On the Observable Behaviour of Composite Components

Rolf Hennicker\textsuperscript{1}, Stephan Janisch\textsuperscript{1} and Alexander Knapp\textsuperscript{2}

\textsuperscript{1}Ludwig-Maximilians-Universität München
\textsuperscript{2}Universität Augsburg
Outline

Component Model with Ports
   Specified and derived behaviours
   Component correctness

Efficient Computation of Observable Behaviours
   Behaviourally neutral components
   Reduction algorithm
   On the role of ports
Metamodel for static structures
(c.f. ports in ROOM (Selic et al. 1994), UML RT (Selic and Rumbaugh 1998))

(Hennicker et al. 2008)
Compressing Proxy Server
UML2 structured classifier specify composite components

- Port and component declarations (name, type)
- Messages sent/received via ports (req/prov interfaces)

(Hennicker et al. 2008)
On the Observable Behaviour of Composite Components
Metamodel with specified and derived behaviours
Semantic domain for behaviours (I/O-TS)
(c.f. interface automata with product in de Alfaro and Henzinger 2001, 2005)

An I/O-transition system $A = (L, Q, q_0, \Delta)$ is given by

- I/O-labelling $L = (I, O, T)$,
- set of states $Q$, initial state $q_0$,
- transition relation $\Delta \subseteq Q \times (I \cup O \cup T \cup \{\tau\}) \times Q$

Operators

- **Hiding** $A/\mathcal{H}$ removes $\mathcal{H}$ from $L$, $\Delta$ with $\tau$-moves (silent)
- **Relabeling** $\rho : L \rightarrow L'$ renaming
- **Composition** $A_1 \otimes A_2$ like standard-|| using
  - shared labels $(I_1 \cap O_2) \cup (I_2 \cap O_1)$
  - product labelling $L_1 \otimes L_2$ moves shared labels to $T$ (internalise)
Port protocols and component behaviours

- **Port protocol**
  \[
  \text{prot}(P) = ((I, O, \emptyset), Q, q_0, \Delta)
  \]
  \[m_1 \in I \quad \text{message } m_1 \text{ received}\]
  \[m_2 \in O \quad \text{message } m_2 \text{ sent}\]

- **Component behaviour**
  \[
  \text{beh}(C) = ((I, O, T'), Q, q_0, \Delta)
  \]
  \[p.m_1 \in I \quad \text{message } m_1 \text{ received via port } p\]
  \[p.m_2 \in O \quad \text{message } m_2 \text{ sent via port } p\]
  \[t \in T \quad \text{internal label}\]
- Port protocol of TxtCompr
  \[ \text{prot}(\text{TxtCompr}) \]

- Internal behaviour specification of Adaptor
  \[ \text{beh}(\text{Adaptor}) \]
Observable behaviour of simple components

- given: component \( C \) with \( \text{beh}(C) = ((I, O, T), Q, q_0, \Delta) \)

- \( obs(C) = \text{beh}(C)/T \) hide internal labels of \( C \)!

Example (Adaptor)

\( obs(\text{Adaptor}) = \text{beh}(\text{Adaptor})/\{\text{njpg}\} \)
Assembly behaviour

- given: \( obs(C_1), \ldots, obs(C_n) \)
- \( beh(A) = c_1.obs(C_1)\sigma_1 \otimes \cdots \otimes c_n.obs(C_n)\sigma_n \)
- where \( \sigma_i \) renames \( c_i.p.m \) to \( con.m \)

(Hennicker et al. 2008)
Example (Compressing proxy assembly)

adapt.\textit{obs}(\textit{Adaptor})\sigma_1 \otimes \textit{gzip.\textit{obs}(\textit{GZip})}\sigma_2 \otimes \textit{gif.\textit{obs}(\textit{GifToJpg})}\sigma_3

30 reachable states, 65 transitions

(Hennicker et al. 2008) On the Observable Behaviour of Composite Components
Observable behaviour of composite components

- **given:** $beh(A)$
- $obs(CC') = (beh(A)\rho_{relay})/T$

(Hennicker et al. 2008)
Example (Observable behaviour of CompressingProxy)

\[ \text{obs(CompressingProxy)} \text{ (after minimisation)} \]
Correct components
Using observational equivalence (weak bisimulation)

Component behaviour at port $p$ (of type $P$)

- $\text{obs}_p(C')$ from $\text{obs}(C)$ by hiding all but $p$-prefixed labels

$C$ correct w.r.t. $p$ (of type $P$) if

- $\text{obs}_p(C') \approx p.\text{prot}(P)$

$C$ is correct if it is correct w.r.t. all of its ports

(Hennicker et al. 2008)
Example: Adaptor Correctness
Adaptor is correct w.r.t. TxtCompr

\( \text{prot}(\text{TxtCompr}) \)

\( \text{obs(Adaptor)} \)

(Hennicker et al. 2008) On the Observable Behaviour of Composite Components
Adaptor is correct w.r.t. TxtCompr
Prefix and projection according to port declaration t:TxtCompr

t_prot(TxtCompr)

\(obst(\text{Adaptor})\) (after minimisation)
Efficient Computation of Observable Behaviours

Outline

Component Model with Ports
Specified and derived behaviours
Component correctness

Efficient Computation of Observable Behaviours

Behavourially neutral components
Reduction algorithm
On the role of ports

\[
beh(A) = c_1 \cdot obs(C_1) \sigma_1 \otimes \cdots \otimes c_n \cdot obs(C_n) \sigma_n \\
obs(CC) = (beh(A) \rho_{\text{relay}}) / T
\]

How to compute observable behaviours more efficiently?

