Projet

À rendre pour le 10 décembre 2012.

Le but de ce projet est d'implanter des techniques de preuve de terminaison en utilisant les ordres dits KBO et ACKBO ((Associative-Commutative) Knuth-Bendix Orderings).

1 KBO

Définition 1 *Soit une signature définie par un ensemble de symboles $\mathcal{F}$. Un ordre KBO est défini par une fonction de poids w et une précédence $\prec$ de la façon suivante :*

- *Fonction de* poids $w : \mathcal{F} \to \mathbb{N}$ *t.q.* $w(a) > 0\ \forall a$ *constante,*
- *Le poids d'une variable est défini comme celui de la constante de poids minimal (on supposera que $\mathcal{F}$ contient au moins une constante),*
- *$\prec$ est une précédence w-compatible si elle est totale et si pour tout symbole f tel que $w(f) = 0$, f est maximal et unique,*
- *$|s|_x$ est le nombre d'occurrences de la variable x dans le terme s,*
- *Le poids d'un terme t, noté $|t|$, est égal à*

$$|g(t_1, \ldots, t_n)| = w(g) + \sum_{i=1}^{n} |t_i|$$

- $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $s >_{KBO} t$ *si* $\forall x, |s|_x \geq |t|_x$ *et soit*
  - $|s| > |t|$
  - $|s| = |t|$, $s = f(s_1, \ldots, s_n), t = g(t_1, \ldots, t_k)$ *et* $g \prec f$
  - $|s| = |t|$, $s = f(s_1, \ldots, s_n), t = g(t_1, \ldots, t_k)$, $f =_\prec g$ *et* $(s_1, \ldots, s_n) >^{lex}_{KBO} (t_1, \ldots, t_k)$
  - $s = f^n(x), t = x$ *si* $w(f) = 0$.
- $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $s =_{KBO} t$ *si*
  - $\forall x, |s|_x = |t|_x$ *et*
  - $|s| = |t|$, $s = f(s_1, \ldots, s_n), t = g(t_1, \ldots, t_k)$, $f =_\prec g$ *et* $(s_1, \ldots, s_n) =_{KBO} (t_1, \ldots, t_k)$.

Étant donné un système de contraintes (C) de la forme

$$\bigwedge_{i=1}^{n} s_i > t_i \wedge \bigwedge_{j=1}^{m} u_j = v_j$$

on cherche une précédence $\prec$ et une fonction de poids w qui définissent un KBO satisfaisant (C).

Plan de campagne :

- Étant donnés $\prec$ et w, écrire une fonction qui teste si deux termes sont $<_{KBO}$, $=_{KBO}$, $>_{KBO}$ ou incomparables.
- Étant donné un problème de terminaison, c'est-à-dire un système de contraintes, traduire ces contraintes en un problème SAT qui reflète les contraintes sur $\prec$ et w (*cfr* TP et [1]).

2 ACKBO

Cette fois, certains symboles peuvent être associatifs-commutatifs. On suppose que l'on dispose d'une fonction is_AC qui teste si un symbole est AC ou non, et que les termes sont aplatis.

Définition 2 *Un ordre ACKBO est défini par une fonction de poids w et une précédence $\prec$ de la façon suivante :*

- *Le poids d'une variable est défini comme celui de la constante de poids minimal,*
- $|g(t_1, \ldots, t_n)| = w(g) + \sum |t_i|$ *si* $g \notin \mathcal{F}_{\mathrm{AC}}$
- $|g(t_1, \ldots, t_n)| = (n-1)w(g) + \sum |t_i|$ *si* $g \in \mathcal{F}_{\mathrm{AC}}$
- *Le poids* généralisé *d'un terme t est une paire $[\![t]\!] = (|t|, var(t))$, où $var(t)$ est le* multi-*ensemble des variables de t. Les poids généralisés se comparent de la façon suivante :*
  - $(n', v') < (n, v)$ *si* $n > n'$ *et* $v \supseteq v'$
  - $(n', v') \leq (n, v)$ *si* $n \geq n'$ *et* $v \supseteq v'$
- $t <_w s$ *si* $[\![t]\!] < [\![s]\!]$
- $t \leq_w s$ *si* $[\![t]\!] \leq [\![s]\!]$
- $t \leq_{top} s$ *si* $t(\Lambda) \prec s(\Lambda)$ *ou* $t(\Lambda) = s(\Lambda)$
- $t \leq_{wtop} s$ *si* $t <_w s$ *ou* $t \leq_w s$ *et* $t \leq_{top} s$
- $t =_{wtop} s$ *si* $t =_w s$ *et* $t(\Lambda) = s(\Lambda)$
- $t <_{wtop} s$ *si* $t <_w s$ *ou* $t \leq_w s$ *et* $t(\Lambda) \prec s(\Lambda)$
- *La simplification de multi-ensembles est réalisée par applications tant que possible de l'opération* $del(\{\{s\}\} \cup M, \{\{t\}\} \cup N) = (M, N)$ *si* $s =_{wtop} t$ *(retrait d'une* occurrence *dans chaque multi-ensemble tant que faire se peut),*
- *La comparaison de deux multi-ensembles M et N relativement à un symbole f est définie, en posant* $M' = \{\{t \in M \mid t \in \mathcal{X} \text{ ou } f \prec t(\Lambda)\}\}$, $N' = \{\{t \in N \mid t \in \mathcal{X} \text{ ou } f \prec t(\Lambda)\}\}$ *et* $(M'', N'') = del(M', N')$*, par*
  - $M >_f N$ *si* $\exists s_0 \in M'' \mid s_0 \notin \mathcal{X}$ *et* $\forall t \in N'', \exists s \in M'' \mid s >_{wtop} t$,
  - $M \geq_f N$ *si* $M >_f N$ *ou bien si* N'' *est vide et* $\forall s \in M'', s \in \mathcal{X}$.

