Université de Nice Sophia-Antipolis Année 2006/2007
Master 1 Informatique

Module Paradigmes et Langages de Programmation
Projets

Projet "Les systèmes de réécriture"
Resp. : Laurence Pierre

Descriptif

Le but de ce projet est de mettre en oeuvre un ***moteur de système de réécriture***.

Un système de réécriture (*) est formé d'un ensemble de règles de réécriture, c'est à dire de règles de la forme : $exp_g \rightarrow exp_d$ qui sont utilisées pour faire des transformations syntaxiques sur des termes.
Exemple : soit le système de réécriture

(1) $x + 0 \rightarrow x$
(2) $x + s(y) \rightarrow s(x + y)$

le terme *s(0 + s(0))* sera réécrit de la façon suivante :
s(0 + s(0)) devient s(s(0 + 0)) par application de la règle (2),
puis *s(s(0))* par application de la règle (1), et plus aucune règle ne s'applique alors.

Une règle exp_g -> exp_d s'applique à un terme t si ce terme contient une instance de exp_g, c'est à dire s'il existe une substitution qui permet de passer de exp_g à un sous-terme de t. Dans l'exemple ci-dessus, la substitution qui permet de passer de x + s(y) à 0 + s(0) est : $x \mapsto 0$, $y \mapsto 0$ et ensuite la substitution qui permet de passer de x + 0 à 0 + 0 est $x \mapsto 0$. Il s'agit en fait d'arriver à *filtrer* un sous-terme de t par exp_g.
Appliquer la règle correspond alors à réécrire le terme t en remplaçant l'instance de exp_g par l'instance de exp_d correspondante (on y applique la même substitution).

On remarquera que l'obtention d'une unique forme normale est garantie si le système de réécriture est *convergent* (c'est à dire qu'il présente les propriétés de *terminaison* et de *confluence*). Cette propriété n'est pas systématiquement garantie, le procédure de Knuth-Bendix peut par exemple être appliquée pour tenter de rendre confluent un système qui ne l'est pas.

On *ne se posera pas* ce problème ici. On supposera que les systèmes de réécriture fournis sont convergents (il suffit de considérer la première règle applicable) et on se contentera de mettre en oeuvre le moteur de réécriture. Etant donnés un système de réécriture S et un terme à réécrire t (données de l'algorithme), on itère le traitement suivant jusqu'à ce que plus aucune règle de S ne s'applique :

- on cherche s'il existe une règle r de S qui puisse s'appliquer à t
- on réécrit t en appliquant r

Mise en oeuvre

Le système de réécriture pourra être placé dans un fichier, dans la syntaxe de votre choix (mais qui devra être expliquée dans votre rapport), fichier préalablement lu et transformé dans un format interne (qui devra également être expliqué dans le rapport) par le moteur de réécriture.
Pour le choix du fichier, la saisie de l'expression (ou des expressions) à réécrire, le traçage des règles appliquées, et l'affichage du résultat, une interface graphique serait souhaitable mais n'est pas exigée.

Applications

1. Utilisation dans un contexte de démonstration automatique : un tel système peut être utilisé pour réaliser des preuves automatisées, par exemple de propriétés arithmétiques. Voyons deux exemples simples, dans un contexte de réécriture pure (pas d'ajout de mécanisme d'induction par exemple).
Attention, il est important que le système de réécriture choisi axiomatise correctement la théorie dans laquelle on souhaite raisonner.

Exemple 1 : soit le système

(1) $x + 0 \to x$
(2) $x + s(y) \to s(x + y)$
(3) $even(0) \to true$
(4) $even(s(0)) \to false$
(5) $even(s(s(x))) \to even(x)$
(6) $1 \to s(0)$
(7) $2 \to s(s(0))$
(8) $3 \to s(s(s(0)))$
...

A-t-on la propriété *even(x + 2) = even(x)* ?
Voici les réécritures mises en jeu :

$even(x + 2)$	$\to even(x + s(s(0)))$	par (7)
	$\to even(s(x + s(0)))$	par (2)
	$\to even(s(s(x + 0)))$	par (2)
	$\to even(s(s(x)))$	par (1)
	$\to even(x)$	par (5)

Exemple 2 : soit le système

(1) $x + 0 \to x$
(2) $x + s(y) \to s(x + y)$
(3) $div2(0) \to 0$
(4) $div2(s(0)) \to 0$
(5) $div2(s(s(x))) \to s(div2(x))$
(6) $1 \to s(0)$
(7) $2 \to s(s(0))$
(8) $3 \to s(s(s(0)))$
...

A-t-on la propriété *(x + 2) / 2 = x/2 + 1*, c'est à dire $div2(x + 2) = div2(x) + 1$?
Voici les réécritures mises en jeu :

	div2(x + 2)	→	div2(x + s(s(0)))	par (7)
		→	div2(s(x + s(0)))	par (2)
		→	div2(s(s(x + 0)))	par (2)
		→	div2(s(s(x)))	par (1)
		→	s(div2(x))	par (5)
et	div2(x) + 1	→	div2(x) + s(0)	par (6)
		→	s(div2(x) + 0)	par (2)
		→	s(div2(x))	par (1)

2. Utilisation dans un contexte de calcul formel : un tel système peut également être utilisé pour réaliser des manipulations symboliques dans un environnement de calcul formel.
Par exemple, on peut concevoir un système de réécriture qui implémente notamment les identités remarquables (voir http://fr.wikipedia.org/wiki/Identités_remarquables) et les identités trigonométriques (voir http://fr.wikipedia.org/wiki/Identité_trigonométrique). Ce système peut servir à transformer syntaxiquement des expressions arithmétiques (attention à ce qu'il soit convergent).

Exemple : soit le système

(1) sqr(a + b) → sqr(a) + 2 * a * b + sqr(b)
(2) sqr(a - b) → sqr(a) - 2 * a * b + sqr(b)
(3) (a + b) * (a - b) → sqr(a) - sqr(b)
(4) sqr(cos(a)) → (1 + cos(2 * a)) / 2
(5) sqr(sin(a)) → (1 - cos(2 * a)) / 2

l'expression sqr(3 + sin(x)) devient :

sqr(3 + sin(x))	→	sqr(3) + 2 * 3 * sin(x) + sqr(sin(x))	par (1)
	→	sqr(3) + 2 * 3 * sin(x) + (1 - cos(2 * x)) / 2	par (5)

Vous proposerez un système convergent plus riche que celui-là, et vous le testerez sur quelques exemples significatifs...

(*) Voir par exemple :
http://fr.wikipedia.org/wiki/Réécriture_(informatique)