

Event Strictness for Components with Complex Bindings

Fabrício Fernandes¹, Robin Passama²
and Jean-Claude Royer¹

¹École des Mines de Nantes

Computer Science Department – ASCOLA/OBASCO Group
INRIA Research Centre Rennes - Bretagne Atlantique – LINA

²DEMAR Group, UM2 - INRIA

Montpellier Laboratory of Computer Science, Robotics, and Microelectronics
(LIRMM)



25-02-2009 / ISEC09

1 Introduction

Motivation

STS-Oriented Component Model

Examples

2 Checking properties

Compatibility and Communication Checking

Behavioural Compatibility

Event Strictness

3 Final remarks

Motivation

- Component Based Software Engineering (CBSE): get correct and reusable software components
- Need of formal analysis methods to analyze component interactions
- Explicit protocols integrated to component interfaces to describe their behaviour in a formal way
- Problem:
 - Explicit protocols are often dissociated from component code (not ensured that component execution will respect protocols rules)
 - Most of current approaches use binary communications: not enough in some situations

STS-Oriented Component Model

Introduction

Motivation

STS-Oriented
Component Model

Examples

Checking
properties

Final remarks

- Components composition in a non intrusive way
- Components are true black boxes but explicit an interface and a protocol
- Protocol as Symbolic Transition System (STS): a powerful formalism close to statechart or process algebras with values
- Uses the notion of rendezvous: useful to get true black box components
- Provide means to analyse the component assemblies and to check properties

Basic concepts

Introduction

Motivation

STS-Oriented
Component Model

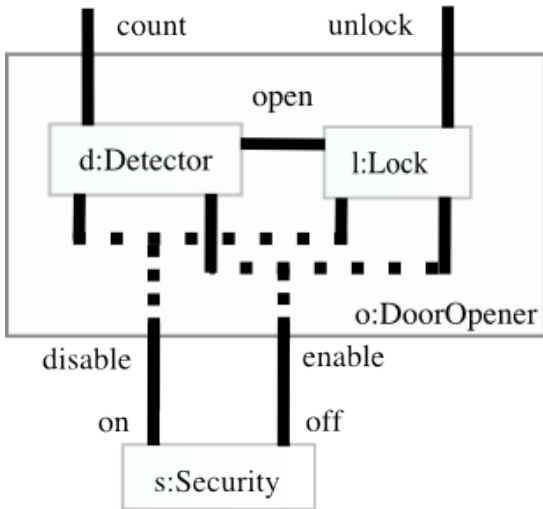
Examples

Checking
properties

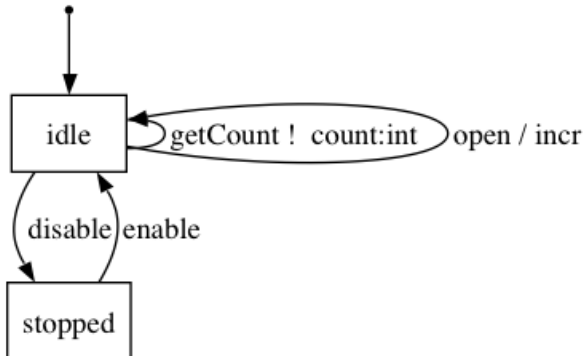
Final remarks

- Primitive component made of ports and a protocol described in the STS formalism
- Composite: reusable composition of components
- STS: states + transitions between states
- STS transition general syntax: [guard] event/action
 - guard: condition to trigger the transition
 - event: dynamic event possibly with emission ! or receipt ? (notification of the action execution)
 - action: action to be performed

Example of Architecture



Detector STS



Component Interactions

Introduction

Motivation
STS-Oriented
Component Model
Examples

Checking
properties

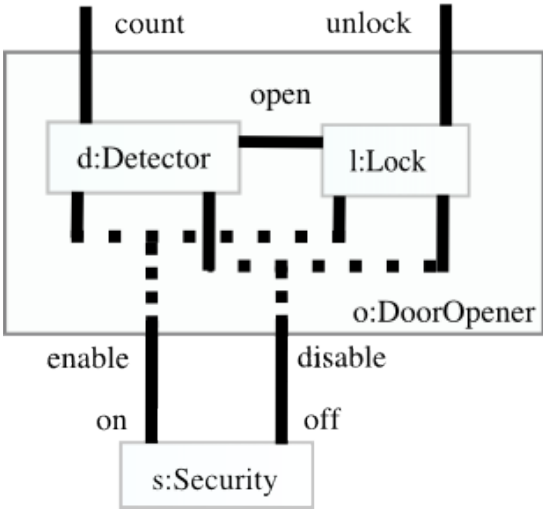
Final remarks

- Ports: connection points on which events are circulating
- Event: name, associated component, if it carries data (emission or receipt) or not, scope
- Events visibility: hidden, visible and exported
- Rendezvous: any number of participants, values communication and each participant can have a specific activity during the synchronisation
- Different types of event export: simple port, duplicated port, branch and merge

Checking assembly properties

- Important to early verify properties on the architectures defined
- Need to formalise these properties
- Behavioural compatibility: checks whether an assembly is deadlock free
- Difficult to check with STS
 - Presence of guards
 - No static criterion in the general case
 - It does not check the relevance of communications
- Problem: communication may not arise while the assembly is compatible
- Event strictness: all communications declared by the designer must appear in the global behaviour

