Component-Interaction Automata

Barbora Zimmerova

Faculty of Informatics, Masaryk University, Brno
zimmerova@fi.muni.cz

ParaDiSe Seminar, Spring 2006

April 10, 2006

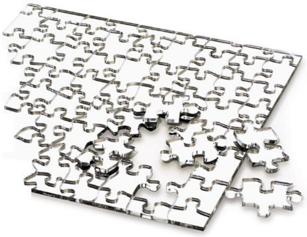
Introduction

Domain: Hierarchical component-based software systems

- Specification of interaction among components
- Verification of interaction properties

Project:

- Verification of Component-Based Systems
  ParaDiSe Laboratory, FI MU, Brno
- Ivana Černá, Luboš Brim,
  Jiří Sochor, Barbora Zimmerova,
  Pavlína Vařeková, Nikola Beneš



Objectives – Verification

Issues of our interest

- Verification of coordination errors
  - Deadlock, computational progress, ...
  - Interaction between specific components
- Reconfiguration correctness
  - Component substitutability
  - Regression verification
- Component-interaction analysis
  - Removal of inactive components
  - Component placement in distributed environment

Objectives – Language

Language for specification of component interactions

1. Flexible – respect various component models
   - Respect the architecture of a system
   - Single/multiple bindings on interfaces
   - Synchronization strategies
2. Capture important information
   - Components – participants of communication
   - Hierarchical structure
3. Be of a manageable complexity
   - To enable automated verification

Specification languages – Overview

Architecture description languages – languages that have been defined within frameworks of architecture description languages

- Tracta
- Wright
- SOFA Behavior protocols

Automata based languages – languages for specification of component interactions that have been defined independently

- I/O automata
- Team automata
- Interface automata

CI automata language

Component-Interaction automata language

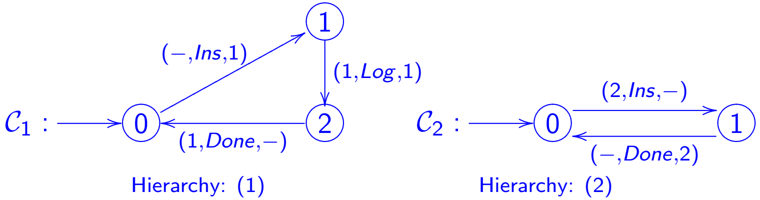
(CI automata for short)

- Automata-based language
  finite state model, infinite executions/traces
- Three types of actions (*input*, *output* and *internal*)
  general used concept
- CCS like synchronization
  one input and one output action which becomes internal later on
- Flexible composition
  can be parametrized by characteristics of a system
- Preservation of important interaction information
  participants of communication, hierarchy

Definition of the CI automaton

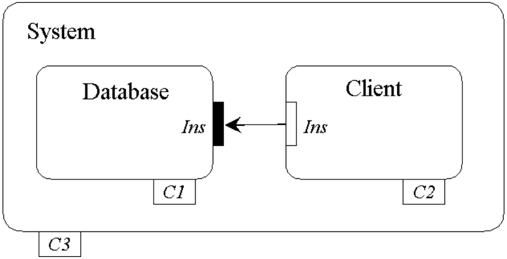
Component-Interaction automaton $\mathcal{C} = (Q, Act, \delta, I, H)$

- States, Initial states
- Actions
- Labeled transitions (structured labels - components, actions)
  - input, output and internal
- Hierarchy

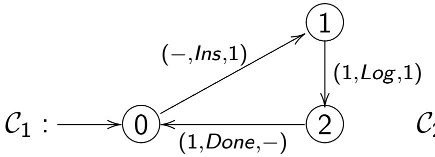


Composition of CI automata 1/2

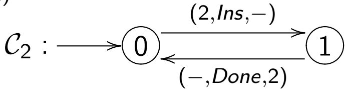
1. Set $\mathcal{S}$ of CI automata to be composed
2. All possible interactions – product automaton with transition set given as a complete transition space $\Delta_{\mathcal{S}}$
3. Feasible interactions – transition set $\delta \subseteq \Delta_{\mathcal{S}}$
4. Composite automaton $\otimes_T \mathcal{S}$ where $\delta = \Delta_{\mathcal{S}} \setminus T$



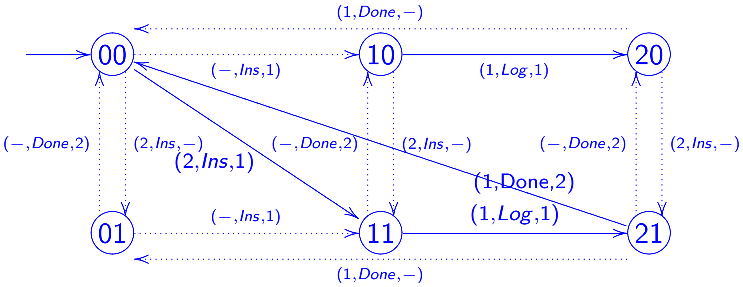
Composition of CI automata 2/2



Hierarchy: (1)



Hierarchy: (2)



Hierarchy: ((1),(2))

