Unification

Sujet proposé par Jean-Jacques Lévy

jean-jacques.levy@inria.fr

1 Préambule

Un problème standard du calcul symbolique est l'*unification*. Ce sujet consiste à programmer l'algorithme d'unification rapide. L'unification est l'opération de base de PROLOG pour appeller des procédures, mais intervient aussi dans les démonstrateurs de théorèmes.

2 Détail du sujet

Considérons par exemple les termes construits avec le symbole + (binaire) et quelques variables x, y, z, ... Alors les termes $x + x$ et $(y + z) + ((z + a) + a)$ qui ont une instance commune $((a + a) + a) + ((a + a) + a)$ sont dits *unifiables*. Mais les termes $x + x$ et $((y + z) + ((y + a) + a)$ ne le sont pas. Donc deux termes t et u sont unifiables s'ils donnent un terme identique par substitution de leur variables. On cherche donc une solution à l'équation $t = u$. L'algorithme de Robinson permet de construire l'*unificateur principal* de deux termes. Il consiste à résoudre un système d'équations par éliminninations de variables.

L'algorithme consiste à considérer un système E d'équations et à'itérer sur lui comme suit. Initialement, $E = \{t = u\}$. Puis, posons $E = E_1 \cup \{t = u\}$. On raisonne par cas:

1. $t = f(t_1, t_2, \ldots t_n)$ et $u = f(u_1, u_2, \ldots u_n)$. Alors on itère avec $E_1 \cup \{t_i = u_i | 1 \leq i \leq n\}$. (même symbole de tête)
2. $t = f(t_1, t_2, \ldots t_n)$ et $u = g(u_1, u_2, \ldots u_m)$ où $f \neq g$. Alors E n'a pas de solution. (symbole de tête différent)
3. $t = x$ et $u = x$. Alors on itère avec E_1.
4. $t = x$ et $u \neq x$. Alors soit x est une variable apparaissant dans u et alors E n'a pas de solution. Sinon, on compose la substitution résultat avec $(x \leftarrow u)$ et on itère avec E_1 où on a sustitué toutes les occurences de x par u. (élimination d'une variable)
5. les deux cas symétriques lorques u est une variable.

Un algorithme plus efficace, dit d'unification rapide, trouvé indépendamment par Martelli et Montanari et par Huet, consiste à utiliser l'algorithme de Fischer et Galler pour gérer dynamiquement des classes d'équivalences. Il nous faut donc présenter les deux algorithmes.

Dans l'unification rapide, on construit une relation d'équivalence $\doteq$ entre les termes à unifier. On oriente cette relation en considérant une relation $\Rightarrow$ telle que $t \Rightarrow u$ implique $t \doteq u$. Intuitivement $t = t_0 \Rightarrow t_1 \Rightarrow t_2 \Rightarrow t_n = u$ signifie que u est le représentant canonique de t dans sa classe d'équivalence. L'algorithme de l'unification rapide procède comme suit:

Initialement $E = \{t = u\}$. Puis, posons $E = E_1 \cup \{t = u\}$. On raisonne à nouveau par cas:

1. $t = f(t_1, t_2, \dots t_n)$ et $u = f(u_1, u_2, \dots u_n)$. Alors on fait $t \Rightarrow u$ et on itère avec $E_1 \cup \{t_i = u_i | 1 \leq i \leq n\}$. (même symbole de tête)
2. $t = f(t_1, t_2, \dots t_n)$ et $u = g(u_1, u_2, \dots u_m)$ où $f \neq g$. Alors E n'a pas de solution. (symbole de tête différent)
3. $t = x$ et $u = x$. Alors on itère avec E_1.
4. $t = x$ et il existe v tel que $x \Rightarrow v$, on recommence avec $E = \{v = u\} \cup E_1$.
5. cas symétrique
6. $t = x$ et il n'existe pas v tel que $x \Rightarrow v$, on pose $t \Rightarrow u$ et on recommence avec $E = E_1$.

Cet algorithme peut retourner des solutions infinies (sur les arbres rationnels, c'est-à-dire des arbres infinis avec un nombre fini de sous-arbres). Par exemple $x = f(x)$ a une solution. On peut néanmoins tester a posteriori la finitude des solutions en vérifiant que la solution ne contient aucune équation $x = t$ avec x strictement contenu dans t.

La deuxième partie consiste à appliquer l'algorithme de Fisher et Galler. En effet, quand on rajoute $\Rightarrow$, on peut mettre cette relation dans un sens ou dans l'autre dans les cas 1 et parfois 6. Or cela influe sur la vitesse pour remonter au représentant canonique de chaque classe. Il est trop long de décrire ici cet algorithme, qu'on peut trouver dans tout livre sur les algorithmes (cf. [5]). Cette algorithme fonctionne en temps quasi-linéaire.

Un troisième algorithme d'unification, dû à Paterson et Wegman, est linéaire, mais très compliqué à programmer.

3 Travail demandé

Il s'agit de lire deux termes à unifier en entrée et de générer la solution.

Les problèmes rencontrés dans ce projet tourneront principalement autour de: 1) la représentation des termes et de l'analyse syntaxique pour les lire et les imprimer, 2) de l'implantation de l'algorithme d'unification.

Tout besoin de documentation supplémentaire sera fourni sur demande.

Références

[1] B. A. Galler, M. Fischer, "An improved equivalence algorithm", Comm. ACM 7, pp. 301–303, 1964.

[2] A. Martelli, U. Montanari, "An efficient unification algorithm, In ACM Transactions on Programming Languages and Systems", Vol 4, (2), pp. 258–282, 1982.

[3] G. Huet, "Résolution d'équations dans les langages d'ordre 1,2,...ω", Thèse d'Etat, Paris 7, 1976.

[4] M. Paterson, M. Wegman, "Linear unification", Journal of Computer and System Sciences, (16) pp. 158–167, 1978.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, "Introduction to Algorithms", MIT Press et McGraw-Hill.