  On peut alors définir la comparaison ACKBO :
- $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $s >_{ACKBO} t$ *si, soit*
  - $t <_w s$
  - $t \leq_w s$, $s = f(s_1, \ldots, s_n), t = g(t_1, \ldots, t_k)$ *et* $g \prec f$
  - $t \leq_w s$, $s = f(s_1, \ldots, s_n), t = g(t_1, \ldots, t_k)$, $f = g$, $is\_AC(f) =$ false *et* $(s_1, \ldots, s_n) >^{lex}_{ACKBO} (t_1, \ldots, t_k)$
  - $t \leq_w s$, $s = f(s_1, \ldots, s_n), t = g(t_1, \ldots, t_k)$, $f = g$, $is\_AC(f) =$ true *et soit*
    - $\{\{s_1, \ldots, s_n\}\} >_f \{\{t_1, \ldots, t_k\}\}$
    - $\{\{s_1, \ldots, s_n\}\} \geq_f \{\{t_1, \ldots, t_k\}\}$ *et* $n > k$
    - $\{\{s_1, \ldots, s_n\}\} \geq_f \{\{t_1, \ldots, t_k\}\}$ *et* $n = k$ *et* $\{\{s_1, \ldots, s_n\}\} >^{mul}_{ACKBO} \{\{t_1, \ldots, t_k\}\}$
  - $t >_{ACKBO} x$*, où x est une variable si et seulement si* $x \in var(t)$ *et* $t \neq x$.
  - $x \not> _{ACKBO} t$*, quand x est une variable.*

Mêmes questions que précédemment mais maintenant dans un cadre AC :

- Étant donnés $\prec$ et w, écrire une fonction qui teste si deux termes sont $<_{ACKBO}, =_{ACKBO}, >_{ACKBO}$ ou incomparables.
- Étant donné un problème de terminaison, c'est-à-dire un système de contraintes, traduire ces contraintes en un problème SAT qui reflète les contraintes sur $\prec$ et w (*cfr* TP et [1]).

3 Logistique

Les systèmes de récriture seront au format TRS (`http://www.lri.fr/~marche/tpdb/format.html`, l'identifiant de théorie pour les symboles associatifs et commutatifs étant `AC`), et les systèmes de contraintes donnés en syntaxe CAML par :

```
type var = int;;
type symb = string;;

type term = Var of var | Term of symb * term list;;

(* resultat de la fonction de comparaison de deux termes *)
type comparison = Eq | Lt | Gt | Uncomparable

(* type des contraintes d'ordre *)
type constraint = Equal of term * term | Greater of term * term

type constraint_system = constraint list
```

Le projet est attendu en OCaML.

Vous pourrez tester vos programmes sur la base de données de problèmes de terminaison TPDB (`http://www.lri.fr/~marche/tpdb`).

Le cas échéant vous pourrez appliquer différents critères pour prouver la terminaison de systèmes de la base, au minimum Manna-Ness et DP.

Pour les critères plus avancés, vous êtes libres d'utiliser la structure de votre choix pour les graphes et une bibliothèque adaptée afin de faciliter vos manipulations de graphes (par exemple ocamlgraph `http://ocamlgraph.lri.fr/`). Nous vous fournissons des fonctions de *filtrage* et d'*unification* implantées en OCaML à `http://www.ensiie.fr/~urbain/enseignement/match_unif.ml`

Le rendu du projet comprend

1. Le code abondamment commenté,
2. Un manuel d'installation et d'utilisation de votre outil,
3. Un rapport de quelques pages expliquant vos choix,
4. Quelques tests.

Références

[1] Harald Zankl and Aart Middeldorp Satisfying KBO Constraints. In *Proceedings of the 18th International Conference on Rewriting Techniques and Applications (RTA 2007)*, volume 4533 of *LNCS*, pages 389-403, 2007. Springer-Verlag.