Refinement of Interface Automata Strengthened by Action Semantics

Sebti Mouelhi, Samir Chouali, Hassan Mountassir
{smouelhi,chouali,mountassir}@lifc.univ-fcomte.fr

Laboratoire d'Informatique de l'université de Franche Comté - LIFC EA 4157
16, route de Gray - 25030 Besançon cedex, France

FESCA09 March, 28 2009
Outline

1. Focus
2. Our interface automata based approach
3. Redefinition of the refinement
4. The CyCab case study
Component interface formalisms

- formal models to specify component-based systems.
- a communicating interface describes how a component can be composed and connected to the others in a system design.
- enough information ensuring a proper components working together properly.
- common used interface formalisms: I/O automata, IA, Coln automata ...

Interface automata IA:
- the same syntax as I/O automata.
- not necessarily input-enabled.
- poor models at the level of action semantics.
- no enough information about the synchronization of shared actions.

⇒ No enough information to verify component interoperability
Our approach: Informally

- **An IA based approach:**
  - Interface automata
  - considering only action signatures.

- **Contributions:** extending action by their semantics:
  - actions + predicates over a set of shared variables.
  - annotate transitions by Pre and Post conditions of actions.

  ⇒ More reliable verification of component interoperability
Let $A = \langle S_A, I_A, \Sigma^I_A, \Sigma^O_A, \Sigma^H_A, \text{Pre}_A, \text{Post}_A, \delta_A \rangle$ be an IA strengthened by action semantics where:

- a finite set $S_A$ of states;
- an initial state $I_A \subseteq S_A$;
- three disjoint sets $\Sigma^I_A, \Sigma^O_A$ and $\Sigma^H_A$ of inputs, output, and hidden actions;
- $\text{Pre}_A$ and $\text{Post}_A$ are the set of pre and post-conditions of actions, they are atomic formulae over the set of variables $V$;
- a set $\delta_A \subseteq S_A \times \text{Pre}_A \times \Sigma_A \times \text{Post}_A \times S_A$ of transitions.
Refinement

- formalizing the relation between abstract and concrete versions of the same component interface.
- the refined specification must allow more legal inputs and fewer outputs than the abstract specification.
- the refinement relation is defined as a simulation.
- $Q \preceq P$:
  - the input steps of $P$ are simulated by $Q$.
  - the output steps of $Q$ are simulated by $P$.

Refinement $\triangleq$ Alternating simulation
Alternating simulation

Definition (ε-closure(v))

Given an interface automaton P and a state v ∈ V_P, ε-closure_P(v) is the smallest U ⊆ V_P such that (1) v ∈ U and (2) if u ∈ U and (u,a,u') ∈ V_P^H, then u' ∈ U.

Definition (Externally enabled actions of a state v)

Consider an interface automaton P and a state v ∈ V_P,

- ExtEn^O_P(v) = \{a | ∃ u ∈ ε-closure(v). a ∈ A^O_P(u)\}
- ExtEn^I_P(v) = \{a | ∀ u ∈ ε-closure(v). a ∈ A^I_P(u)\}

Definition (Externally reachable states from v and an ExtEnA a)

- ExtDest_P(v, a) = \{u' | ∃(u, a, u') ∈ τ_P. u ∈ ε-closure(v)\}
A binary relation \( \preceq \subseteq S_P \times S_Q \) from Q to P is an alternating simulation if for all \( s \in S_P, r \in S_Q \) such that \( r \preceq s \) the following conditions holds

1. \( \text{ExtEn}_P^I(s) \subseteq \text{ExtEn}_Q^I(r) \);
2. \( \text{ExtEn}_Q^O(r) \subseteq \text{ExtEn}_P^O(s) \);
3. \( \forall a \in \text{ExtEn}_P^I(s) \cup \text{ExtEn}_Q^O(r) \) and \( \forall r' \in \text{ExtDest}_Q(r, a) \): \( \exists s' \in \text{ExtDest}_P(s, a) \) such that \( r' \preceq s' \) and
   - if \( a \in \text{ExtEn}_P^I(s) \) then \( \text{Pre}_P,a \Rightarrow \text{Pre}_Q,a \) and \( \text{Post}_Q,a \Rightarrow \text{Post}_P,a \).
   - else if \( a \in \text{ExtEn}_Q^O(r) \) then \( \text{Pre}_P,a \Leftrightarrow \text{Pre}_Q,a \) and \( \text{Post}_P,a \Leftrightarrow \text{Post}_Q,a \).

over the set of variables \( V' \).
Intuitively

- **Input actions (provided services)**
  - add more provided services
  - strengthen their former operation: add constraints on their pre and post-conditions.
    - fewer precondition
    - stronger postcondition

