Déduction dans des théories équationnelles avec contraintes symboliques

Laurent Vigneron

CRIN-CNRS & INRIA-Lorraine
BP 239, 54506 Vandœuvre-lès-Nancy Cedex, France
E-mail : Laurent.Vigneron@loria.fr

La déduction automatique, lorsqu'elle est effectuée dans des théories équationnelles, soulève des problèmes importants comme celui de disposer d'un algorithme d'unification dans ces théories. Nous montrons dans ce résumé que l'utilisation de contraintes permet d'alléger ces problèmes et d'accéder à une gamme plus large de théories.

Les contraintes symboliques ont été introduites dans [KKR90] pour exprimer les problèmes d'unification et d'ordre. Elles généralisent la stratégie basique définie dans [Hul80], comme cela a été montré dans [BGLS92, NR92a, NR92b] pour la théorie vide et dans [NR94, Vig94] pour les théories associatives et commutatives, où des procédures réfutationellement complètes, basées sur les stratégies de paramodulation et de superposition, ont été définies pour ne jamais avoir à calculer un seul unificateur. Seule la satisfaisabilité des problèmes d'unification est nécessaire, ce qui est en général beaucoup plus simple.

Nous montrons ici que ces techniques peuvent être appliquées à d'autres théories équationnelles, et nous soulignons les avantages apportés, mais aussi les nouveaux centres d'intérêt qu'elles ouvrent.

1 Contraintes et règles d'inférence

Notons E l'ensemble des équations représentant la théorie dans laquelle nous voulons travailler. Nous allons utiliser deux types de contraintes symboliques élémentaires : les contraintes d'unification $(t =^? t')$ et les contraintes d'ordre $(t >^? t')$, appliquées à des termes ou des atomes t et t' de $\mathcal{T}(\mathcal{F}, \mathcal{X}) \cup \mathcal{A}(\mathcal{P}, \mathcal{F}, \mathcal{X})$.

Une contrainte d'unification $(t =^? t')$ représente toutes les substitutions σ telles que $\sigma(t)$ est E-égal à $\sigma(t')$. Une contrainte d'ordre $(t >^? t')$ représente toutes les substitutions σ telles que $\sigma(t) >_E \sigma(t')$, où $>_E$ est un ordre de simplification, total sur les classes de E-congruence.

Chaque clause C est accompagnée d'une conjonction de contraintes élémentaires c. Ainsi, $C\,[\![\, c \,]\!]$ représente toutes les clauses $\sigma(C)$ telles que σ est une substitution satisfaisant les contraintes contenues dans c.

Nous illustrons ci-dessous l'utilisation de ces contraintes pour des règles d'inférence classiques.

Définition 1 (Résolution)

$$\frac{A_1 \vee D_1 \llbracket c_1 \rrbracket \qquad \neg A_2 \vee D_2 \llbracket c_2 \rrbracket}{D_1 \vee D_2 \llbracket c_1 \wedge c_2 \wedge c_3 \rrbracket}$$

où c_3 *est* $(A_1 =^? A_2) \wedge (A_1 >^? D_1) \wedge (A_2 >^? D_2)$.

La clause déduite par une étape de résolution est composée des atomes résultant de la déduction, et d'une conjonction de contraintes composée des contraintes des clauses parentes, d'une contrainte d'unification $(A_1 =^? A_2)$, et de deux contraintes d'ordre permettant de garantir que les atomes utilisés A_1 et A_2 étaient maximaux dans leur clause respective.

Le principal intérêt des contraintes d'ordre est de conserver une trace des choix effectués, et ainsi d'éviter toute future incohérence.

La règle d'inférence suivante est la superposition, où $L_{2|_p}$ dénote le sous-terme de L_2 à la position p, $L_2[r_1]_p$ signifie que le sous-terme de L_2 à la position p a été remplacé par r_1.

Définition 2 (Superposition)

$$\frac{(l_1 = r_1) \vee D_1 \llbracket c_1 \rrbracket \qquad L_2 \vee D_2 \llbracket c_2 \rrbracket}{L_2[r_1]_p \vee D_1 \vee D_2 \llbracket c_1 \wedge c_2 \wedge c_3 \rrbracket}$$

où L_2 *est* $(l_2 = r_2)$ *ou* $\neg(l_2 = r_2)$, p *est dans* l_2,
et c_3 *est* $(L_{2|_p} =^? l_1) \wedge (l_1 >^? r_1) \wedge (l_2 >^? r_2) \wedge ((l_1 = r_1) >^? D_1) \wedge ((l_2 = r_2) >^? D_2)$.

