Geccoo

Génération de code certifié pour des applications orientées objet

Spécification, raffinement, preuve et détection d'erreurs

http://geccoo.lri.fr

Christine Paulin-Mohring

INRIA Futurs & Université Paris Sud

Réunion finale - 15 janvier 2007 - Orsay

Programme de la journée

13h30 C. Paulin, Présentation générale

14h00 M. Huisman, Langages de haut niveau pour les propriétés de sécurité

14h30 M.-L. Potet, Modularité et Raffinement

15h00 Pause café

15h30 C. Marché, Des modèles adaptés à la génération d'obligations

16h00 S. Ranise, Démonstration automatique d'obligations de preuve

16h30 A. Giorgetti, Détection d'erreurs

Plan

Les équipes

TFC	LIFC, Besançon	F. Bellegarde F. Bouquet/A. Giorgetti
Cassis	LORIA, Nancy projet commun LIFC	S. Ranise
Everest	INRIA-Sophia Antipolis	M. Huisman
ProVal	INRIA Futurs-LRI, Saclay	C. Paulin
VaSCo	LSR, Grenoble	M.-L. Potet

Objectifs

Outils de développement de code orienté objet certifié

- Code critique: cartes à puce (JavaCard), terminaux

Couvrir la chaîne de développement

- Spécifier des propriétés de sécurité, les traduire en propriétés logiques
- Développement modulaire : raffinement et composition
- Génération et résolution automatique des obligations de preuve
- Détection d'erreurs : simulation et test

Approche

Une approche formelle accessible aux programmeurs

- Programmes JAVA annotés en JML.
- Automatisation des preuves.

Des modèles rigoureux de haut niveau pour raisonner

- Systèmes d'évènements, méthode B.
- Assistant de preuve COQ.

Développement d'outils

Collaborations

- 3 réunions pleinières par an
- relations bilatérales
- mobilité de jeunes chercheurs

Application Demoney (1/2)

(Trusted Logic, projet Secsafe)

Propriétés de sécurité :

- Absence d'erreurs d'exécution : division par zéro, déréférencement du pointeur nul, accès en dehors des bornes d'un tableau...
- Authentification : différents niveaux d'identification requis pour chaque opération.
- Enchainement des opérations : l'opération d'initialisation d'une transaction doit être immédiatement suivie de l'opération de complétion de la transation; interruption par arrachage ...

Application Demoney (2/2)

Résultats :

- Spécification en B : messages d'erreur, états des transactions et des canaux de communication. Utilisation de l'outil Génésyst.
- Spécification par raffinement en JML. Utilisation de l'outil JML-TT.
- Spécification et preuve partielle du code en Krakatoa.

Des expériences à analyser, consolider et pérenniser.

Outils

- Des outils multiples, parfois similaires
- Effort d'intégration : génération d'obligations de preuves pour différents prouveurs, transformation des propriétés de sécurité en obligations
- Identification de classes nouvelles de problèmes pour la démonstration automatique : SMT-lib
- Des outils nouveaux pour JML : simulation, test, vérification

Méthode B et OO

- Le modèle B et les outils associés ont servi de support à la mise en place de méthodes et outils pour JML.
  - JAG, JML-TT, JML2B
- Le traitement des invariants avec ownership de spec# permet d'expliquer et généraliser la composition de modules dans B.
- Réflexion sur des spécifications abstraites, raffinement

Conclusion

Résultats :

- Collaborations effectives
- Visibilité : publications, outils, standard SMT-lib, conférences AFADL, B' 2007
- Avancées sur le plan de la gestion de propriétés de sécurité
- Développement d'outils généraux, dépassant le cadre du projet

Futur :

- Propriétés de sécurité
- Invariants
- Modèles abstraits, raffinement, analyse de spécification
- Outils de preuve