Example of Architecture



Global behaviour for the DoorOpener

Introduction

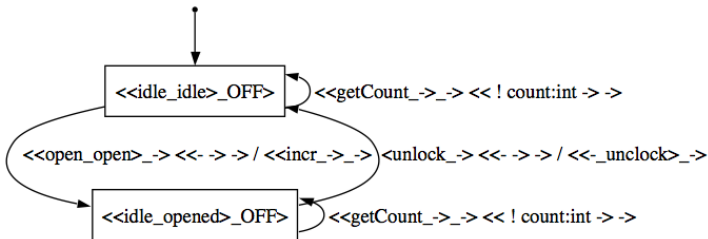
Checking
properties

Compatibility and
Communication
Checking

Behavioural
Compatibility

Event Strictness

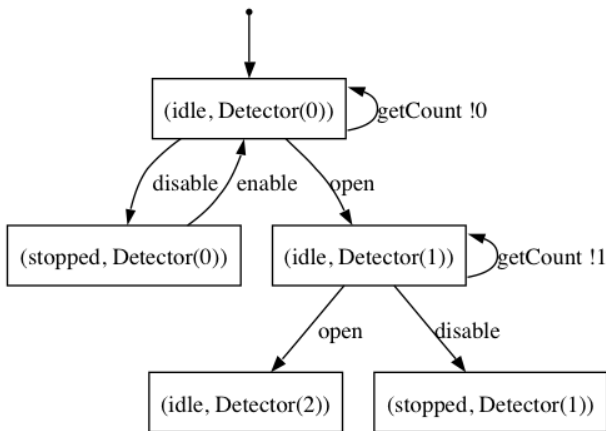
Final remarks



Formalising some concepts

- STS
 - Dynamic behaviour with a data type description
 - Algebraic Data Type for each STS and transitions use operations defined in the ADT
 - Event: set of atomic activities that occur in the components named with labels
- Configuration Graph (CFG)
 - Used to define the semantics of the STS
 - Obtained from the unfolding of communications
 - Particular STS: no receipt, guards equal to true, emissions terms in normal form

Part of the Detector CFG



Formalising some concepts

- Composite STS
 - Assembly of STS including subcomponents, bindings and communications
 - N-party STS: structure of the composite in states and transitions (thus in events, guards, communication offers and actions)
- Structured product
 - Extension of the synchronous product
 - Product defines an STS with tree of states and transitions
 - Synchronization vector: tuple of events denoting a synchronisation between transitions associated to the events

Semantics and verification overview

Event
Strictness for
Components
with Complex
Bindings

Fernandes,
Passama and
Royer

Introduction

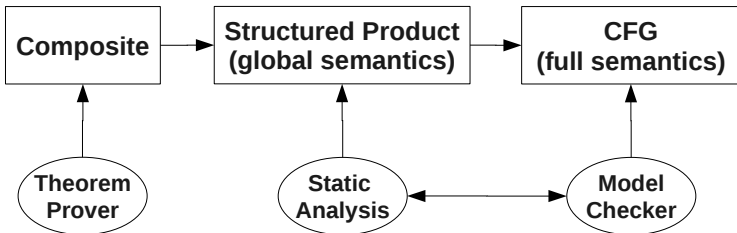
Checking
properties

Compatibility and
Communication
Checking

Behavioural
Compatibility

Event Strictness

Final remarks



Behavioural Compatibility

- **Definition:** a component c is behaviourally compatible with a component assembly if the resulting composite has no deadlock in its behaviour
- Global behavioural compatibility is related to the structured product of the composite viewed as an LTS
- Full behavioural compatibility is based on the configuration graph
- **Theorem:** the full behavioural compatibility is decidable if the CFG is bounded or if component behaviours are finite I/O LTS
- No single relation between the global and the full behavioural compatibility

Event Strictness

Introduction

Checking
properties

Compatibility and
Communication
Checking

Behavioural
Compatibility

Event Strictness

Final remarks

- Model with sophisticated connections and renaming mechanism (behaviour not easy to understand)
- If a user defines some synchronisations in the behaviour, he expects to observe them
- *Event strict* architecture: at each level, the event matchings declared in the communications clauses occur at least once in the behaviour

Event Strictness

Introduction

Checking
properties

Compatibility and
Communication
Checking

Behavioural
Compatibility

Event Strictness

Final remarks

- Two levels: global event strictness and full event strictness
- Global event strictness: same recursive schema as for the structured product
- **Theorem:** full event strictness is decidable if the configuration graph is bounded or in case of finite I/O LTS behaviours
- Contrary to behaviour compatibility, there is a static condition for it
- **Theorem:** if the composite is not globally event strict then it is not fully event strict

Event Strictness Conclusions

- Orthogonal property to behavioural compatibility
- Computing the structured product is a way to early check the errors
- In our context, it is an important property and it must be checked before component compatibility
- Checking is possible because we compute the structured product associated to a composite
- Global event strictness is required to be true
- Full event strictness may be true or false depending on the restrictions applied
 - Designers want to block some synchronisations
 - Synchronisation has to solve a dynamic error which should not occur

Conclusions

- Hierarchical component model based on STS, a nary rendezvous and sophisticated protocols with guards
- Tools to compute the global symbolic protocol associated to the architecture
- Provide a full set of information including the structure of the component and the communications
- Architectural verification: formalisation of communications with complex bindings which provide full component interactions
- Necessary support to analyze the communications and the compatibility of components
- Early properties checking component assemblies: full behavioural compatibility and event strictness

Future work

- Explore more the relation between the full event strictness and other properties
- Define more complete set of verifications
- Improvements on implementation of our STSLib tool

Questions?

- Questions?

Event Strictness for Components with Complex Bindings

Fabrício Fernandes¹, Robin Passama²
and Jean-Claude Royer¹

¹École des Mines de Nantes

Computer Science Department – ASCOLA/OBASCO Group
INRIA Research Centre Rennes - Bretagne Atlantique – LINA

²DEMAR Group, UM2 - INRIA

Montpellier Laboratory of Computer Science, Robotics, and Microelectronics
(LIRMM)



25-02-2009 / ISEC09