SIMPA Infers Models Pretty Automatically

Démonstration d'outils

Catherine Oriat Roland Groz Emmanuel Perrier

Laboratoire d'Informatique de Grenoble, Équipe VASCO
Université de Grenoble – Alpes

7 juin 2016

Introduction

- Importance des *modèles* en Génie Logiciel (méthodes formelles)
  - Validation : tests basés sur les modèles
  - Vérification de propriétés (preuve, model-checking)
- Problème : modèles absents
  - Développement de code sans modèles ou spécifications
  - Évolution du code $\Rightarrow$ modèles non cohérents avec le code
- $\Rightarrow$ Intérêt de reconstruire des modèles à partir d'un système

Inférence de modèles

Principe

- Génération automatique d'un modèle à partir d'observations (*tests*)
- On fournit des entrées, on observe des sorties

Types d'inférence

- Inférence *passive*
  - un ensemble d'observations est donné au départ
- Inférence *active*
  - les données d'entrées sont choisies au fur et à mesure de l'inférence

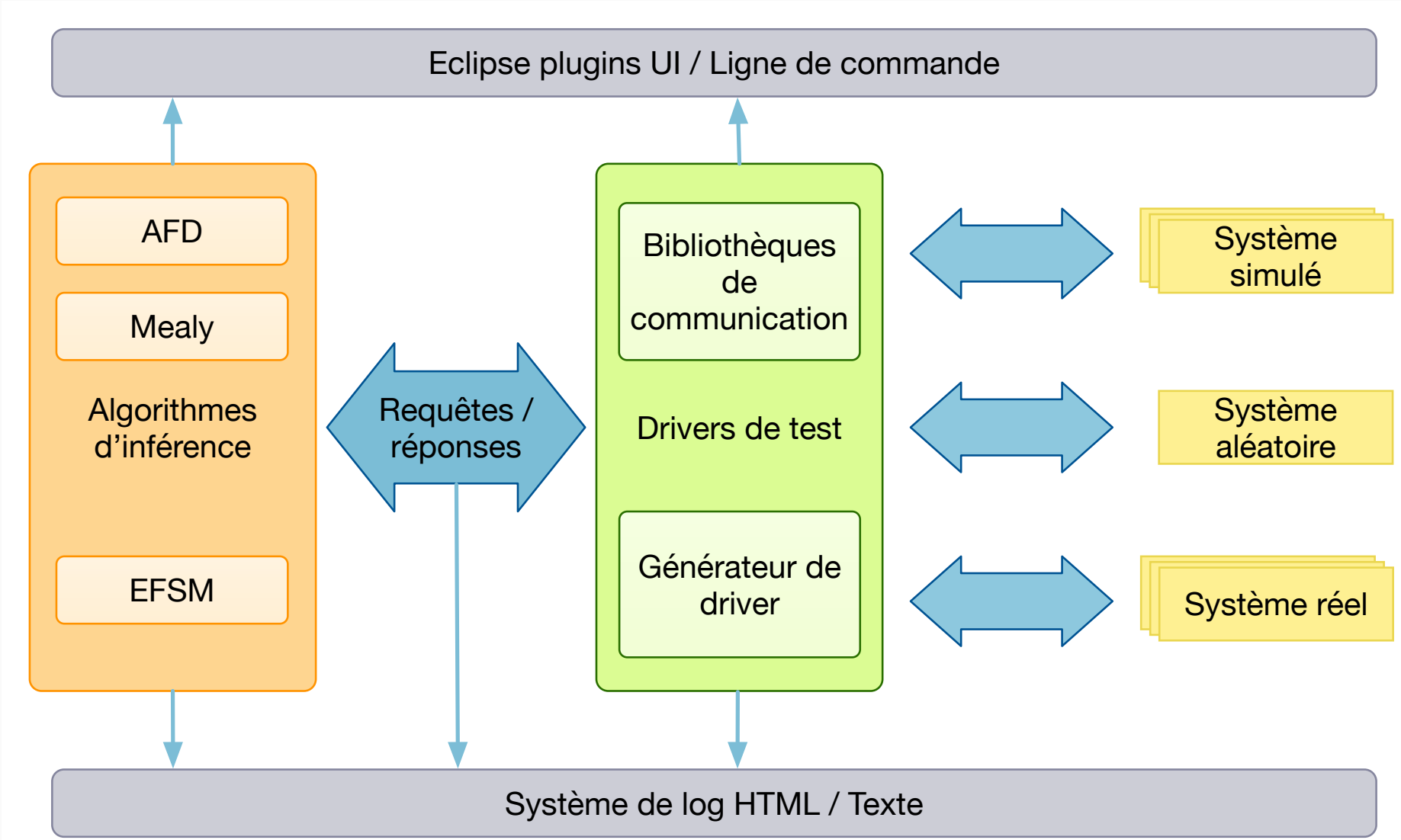
Inférence active d'automates en boîte noire