- **Output actions (required services)**
  - refinement contains less output actions than the abstraction
  - constraints still unchanged in the refinement
    - pre and post-conditions of a remaining output action are equivalent to their correspondents in the abstract one.
Alternating simulation

\[ u \xrightarrow{\preceq} v \]
\[ \iff \]
\[ \text{Input actions} \]

\[ v \xrightarrow{\preceq} v' \]
\[ \iff \]
\[ \text{Input actions} \]
The refinement definition

Definition (Refinement)

The interface automaton $Q$ refines the interface automaton $P$, written $Q \preceq P$ according to the set of variables $V'$ if

- $\Sigma^I_P \subseteq \Sigma^I_Q$ and $\Sigma^O_P \supseteq \Sigma^O_Q$;
- there is an alternating simulation $\preceq$ from $Q$ to $P$ such that $I_Q \preceq I_P$. 
Refinement IA A.Semantics

Redefinition of the refinement

Substitution

Env: \( x \xrightarrow{\text{Pre}_{\text{Env}, a, a!}, \text{Post}_{\text{Env}, a}} y \)

\( x \xrightarrow{\text{Pre}_{\text{Env}, a, a?}, \text{Post}_{\text{Env}, a}} y \)

P: \( 1 \xrightarrow{\text{Pre}_{\text{P}, a, a?}, \text{Post}_{\text{P}, a}} 2 \)

\( 1 \xrightarrow{\text{Pre}_{\text{P}, a, a!}, \text{Post}_{\text{P}, a}} 2 \)

Q: \( 1' \xrightarrow{\text{Pre}_{\text{Q}, a, a?}, \text{Post}_{\text{Q}, a}} 2' \)

\( 1' \xrightarrow{\text{Pre}_{\text{Q}, a, a!}, \text{Post}_{\text{Q}, a}} 2' \)
Theorems

**Theorem (transitivity)**

For all interface automata $P$, $Q$, and $R$, if $P \preceq Q$ and $Q \preceq R$, then $P \preceq R$.

**Theorem (substitution)**

Consider three interface automata $P$, $Q$, and $R$ such that $Q$ and $R$ are composable and $\Sigma_I^Q \cap \Sigma_O^R \subseteq \Sigma_I^P \cap \Sigma_O^R$. If $P$ and $R$ are compatible and $Q \preceq P$, then $Q$ and $R$ are compatible and $Q \parallel R \preceq P \parallel R$. 
Theorems

Corollary

Consider four automata $P$, $Q$, $R$, and $S$ such that

- $Q$ and $R$ are composable;
- $\Sigma^Q_I \cap \Sigma^R_O \subseteq \Sigma^P_I \cap \Sigma^R_O$;
- $S$ and $Q$ are composable;
- $\Sigma^S_I \cap \Sigma^Q_O \subseteq \Sigma^R_I \cap \Sigma^Q_O$;

If $P$ and $R$ are compatible, $Q \preceq P$, and $S \preceq R$ then $Q$ is compatible with $R$ and $S$ is compatible with $Q$ and $Q\parallel S \preceq P\parallel R$. 
Components

Vehicle

position!

halt?

far?

emergency?

reset!

start?

Station

position?

halt!

far!

emergency!

reset?

Emergency Halt

Starter

start!

start?
Components

Vehicle
- position!
- halt?
- far?
- emergency?
- reset!
- start?

Starter
- start!

Station
- position?
- halt!
- far!
- emergency!
- reset?

Emergency Halt
The Vehicle and Station interfaces

- **VPrS**: start?
- **VPrP**: pos!
- **VPrF**: far?
- **VPrH**: halt?
- **VPrE**: emrg?
- **SPrP**: pos?
- **SPrF**: far!
- **SPrH**: halt!
- **SPrE**: reset!

Diagram showing the transitions between states for the vehicle and station interfaces.

**Sebti Mouelhi (LIFC)**
Refinement of the Vehicle interface

Vehicle'

- fstart?
- pos!
- halt?
- far?
- emrg?
- reset!

Start?

1

2

3

4

5

6

fstart
start
emrg
far
halt
pos
reset
move;
stop;
reset!
emrg?

halt?
start?
emrg?
far?
pos!
emrg?
emrg?
That’s all, thanks for your attention
References I

L. Alfaro and T. Henzinger.
Interface automata.

S. Chouali, H. Mountassir and S. Mouelhi.
An I/O automata based approach to verify component compatibility: application to the CyCab car.
ENTCS, 2008.

B. Gérard, G. Philippe, M. Hervé and P. Gibollet
The INRIA Rhône Alpes Cycab.