Vérification automatique de systèmes à base de règles avec le narrowing stratégique

Anderson Santana de Oliveira[1], David Déharbe[2], Pierre-Etienne Moreau[3]

[1]Departamento de Ciências Exatas e Naturais – Universidade Federal Rural do Semi-Árido(UFERSA) – Mossoró – RN – Brésil

[2]Departmento de Informática e Matemática Aplicada– Universidade Federal do Rio Grande do Norte (UFRN) – Natal– RN – Brésil

[3]INRIA-Nancy-Grand Est— France

Résumé *Nous proposons la construction d'un outil pour la vérification automatique des buts d'atteignabilité sur des systèmes à base de règles contrôlés par une stratégie d'exécution explicite. La procédure de vérification est basée sur le narrowing, qui permet de simuler les exécutions possibles d'un ensemble de règles avec des variables. Le langage TOM sera utilisé pour le développent de cet outil, étant donné ces capacités de réécriture avec des stratégies.*

Abstract *We propose to build a tool aimed to the automated verification of reachability goals over rule-based systems controlled by strategies. The verification procedure is bases on narrowing, which allows one to simulate the possible executions of the rewrite rules with variables. The Tom language will be employed in the development of the tool, given its strategic rewriting capabilities.*

1. Introduction

La réécriture est un formalisme général et expressif pour la spécification et la vérification de divers types de systèmes axiomatisés par des règles de réécriture. La réécriture fournit un style de programmation déclaratif qui peut être utilisé dans la modélisation de la quasi-totalité des applications dans ce domaine grâce aux théories équationnelles et aux stratégies de réécriture ajoutées à la réécriture simple. Ces aspects apparaissent dans la vérification des programmes, mais aussi dans des domaines plus particuliers comme la modélisation de protocoles cryptographiques (Escobar, Meadows and Meseguer 2007) ou des politiques de sécurité (Dougherty, et al. 2007).

Étant donné qu'un ensemble de règles de réécriture définit un système de transitions (dont les états sont des ensembles de termes dans la même classe équationnelle) et que les règles établissent les transitions atomiques entre ces états, il est naturel de poser des questions d'atteignabilité sur ces systèmes. Par exemple, à partir d'un état initial t d'un système, est-il possible d'atteindre une configuration t' qui n'est pas considéré sûre selon une politique de sécurité donnée, c'est-à-dire, si $t \rightarrow^* t'$, quelles que soient les valeurs des variables de t?

À l'origine proposé comme une technique de résolution des buts dans des théories équationnelles ($\exists x.\ (t = t')$), le narrowing a été reconnu comme un mécanisme clé pour répondre à des questions d'atteignabilité, car il fournit les abstractions nécessaires pour simuler les exécutions d'un

ensemble de règles de réécriture. Son avantage par rapport aux techniques de model-cheking traditionnelles est de permettre la vérification des systèmes avec un nombre potentiellement infini d'états.

Notre but est donc de généraliser la technique de résolution basée sur le narrowing, suggéré initialement par (Escobar, Meseguer and Thati 2007) de façon à considérer des systèmes imposant une certaine stratégie d'application des règles. Ceci est important dans la spécification de modèles de sécurité assez courants, comme les systèmes de pare-feu (Kirchner, Kirchner and de Oliveira 2009), qui attribuent des priorités aux règles par exemple. Les stratégies ont été considérées seulement comme une façon d´accélérer la résolution des buts. Dans notre proposition, nous avons une contrainte additionnelle, qui est la stratégie adoptée par le système en train d'être analysé. De cette manière le narrowing stratégique est perçu comme un complément aux approches déjà existantes pour la vérification de modèles avec une infinité d'états (Clarke, Talupur and Veith 2008), telles que les tests d´atteignabilité basés sur les automates d'arbre (Boichut, et al. 2007), et l'approche originale basée sur le narrowing de (Escobar, Meseguer and Thati 2007).

Le produit principal de cette démarche sera un système pour la vérification automatique de programmes à base de règles avec des stratégies, plus particulièrement, nous sommes intéressés par les politiques de sécurité (Cirstea, Moreau and de Oliveira 2009). Ce système sera développé dans le langage Tom (Balland, et al. 2007), qui intègre la réécriture avec des stratégies dans des langages de programmation générales.

2. Le système Tom

Le système Tom fournit un moyen générique d'intégrer les signatures algébriques et le filtrage dans les langages de programmation existants, comme Java, C ou ML. Le langage Tom n'est pas conçu comme un langage à part entière. Rappelant le concept de Domain-Specific Language, sa conception repose sur l'idée d'« îlot formel » : un programme Tom est composé de constructions algébriques nouvelles, correspondant aux notions de signature, de filtrage, de règle, et de stratégie. Ces constructions, imagées par la notion d'îlot, sont de petits espaces isolés dans un ensemble d'une autre nature : le programme écrit dans le langage hôte. Il y a peu de restriction sur la nature du langage hôte et celui-ci peut sans difficulté correspondre aux langages C, C++, C#, Java, Eiffel, Python, Caml, etc.