In figures, states ij stand for (i,j)

Example – Simple system 1/4

Simple system

- Consist of three components – database $\mathcal{C}_1$, and clients $\mathcal{C}_2$, $\mathcal{C}_3$
- Respect architectural description
- Use one-to-one handshake synchronization

$$\begin{tikzcd}
 & & 1 \arrow[d, "{(1,\mathit{Log},1)}"] \\
\mathcal{C}_1 : \arrow[r] & 0 \arrow[ur, "{(-,\mathit{Ins},1)}"] & 2 \arrow[l, "{(1,\mathit{Done},-)}"]
\end{tikzcd}$$

Hierarchy: (1)

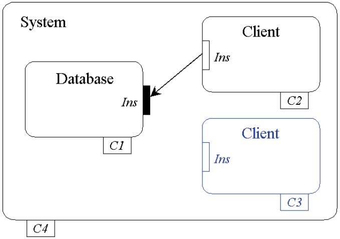
Hierarchy: (2)

$$\begin{tikzcd}
\mathcal{C}_2 : \arrow[r] & 0 \arrow[r, shift left, "{(2,\mathit{Ins},-)}"] & 1 \arrow[l, shift left, "{(-,\mathit{Done},2)}"]
\end{tikzcd}$$

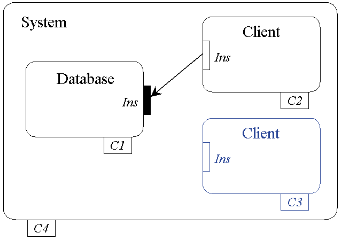
$$\begin{tikzcd}
\mathcal{C}_3 : \arrow[r] & 0 \arrow[r, shift left, "{(3,\mathit{Ins},-)}"] & 1 \arrow[l, shift left, "{(-,\mathit{Done},3)}"]
\end{tikzcd}$$

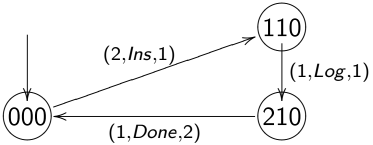
Hierarchy: (3)

Example – Simple system 2/4



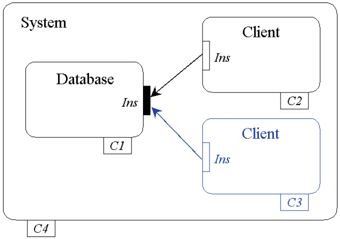
Example – Simple system 2/4



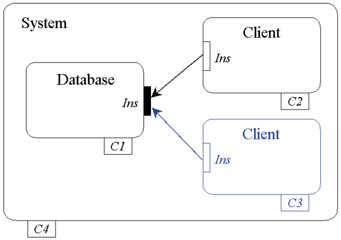


Hierarchy: ((1),(2),(3))

Example – Simple system



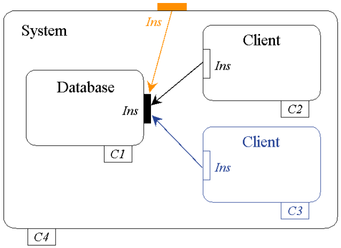
Example – Simple system



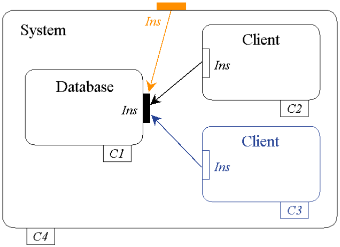
$$\begin{tikzcd}
101 \arrow[d, "{(1,Log,1)}"'] & {} \arrow[d] & 110 \arrow[d, "{(1,Log,1)}"] \\
201 \arrow[r, "{(1,Done,3)}"'] & 000 \arrow[ul, "{(3,Ins,1)}"'] \arrow[ur, "{(2,Ins,1)}"] & 210 \arrow[l, "{(1,Done,2)}"]
\end{tikzcd}$$

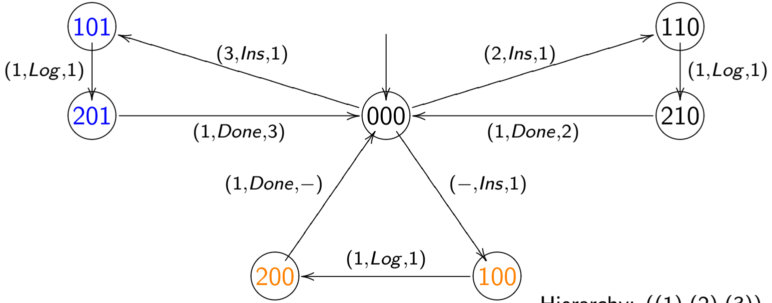
Hierarchy: ((1),(2),(3))

Example – Simple system 4/4



Example – Simple system





Hierarchy: ((1),(2),(3))

Ongoing and future work

Ongoing work

- Composition operator (and others)
- Behavioural equivalencies
- Temporal logic
- CI automata $\longrightarrow$ DiVinE input language

Future work

- Verification of coordination errors
- Reconfiguration correctness
- Component-interaction analysis

Thank you

Thank you for your attention