- Le système est une boîte noire
  - On ne dispose pas du code source, *et pas forcément du code binaire*
  - On ne connait au départ que les entrées possibles
  - Possibilité d'inférer des systèmes à travers un réseau (par exemple des applications web)
- Inférence de la partie *contrôle* du système
  - Abstraction d'une partie des *données*
- Inférence d'*automates*
  - Machines de Mealy (machines d'états finies avec entrées/sorties)
  - Machines d'états finies étendues avec des paramètres, variables et gardes

SIMPA (Simpa Infers Models Pretty Automatically)

- Plateforme d'inférence d'automates
- Développée initialement par K. Hossen dans sa thèse
- Projet européen SPaCIoS : détection de vulnérabilités dans les applications Web

Architecture de SIMPA



Algorithmes d'inférence

- Algorithme fondateur de l'inférence active : algorithme L^* d'Angluin (1987), sur les automates reconnaisseurs, basé sur une représentation des observations sous forme de tables

- Algorithmes implémentés dans Simpa
  - *Lm**, extension de L^* pour les automates de Mealy
  - *Z-Quotient*, basé sur une représentation arborescente des observations, plus efficace que les tables
  - *EFSM* : inférence d'automates étendus, avec paramètres, variables, gardes et prise en compte de nonces et identificateurs de sessions (applications Web)
  - *NoReset*, qui permet l'inférence de systèmes sans réinitialisation