Cette règle consiste à remplacer un sous-terme du littéral L_2, unifiable avec l_1 et se trouvant à une position non variable p, par r_1. Les contraintes ajoutées expriment, en plus du problème d'unification, que le remplacement doit être effectué dans le plus grand membre de l'équation L_2, c'est-à-dire l_2. Les autres contraintes d'ordre expriment la maximalité des littéraux utilisés pour la déduction.

Travailler dans des théories équationnelles signifie également qu'il faut traiter le cas des extensions, c'est-à-dire des clauses créées par les axiomes de la théorie E. Une solution simple est de considérer que ces extensions sont systématiquement ajoutées à l'ensemble de clauses, et qu'elles peuvent être être utilisées lors d'étapes de superposition. Pour plus de détails, voir [PS81].

2 A propos des règles de simplification

Une définition de règles de simplification compatibles avec des règles d'inférence comme celles citées ci-dessus a été proposée dans [BGLS92] pour la théorie vide, et dans [Vig94] pour les théories associatives et commutatives. Il existe deux règles principales de simplification :

- *subsomption*: étant donnée une clause $C_1 \llbracket c_1 \rrbracket$ et une solution σ_1 de c_1, l'instance $C_1 \llbracket \sigma_1 \rrbracket$ de $C_1 \llbracket c_1 \rrbracket$ subsume une clause $C_2 \llbracket c_2 \rrbracket$ si, pour chaque solution σ_2 de c_2, il existe une substitution ρ telle que $\rho(\sigma_1(C_1))$ est E-égal à $\sigma_2(C_2)$.
- *simplification*: c'est une variante de la règle de superposition, sans adjonction de contraintes ; elle est appliquée de l'instance $C_1 \llbracket \sigma_1 \rrbracket$ d'une clause $C_1 \llbracket c_1 \rrbracket$ si $\sigma_1(l_1) >_E \sigma_1(r_1)$. Il est à noter que cette clause $C_1 \llbracket c_1 \rrbracket$ peut être une clause étendue.

L'application de la règle de subsomption consiste à supprimer les clauses strictement subsumées, c'est-à-dire les clauses qui sont subsumées par une autre clause mais qui ne subsument pas cette clause.
L'application de la règle de simplification consiste à remplacer les clauses simplifiables par le résultat des étapes de simplification. Cependant, pour conserver la propriété de complétude, une autre condition doit être satisfaite : si une instance $C_1 [\![\sigma_1]\!]$ est utilisée pour subsumer ou simplifier toutes les instances $C_2 [\![\sigma_2]\!]$ d'une clause $C_2 [\![c_2]\!]$, chaque terme du co-domaine de σ_1 doit être inclus dans un terme du co-domaine de chaque σ_2. Lorsque cette condition n'est pas satisfaite, les étapes de simplification peuvent quand même être appliquées, à condition de transformer l'instance $C_1 [\![\sigma_1]\!]$ en une nouvelle clause $C_1\sigma_1' [\![\sigma_1'']\!]$, où $\sigma_1 = \sigma_1'\sigma_1''$ et la condition sur les co-domaines est satisfaite pour σ_1''.

Il est également possible de définir des règles de simplification qui ne s'appliquent qu'à une instance d'une clause. A partir de ce point, il a été montré que, si une instance $C_2 [\![\sigma_2]\!]$ d'une clause $C_2 [\![c_2]\!]$ est simplifiable dans le co-domaine de σ_2, cette instance est inutile. Pour plus de détails, consulter [BGLS92] et [Vig94].

3 Avantages et nouvelles préoccupations

L'application de procédés comme ceux décrits dans la section 1 à la déduction dans des théories équationnelles met en valeur les avantages de la stratégie contrainte.

- Une grande partie du contrôle utilisé dans le processus de déduction est traduit par les contraintes et leur condition de satisfaisabilité. C'est en particulier le cas de la stratégie ordonnée au travers des contraintes d'ordre.

