DISPO à l'EMN

Hervé Grall, Jacques Noyé, Sebastian Pavel, Jean-Claude Royer
et al.

OBASCO project - EMN/INRIA, LINA

22 septembre 2006

Contexte

- Changement d'échelle du logiciel → nouveaux modes de développement du logiciel
  - Construction d'applications par assemblage/composition de composants
  - Architecture : la structure de l'assemblage (aussi principaux choix qui guident la conception)
- Composant = implémentation + interface
  - Hybride des modules et des classes : existence dynamique/instanciation
  - Interface décrit la structure (signatures) et le comportement (protocoles) des services fournis et requis
  - Et les aspects ?
    $\text{composant}_i + \text{aspect} \rightarrow \text{composant}'_i + \text{composant}_a$
- Lors de l'assemblage des composants, on veut assurer des propriétés de disponibilités des services.

Modèle de composants (niveau spécification formelle)

- Un modèle hiérarchique
- Composant = interface + protocole + description fonctionnelle
- Des protocoles évolués : systèmes de transitions symboliques
- Des communications synchrones et asynchrones
- Une composition expressive
- DOA 2003, papier de revue en cours

Spécification formelle

- Description fonctionnelle sous forme algébrique
- Cohérence avec le STS
- Aide à l'extraction de la spécification
- Technique de preuve (PVS)
- Analyse des communications asynchrones peut aider (simplification, propriétés sur les communications)
- Lien entre logique temporelle (CTL*) et une approche plus classique avec des types de données
- `AGEX true` $\equiv$
  $\forall self : TI, \exists *, D_{TI}(self) => \bigvee_{op_R}(prec_{op_R}(self, *))$
- Passage à l'échelle ?
- IASSE 2004, présentation à WADT 2004

Vérification pour les protocoles

- Difficultés dues aux gardes essentiellement
- Analyse des boîtes à lettre en communication asynchrone
- Abstraction utile pour la vérification : oublier l'ordre des messages (DOA 2004)
- Étude de la *boundedness* des STS
- Algorithme de *boundedness* et notion de décomposition/recomposition sont utiles (FMOODS 2006)
- Modélisation d'exemples (allocateur de ressources, réservation de tickets) avec propriété de disponibilité
- Prototype support pour les STS en Python, réécriture en cours (Java)

Approche par synthèse

- On part d'un composant pour lequel on veut assurer une certaine propriété (exprimée sous la forme d'un protocole = un ensemble de traces)
- On veut générer un composant (moniteur) qui, composé, avec le composant initial fournit un nouveau composant avec la propriété voulue
- Synthèse en 2 temps
  - synthèse du moniteur en tant que protocole de contrôle
  - synthèse de code à partir de ce protocole de contrôle vers une langage de composants

Concrétisation du modèle en un langage de composants

- Java (signature, implémentation)
- langage de description des protocoles, langage de description des connexions
- construction des composants :
  - atomiques : code Java + protocole ou protocole seul (moniteur "adapteur")
  - composites : par connexion de composants existants (compatibilité : absence de deadlock)
- réalisation du produit synchronisé, 2 modèles opérationnels :
  - communications n-m, les décisions sont prises globalement (distribution de la décision ?)
  - communications 1-1, les décisions sont prises localement (cas limite des états "mixtes")
- SC'05

Modèle et langage d'aspects

- composant$_i$ + aspect $\longrightarrow$ composant$'_i$ + composant$_a$
- *stateful* aspect modélisé par un LTS étendu (reconnaissance d'une séquence d'évènement dans le composant + actions)
- automate étendu peut être transformé en un LTS au sens strict
- définit une transformation du LTS du programme de base (instrumentation)
- définition d'un modèle et d'un langage d'aspects avec génération de l'aspect et de l'instrumentation (GPCE'06, avec Rémi Douence, Mario Südholt et Didier Le Botlan)

Synthèse de moniteurs

Le problème : étant donné un protocole, définir un moniteur (qui va jouer le rôle de mandataire) garantissant que tout se passe comme si le client utilisait le composant contrôlé suivant ce protocole.
Généralisation des travaux sur les *security automata* (Schneider) et les *edit automata* (Bauer, Ligatti et Walker) :

- modèle formel général de moniteurs
- définition d'un critère de synthèse
- preuve constructive basée sur 4 mécanismes de contrôle : synchronisation, correction, transaction, anticipation
- Rapport de recherche EMN

Perspectives

- Applications
- Implémentation de la synthèse de moniteurs
- Intégration composants/aspects
- Aspects de disponibilité/tissage de moniteurs
- Distribution du contrôle
- Expression des propriétés de sécurité
- ...