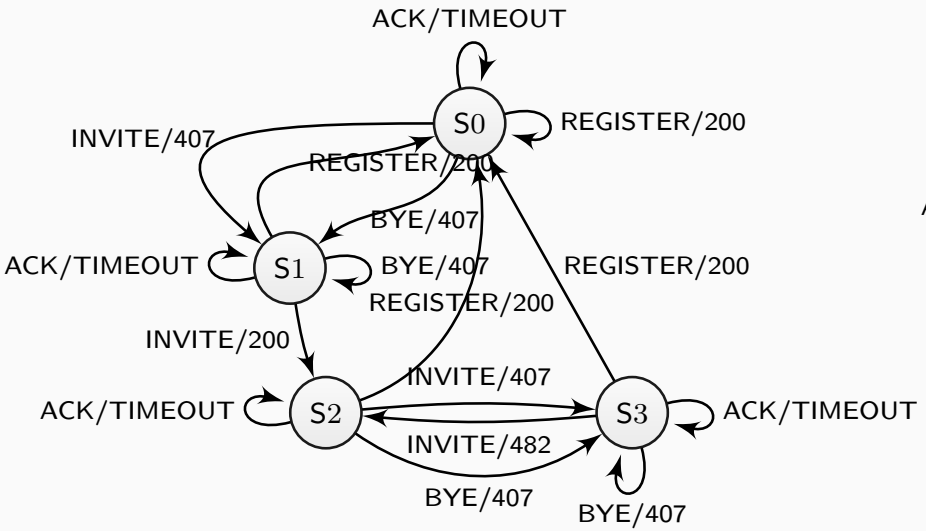
Systèmes inférés et drivers

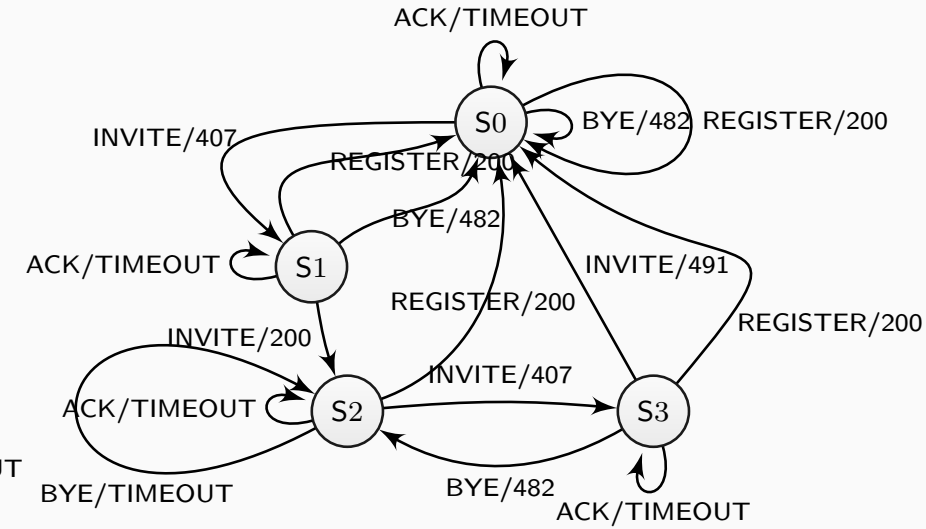
- Systèmes inférés
  - Systèmes *réels*
  - Systèmes *simulés* (tests d'algorithmes)
  - Systèmes *aléatoires* (statistiques)

- Nécessité de drivers (composants d'interfaçage, ou adaptateurs)
  - Cas général : écrits à la main
  - Cas des applications Web : SIMPA génère automatiquement le driver à partir d'un *collecteur* (*crawler*) qui parcourt l'ensemble des pages html et détermine leurs entrées possibles

Exemple : inférence de modèles pour deux serveurs SIP

- SIP (Session Initiation Protocol) : protocole permettant de gérer des sessions multimédias
- Modèles inférés pour deux serveurs SIP : iptel.org et SIP2SIP





- Comportements différents pour *Invite*[4] et *Bye*

Applications

- Générer des modèles
  - pour générer des tests (validation)
  - pour vérifier des propriétés (preuve, model-checking)

- Déterminer le comportement du système
  - détecter des comportements inattendus : acceptés par le modèle mais non autorisés sur l'application (bouton désactivé)

- Réutilisation de composants : vérifier leur interopérabilité
  Test d'intégration

- Détection de vulnérabilités dans les applications Web
  - Cross-Site Scripting (XSS) : réfléchi et persistant
    Injection de contenu malicieux dans une page
  - Cross-Site Request Forgery (CSRF)
    Forcer l'utilisateur à effectuer une action malicieuse, de façon à contourner l'authentification

Outils similaires

- Plate-forme *LearnLib*, dévéloppée à Dortmund, implémente plusieurs algorithmes d'inférence
- Outil *Tomte*, dévéloppé à Nijmegen, permet de générer des drivers entre les algorithmes implémentés par LearnLib et des systèmes réels

Travaux futurs

- Développer de nouveaux algorithmes d'inférence
  - plus efficaces
  - plus généraux (certains algorithmes fonctionnent avec des hypothèses fortes)
- Applications Web
  - traiter des applications utilisant JavaScript (Ajax)
  - améliorer la détection d'attaques XSS, CSRF
  - traiter d'autres types d'attaques (SQL injection)

Questions ?

- Merci de votre attention
- Questions ?