- Il n'est plus besoin de calculer les E-unificateurs, grâce aux contraintes d'E-unification. Chaque déduction ne produit qu'une seule nouvelle clause, au lieu d'autant de clauses qu'il y a d'unificateurs principaux. Ainsi, des théories dont l'algorithme d'unification est décidable mais infinitaire peuvent être traitées sans aucun problème.

- L'accumulation de contraintes est une manière de conserver une trace des choix effectués : avec les contraintes d'ordre, chaque choix d'orientation d'une équation ou de maximalité d'un littéral est conservé. Cela permet de garantir la cohérence des déductions futures.

- Cette utilisation de contraintes est assez général pour couvrir différentes stratégies. Par exemple, les stratégies basées sur la paramodulation et sur la superposition sont similaires : pour la superposition, les remplacements sont interdits dans les membres droits des équations, alors que pour la paramodulation, ils sont permis dans ces membres droits, mais à chaque déduction le terme remplaçant n'est pas introduit dans la clause déduite ; il est remplacé par une nouvelle variable pour éviter toute inférence dans celui-ci, et une contrainte spécifiant l'égalité entre cette nouvelle variable et le terme remplaçant est ajoutée.

Cependant, l'introduction de contraintes symboliques soulève de nouveaux problèmes.

- Comme nous mélangeons différents types de contraintes, un solveur puissant doit être conçu pour les traiter. Combiner des solveurs pour chaque type de contraintes n'est pas une chose facile.
- Il serait très intéressant de disposer d'un solveur ayant la propriété d'incrémentalité. Le problème est d'optimiser le procédé de résolution de contraintes pour détecter les inconsistances, lorsqu'un nouvel ensemble de contraintes est ajouté à un ensemble satisfaisable. Par exemple, un algorithme propageant les contraintes d'ordre serait nécessaire.
- L'application de règles de simplification est plus complexe. Comme les E-unificateurs ne sont pas calculés, une clause est simplifiable si toutes ses instances sont simplifiables. Mais, ce n'est pas suffisant ; d'autres conditions sur les solutions des contraintes d'unification doivent être vérifiées pour appliquer une simplification. Si nous voulons appliquer la stratégie contrainte pour compléter un ensemble de clauses, les simpifications sont essentielles et une procédure particulière doit être utilisée pour éliminer les instances redondantes.

References

[BGLS92] L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic paramodulation and superposition. In *Proceedings 11th International Conference on Automated Deduction, Saratoga Springs (N.Y., USA)*, pages 462–476, 1992.

[Hul80] J.-M. Hullot. *Compilation de Formes Canoniques dans les Théories équationelles.* Thèse de Doctorat de Troisième Cycle, Université de Paris Sud, Orsay (France), 1980.

[KKR90] C. Kirchner, H. Kirchner, and M. Rusinowitch. Deduction with symbolic constraints. *Revue d'Intelligence Artificielle*, 4(3):9–52, 1990. Special issue on Automatic Deduction.

[NR92a] R. Nieuwenhuis and A. Rubio. Basic superposition is complete. In B. Krieg-Brückner, editor, *Proceedings of ESOP'92*, volume 582 of *Lecture Notes in Computer Science*, pages 371–389. Springer-Verlag, 1992.

[NR92b] R. Nieuwenhuis and A. Rubio. Theorem proving with ordering constrained clauses. In D. Kapur, editor, *Proceedings of CADE-11*, volume 607 of *Lecture Notes in Computer Science*, pages 477–491. Springer-Verlag, 1992.

[NR94] R. Nieuwenhuis and A. Rubio. AC-superposition with constraints: no AC-unifiers needed. In A. Bundy, editor, *Proceedings 12th International Conference on Automated Deduction, Nancy (France)*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 545–559. Springer-Verlag, June 1994.

[PS81] G. Peterson and M. E. Stickel. Complete sets of reductions for some equational theories. *Journal of the ACM*, 28:233–264, 1981.

[Vig94] L. Vigneron. Associative-Commutative Deduction with Constraints. In A. Bundy, editor, *Proceedings 12th International Conference on Automated Deduction, Nancy (France)*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 530–544. Springer-Verlag, June 1994.