Tom ajoute principalement deux nouvelles constructions : « `%match` » et « ‘ » (appelé backquote). La première est une extension de switch/case permettant de discriminer sur un terme plutôt que sur des valeurs atomiques comme des entiers ou des caractères. Les motifs sont utilisés pour discriminer et récupérer de l'information dans la structure de données algébrique filtrée. La seconde, inspirée du langage Lisp, permet de construire un terme.

Exemple 1 Un exemple élémentaire pour présenter Tom est la définition de l'addition sur les entiers de Peano, représentés par la constante `Zero()` et le successeur `Suc(Nat)`. En considérant Java comme langage hôte, l'addition peut être définie en Tom de la façon suivante :

```
public class Peano {
   public final static void main(String[] args) {
     Nat two = ‘Suc(Suc(Zero()));
```

```
    System.out.println("2 + 2 = " + plus(two,two));
}

  Nat plus(Nat t1, Nat t2) {
    %match(t1, t2) {
      x, Zero() -> { return `x; }
      x, Suc(y) -> { return `Suc(plus(x,y)); }
    }
  }
}
```

Dans cet exemple, nous distinguons bien les parties îlots introduites par `%match` et « ` », dont la portée correspond à un terme bien formé. Il est à noter qu'un îlot peut contenir des morceaux écrits en langage hôte, appelés « lacs », comme le sont les actions introduites par `-> { ... }`. Ces lacs, pouvant bien sûr contenir des îlots.

La construction « ` » est utilisée pour exprimer la construction de termes de manière algébrique. Elle permet d'utiliser les variables qui sont instanciées par filtrage (comme x et y), des constructeurs algébriques (`Suc`), et des fonctions du langage hôte, telles que plus dans cet exemple. La définition de plus est donnée par filtrage et correspond à une fonction Java, qui peut être utilisée de manière classique, partout ailleurs.

Afin d'être le plus fidèle possible au concept d'îlot formel, le compilateur Tom ne parse pas les parties écrites en langage hôte. Comme l'illustre la Figure 1, seules les constructions ajoutées sont examinées pour être traduites en des instructions du langage hôte. Cette étape de « dissolution » est faite in situ : la traduction se fait en lieu et place des constructions reconnues, sans modifier ni déplacer le code hôte. Cette caractéristique est essentielle pour permettre le déboguage des applications ainsi produites.

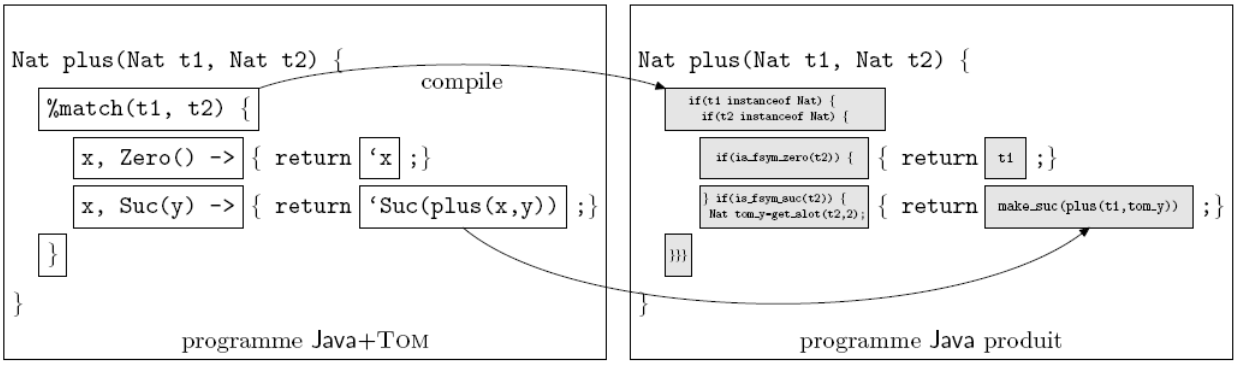

Figure 1 - Compilation de Tom

La troisième contribution, essentielle dans l'environnement Tom, consiste à définir un ensemble minimal d'opérateurs de stratégie, dont la combinaison permet de décrire des parcours et des transformations arbitraires. Le langage proposé autorise des définitions récursives et rend explicite la notion de position dans un terme, permettant ainsi de décrire des stratégies d'exploration non-déterministe. Les stratégies sont réifiées au niveau des termes, permettant de filtrer, de construire, et même de transformer dynamiquement une stratégie. Enfin, une stratégie étant un terme, il est possible d'appliquer une stratégie sur une stratégie. L'ensemble est implanté et intégré à Tom, ce qui rend le langage de stratégie facilement utilisable en présence des toutes les constructions de Tom.

