Abstract

The poster presents a new verification-oriented automata-based language, Component-Interaction automata, which is designed for specification of component interactions (communication among components) in hierarchical component-based software systems. The model respects an interconnection structure of the system (bindings among components and other specifics), and is close to verification tools.

1 Introduction

- Languages for specification and verification of component interactions in component-based systems typically fall into two classes
  - Architecture description languages, ADLs (Wright, Darwin/Tracta, SOFA)
    - elaborate specification of system architecture
    - verification of only a small fixed set of properties
  - Formal automata-based languages (Interface automata, I/O aut., Team aut.)
    - convenient for formal verification
    - do not respect the architecture of a system

The goal of our research

- To support a component interaction specification and verification process by combining the benefits of ADLs for a specification and formal automata-based models for a verification phase.
- To evolve a new automata-based language supporting this concept.
  - Accept an architecture description as an input
  - Respect it when composing the component automata to build the hierarchy
  - Be close to verification tools

2 CI Automata Language

The Component-Interaction automata language aims to provide a direct and desirable way of modelling interactions in component-based systems which is meant to be transparent and understandable thanks to the primary purpose oriented to component-based systems and their specifics.

Features of CI Automata

- The model is inspired by some features of current automata-based languages and differs in many others.
  - Three types of actions (input, output and internal)
  - Synchronization of one input and one output action with the same name which becomes internal later on (inspired by Interface automata)
  - Freedom of choosing the transition set (inspired by Team automata) — to be adjustable according to the architecture description and other specifics of the system
  - Preservation of important interaction information — to provide a rich base for further verification
    - Components and action involved in each synchronization
    - Hierarchical structure of the system
    - Interaction properties of partial automata (components)

3 CI Automaton

A component-interaction automaton is a tuple $\mathcal{C} = (Q, \ act, \ \delta, \ I, \ S)$:

- $Q$ is a finite set of states, $I$ is a nonempty set of initial states
- $\ act$ is a finite set of actions involved in transition labels (triplets)
  - input label $(\rightarrow, a, B)$ represents that the component $B$ inputs an action $a$
  - output label $(A, a, \rightarrow)$ represents that the component $A$ outputs an action $a$
  - internal label $(\rightarrow, a, \rightarrow)$ represents that $A$ and $B$ synchronize on $a$
- $\delta$ is a finite set of labelled transitions
- $S$ represents a hierarchy of components in $\mathcal{C}$

Example

Let us consider a simple system which consists of one database component (CI automaton $C_1$) and one or more client components ($C_2$, $C_3$). The client component can use $Ins$ and $Del$ services provided by the database component. The database logs events using internal $Log$ service.

4 Composition of CI automata

- CI automata can be composed to form a hierarchical structure.
- Let $S = \{ (Q_i, \ act_i, \delta_i, I_i, S_i) \}_{i \in \mathcal{I}}$ be a system of CI automata.
- Then the component-interaction automaton over $S$ takes form of $\mathcal{C} = (\bigcup_{i \in \mathcal{I}} Q_i, \bigcup_{i \in \mathcal{I}} \ act_i, \bigcup_{i \in \mathcal{I}} \delta_i, \bigcup_{i \in \mathcal{I}} I_i, (S_i)_{i \in \mathcal{I}})$ where the transition set $\delta$ is formed as a subset of a complete transition space $\Delta$ determined by all transitions of the component automata and the transitions representing their synchronizations. The set $\delta$ is then determined by an architecture description and other characteristics of the system.

Example

The component-interaction automaton $C_4$ demonstrates several possibilities of mentioned CI automata composition according to the architecture of the system.

---