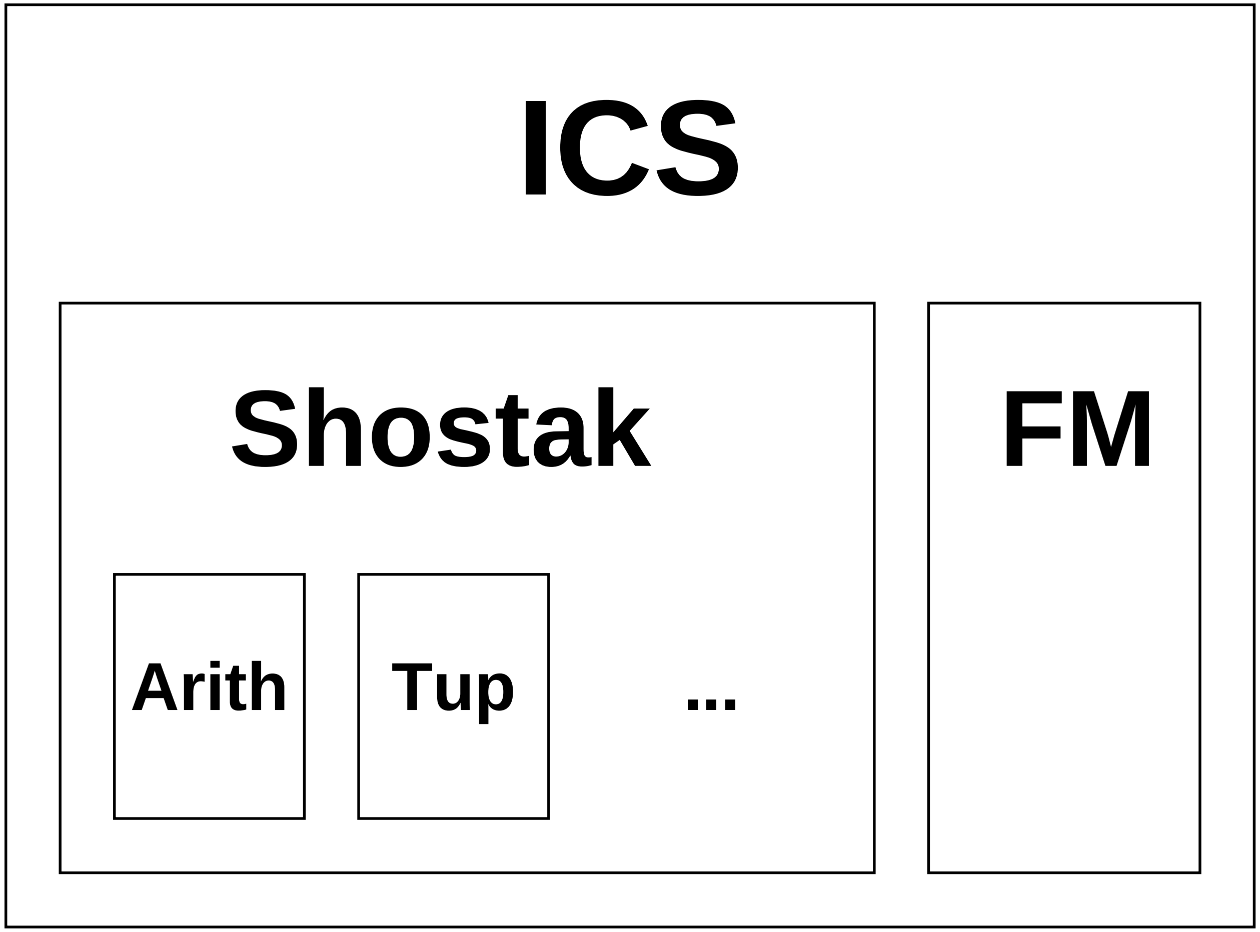
ICS repose sur

– l'algorithme de Shostak pour l'égalité

– une élimination de Fourier-Motzkin pour les inégalités

– une combinaison à l'intérieur d'un SAT solver

Conception modulaire d'ICS



Termes

$$\begin{array}{llll} \text{termes} & t & ::= & x \mid f(t, \ldots, t) \\ \text{atomes} & a & ::= & t = t \mid t \neq t \mid t < t \mid t \leq t \mid \mathsf{true} \mid \mathsf{false} \\ \text{propositions} & p & ::= & a \mid \mathsf{if}\ a\ \mathsf{then}\ p\ \mathsf{else}\ p \end{array}$$

Théorie “canonisable”

1. $\models a = b$ si et seulement si $\sigma(a) \equiv \sigma(b)$
2. $\sigma(x) \equiv x$
3. $vars(\sigma(a)) \subseteq vars(a)$
4. $\sigma(\sigma(a)) \equiv \sigma(a)$
5. Si $\sigma(a) \equiv f(b_1, \ldots, b_n)$ alors $\sigma(b_i) \equiv b_i$ pour $1 \leq i \leq n$

Exemple : $\sigma(-4 + 3y + 5x - x + 1) \equiv -3 + 4x + 3y$

Théorie “solvable”

– $solve(a = b) = \perp$ si $a = b$ n'est pas satisfiable
– $solve(a = b) = \{x_1 = e_1, \ldots, x_n = e_n\}$ avec
  1. $x_i \in vars(a = b)$
  2. $x_i \notin vars(e_j)$
  3. $x_i \not\equiv x_j$
  4. $\sigma(e_i) \equiv e_i$
  5. $solve(a = b)$ et $a = b$ sont σ-équivalents

Exemple : $solve(-3x + 2 = y + x) = \{x = 1/2 - y/4\}$

L'algorithme de Shostak

L'algorithme de Shostak original (1984) est en fait incomplet et non-terminant.