(Hennicker et al. 2008)
Efficient Computation of Observable Behaviours

Reduction strategy

Observation: Several components within the assembly may not play any role for the observable behaviour of $CC$.

Goal: Reduce the original assembly to a simpler assembly such that both composite components have the same observable behaviour.

(Hennicker et al. 2008)
Neutrality for I/O-transition systems


Let $A$ and $B$ be I/O-transition systems

$B$ is neutral for $A$ if its composition does not restrict $A$:

“$A \otimes B \approx A$”

More precisely

$$A \otimes B \approx A \Theta_s$$

where

$\Theta_s$ internalises shared labels according to $\otimes$. 

(Hennicker et al. 2008)
Behavourially neutral components

$D$ is behaviourally neutral for $C$ if

$$c.\text{obs}(C)\sigma_1 \otimes d.\text{obs}(D)\sigma_2 \approx c.\text{obs}(C)\Theta_S$$

(Hennicker et al. 2008)
Reduction algorithm

```
reduce : Assembly → Assembly
while neutralleaves(A) ≠ ∅
do  d : D ← choose from neutralleaves(a)
  let A = ⟨c : C, d : D, C; k : (c.p : P, d.q : Q), K⟩
  with d : D neutral at q for c : C at p
  A ← ⟨c : C \ (p : P), C; K⟩ od
```

Theorem

```
obs(⟨A, RelayPorts, DelegateConnectors⟩) ≈
obsp(⟨reduce(A), RelayPorts, DelegateConnectors⟩)
```

(Hennicker et al. 2008)
On the role of ports

Neutrality-checks involve products of component behaviours

Goal:

Reduce neutrality-checks for components to neutrality-checks for ports!

(Hennicker et al. 2008)
**Weak determinism**  
(c.f. Milner 1989)

An I/O-transition system $A$ is **weakly deterministic** if for all weak traces $\lambda$ of $A$

$$ q_0 \overset{\lambda}{\rightarrow} q_1 \text{ and } q_0 \overset{\lambda}{\rightarrow} q_2 $$

then $q_1 \approx q_2$

**Proposition**

$A$ is weakly deterministic iff there is a strongly deterministic, $\tau$-free I/O-transition system $B$ such that $A \approx B$.

(Hennicker et al. 2008)
Weak determinism and neutrality

Proposition

Let $A, B, C$ be I/O-transition systems with labelings $L(A) = L(B) \subseteq L(C)$,

Let $C/H \approx A$ where $H = L(C) \setminus L(A)$.

If $B$ is weakly deterministic and
if $B$ is neutral for $A$
then $B$ is neutral for $C$.

(Hennicker et al. 2008)
Behavourially neutral components (revisited)

\[ D \text{ is behaviourally neutral for correct } C \text{ if } \text{prot}(Q) \text{ is weakly deterministic and behaviourally neutral for } \text{prot}(P). \]
Efficient Computation of Observable Behaviours

On the role of ports

Reduction algorithm (reconsidered)

\[
\text{reduce} : \text{Assembly} \rightarrow \text{Assembly}
\]

while neutralleaves(A) \neq \emptyset

\[
d \leftarrow \text{choose from neutralleaves}(A)
\]

let \[
A = \langle c : C, d : D, C; k : (c.p : P, d.q : Q), K \rangle
\]

with \[d : D \text{ neutral at } q \text{ for } c : C \text{ at } p\]

\[
A \leftarrow \langle c : C \setminus (p : P), C; K \rangle \circ d
\]

Theorem

\[
\text{obs}(\langle A, \text{RelayPorts}, \text{DelegateConnectors} \rangle) \approx \text{obs}(\langle \text{reduce}(A), \text{RelayPorts}, \text{DelegateConnectors} \rangle)
\]

(Hennicker et al. 2008)
Observable behaviour of CompressingProxy

\[ \begin{align*}
\text{obs(CompressingProxy)} &= \text{beh(Assembly)} \rho_{\text{relay}} / T \\
\text{Assembly} \quad \text{obs(Adaptor)} \otimes \text{obs(GZip)} \otimes \text{obs(GifToJpg)} \\
&\text{[30 states, 65 transitions]}
\end{align*} \]

(Hennicker et al. 2008)
1. The three subcomponents are correct.
2. \( \text{prot}(\text{Zip}) \) is weakly deterministic and neutral for \( \text{prot}(\text{TxtCompr}) \).
3. \( \text{GifCompr} \) is neutral for \( \text{Adaptor} \setminus \{t:\text{TxtCompr}\} \) [12 states, 23 transitions]
4. \( \text{obs}(\text{CompressingProxy}) \approx \text{obs}(\text{Adaptor} \setminus \{t, g\})\rho_{\text{relay}} \)
Further topologies
(c.f. arbitrary topologies in Aldini and Bernardo 2005)

- Relay ports to different subcomponents

- Topologies containing cycles

(Hennicker et al. 2008)
Conclusion

Summary

Part I: Metamodel and behaviours

- Structural properties
- Specified vs derived behaviours
- Hierarchical composition

Part II: Efficient computation of observable behaviours

- Neutrality for I/O-transition systems and components
- Use port protocols for neutrality-checks
Next plans

- Frame protocols: specification of observable behaviours
- Apply results to a setting with asynchronous communications
- Integrate data (invariants, pre/postconditions)
- Tools (Java/A, ...)

(Hennicker et al. 2008)
Thank you!
Adaptor port GifCompr