3. Un moteur de narrowing stratégique

Dans toute sa généralité, le narrowing est une méthode de résolution de buts d´atteignabilité correcte mais pas forcément complète. Par contre, dans (Escobar, Meadows and Meseguer 2007) ont montré que le narrowing est une procédure complète dans des cas avec des intérêts assez pratiques, par exemple quand nous cherchons des solutions normalisées.

L'efficacité du narrowing a été améliorée avec l´utilisation de certaines stratégies, surtout dans le sens de trouver des séquences de dérivations optimales où seulement les positions strictement nécessaires sont rétrécies. En général, les résultats de complétude demandent des restrictions assez fortes sur la forme des règles. Par contre, dans (Escobar, Meadows and Meseguer 2007) les auteurs ont montré que ces restrictions peuvent être relaxées dans le cadre de l'analyse d'atteignabilité, car en général nous sommes intéressés par des formes normales seulement.

Nous proposons le développent d'un outil de vérification en utilisant le système TOM. Le système consistera dans un moteur capable d'exécuter des pas de narrowing selon une stratégie donné. Le langage de stratégie de Tom est suffisamment extensible pour comporter la définition de nouvelles situations typiquement issues du narrowing, comme des stratégies pour rendre le processus plus efficace. Néanmoins, ce processus sera contraint par la stratégie sous-jacente au système sous analyse. Nous devrons donc étudier des méthodes pour déterminer quand une stratégie est une sous-stratégie contenant des pas de réécriture strictement compris parmi les dérivations d'une stratégie dominante.

Le développement d'un tel outil devra comprendre la construction d'une interface avec l´utilisateur pour rendre facile l'introduction du modèle à être vérifié aussi bien que les propriétés qui doivent être validés. Tom est bien adapté pour le traitement d'un langage d'entrée que pour la conception de ce moteur de narrowing, étant donné qu'il intègre la réécriture et un langage de stratégie.

La présente proposition fera partie d'une demande de financement en réponse à l'appel Capes-Coffecub qui comprendra la collaboration entre les équipes Pareo et Mosel, de l'INRIA Nancy-Grand-Est et de l'UFRN et de l'UFERSA.

5. Références

Balland, Emilie, Paul Brauner, Radu Kopetz, Pierre-Etienne Moreau, et Antoine Reilles. «Tom: Piggybacking Rewriting on Java.» 2007. 36-47. «Term Rewriting and Applications, 18th International Conference, RTA 2007, Paris, France, June 26-28, 2007, Proceedings.» Springer, 2007.

Boichut, Yohan, Thomas Genet, Thomas P. Jensen, et Luka Le Roux. «Rewriting Approximations for Fast Prototyping of Static Analyzers.» 2007. 48-62. «Term Rewriting and Applications, 18th International Conference, RTA 2007, Paris, France, June 26-28, 2007, Proceedings.» Springer, 2007.

Cirstea, Horatiu, Pierre-Etienne Moreau, et Anderson Santana de Oliveira. «Rewrite Based Specification of Access Control Policies.» *Electronic Notes in Theoretical Computer Science* 234 (2009): 37-54.

Clarke, Edmund M., Muralidhar Talupur, et Helmut Veith. «Proving Ptolemy Right: The Environment Abstraction Framework for Model Checking Concurrent Systems.» 2008. 33-47. «Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings.» Springer, 2008.

Dougherty, Daniel J., Claude Kirchner, Hélène Kirchner, et Anderson Santana de Oliveira. «Modular Access Control Via Strategic Rewriting.» 2007. 578-593. «Computer Security - ESORICS 2007, 12th European Symposium On Research In Computer Security, Dresden, Germany, September 24-26, 2007, Proceedings.» Springer, 2007.

Escobar, Santiago, Catherine Meadows, et José Meseguer. «Equational Cryptographic Reasoning in the Maude-NRL Protocol Analyzer.» *Electr. Notes Theor. Comput. Sci.* 171 (2007): 23-36.

Escobar, Santiago, José Meseguer, et Prasanna Thati. «Narrowing and Rewriting Logic: from Foundations to Applications.» *Electr. Notes Theor. Comput. Sci.* 177 (2007): 5-33.

Kirchner, Claude, Hélène Kirchner, et Anderson Santana de Oliveira. «Analysis of Rewrite-Based Access Control Policies.» *Electronic Notes in Theoretical Computer Science* 234 (2009): 55-75.