Récemment « réparé » dans

Deconstructing Shostak (Shankar and Rueß, LICS 2001)

```
type estate (* ensemble d'équations *)

val eprocess : estate → term = term → ⊥+estate

val canon : estate → term → term
```

Inégalités

Élimination de Fourier-Motzkin

```
type istate (* ensemble d'inéquations *)

val iprocess : istate → ⊥+équations
```

État de la procédure de décision

L'état d'ICS est un quadruplet (E, D, I, P) où

- E est un ensemble d'équations (un état pour Shostak)
- D est un ensemble de dis-équations
- I est un ensemble d'inéquations
- P est un ensemble de propositions

La fonction process

$$
\begin{array}{lll}
\multicolumn{3}{l}{\mathit{process}(E,D,I,P)(p) = \text{match } \mathit{canon}(E)(p) \text{ with}} \\
\mid a = b & \Rightarrow & \text{let } E' = \mathit{eprocess}(E)(a = b) \text{ in} \\
 & & \mathit{processmany}(E', \emptyset, \emptyset, \emptyset)(D \cup I \cup P) \\
\mid a \neq b & \Rightarrow & \text{if } a \equiv b \text{ then } \bot \text{ else } (E, \{a \neq b\} \cup D, I, P) \\
\mid a \lhd b & \Rightarrow & \text{let } E' = \mathit{iprocess}(\{a \lhd b\} \cup I) \text{ in} \\
 & & \mathit{processmany}(E, D, \{a \lhd b\} \cup I, P)(E') \\
\mid \text{true} & \Rightarrow & (E, D, I, P) \\
\mid \text{false} & \Rightarrow & \bot \\
\mid p' & \Rightarrow & (E, D, I, \{p'\} \cup P)
\end{array}
$$

$$\mathit{processmany}(S)(\{p_1, \ldots, p_n\}) = \mathit{process}((\ldots \mathit{process}(S, p_1) \ldots), p_n)$$

La fonction `check`

On vérifié en fait l'inconsistance d'un état, vu que

$$check(S)(p) \quad \equiv \quad inconsistent(process(S)(\neg p))$$

La fonction `inconsistent`

$inconsistent(E, D, I, P) =$ match P with

$\mid \emptyset \Rightarrow$ false

$\mid \{$if a then p else $q\} \cup P' \Rightarrow$

let $S' = processmany(E, D, I, P')(\{a, p\})$ in

let $S'' = processmany(E, D, I, P')(\{\neg a, q\})$ in

match S', S'' with

$\mid \bot, \bot \Rightarrow$ true

$\mid \bot, \_ \Rightarrow inconsistent\ S''$

$\mid \_, \bot \Rightarrow inconsistent\ S'$

$\mid \_ \Rightarrow inconsistent\ S' \ \wedge \ inconsistent\ S''$

Résultats

$process$ termine et est correcte.

$inconsistent$ termine, est correcte et complète.

Deciding Propositional Combinations of Equalities and Inequalities,

J.-C. Filliâtre, S. Owre, H. Ruess et N. Shankar

Soumis à FroCos'02 (`www.lri.fr/~filliatr`)

3

Quelques mots sur l'implantation

En quelques mots

ICS est écrit en Objective Caml

Approximativement 5000 lignes de code

Utilise GMP (GNU Multi Precision library)

ICS est documenté avec ocamlweb, un outil de programmation littéraire pour Ocaml $\Rightarrow$ document de 80 pages

ICS est partie intégrante de PVS

mais aussi bibliothèque indépendante (C / Ocaml / Lisp)

Efficacité

- hash-consing : les termes égaux sont partagés $\Rightarrow$ l'égalité est immédiate
- structures de données efficaces pour les polynômes, les états, les dictionnaires indexés par des termes, etc.
- structures de données persistantes $\Rightarrow$ pas de copie
- un langage fortement typé $\Rightarrow$ représentation compacte des valeurs

Distribution

ICS est une bibliothèque et un programme interactif

```
% ics

> assert y <= x; y >= x + z; z >= 0.

> check x = y.
Valid.
```

Comparaison avec SVC (Stanford Validity Checker)

Système linéaire de N équations avec solutions entières

N	ICS		SVC	
5	0.05		0.07	
10	0.43		0.44	
15	2.36		1.84	
20	8.85		8.84	
25	25.04		23.05	
30	61.95	(29 Mo)	71.18	(91 Mo)
35	130.25	(40 Mo)	$>$ 4 mn	($>$ 150 Mo)

Conclusion

Aspects positifs

- Une procédure de décision indépendante de PVS
- Une propriété de complétude
- Une implantation fonctionnelle $\Rightarrow$ argument simple de correction (voire une preuve formelle)

Aspects négatifs

- Implantation naïve de Fourier-Motzkin (SVC ou Omega plus efficaces)
- N'entre pas en compétition avec des procédures de décision spécialisées
- Nécessite encore des tests de performance poussés (travail en cours)