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

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, CoIn 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.

$\Rightarrow$ 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.

$\Rightarrow$ More reliable verification of component interoperability

Formal definition

Definition

Let $A = \langle S_A, I_A, \Sigma_A^I, \Sigma_A^O, \Sigma_A^H, Pre_A, Post_A, \delta_A \rangle$ *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* Σ_A^I, Σ_A^O *and* Σ_A^H *of inputs, output, and hidden actions;*
- Pre_A *and* $Post_A$ *are the set of pre and post-conditions of actions, they are atomic formulae over the set of variables V;*
- *a set* $\delta_A \subseteq S_A \times Pre_A \times \Sigma_A \times Post_A \times S_A$ *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 $v \in V_P$, ε-closure$_P(v)$ is the smallest $U \subseteq V_P$ such that (1) $v \in U$ and (2) if $u \in U$ and $(u,a,u') \in {V_P}^H$, then $u' \in U$

Definition (Externally enabled actions of a state v)

Consider an interface automaton P and a state $v \in V_P$,

- $ExtEn^O_P(v) = \{a \mid \exists\ u \in \varepsilon\text{-}closure(v).a \in A^O_P(u)\}$
- $ExtEn^I_P(v) = \{a \mid \forall\ u \in \varepsilon\text{-}closure(v).a \in A^I_P(u)\}$

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

- $ExtDest_P(v,a) = \{u' \mid \exists (u,a,u') \in \tau_P . u \in \varepsilon\text{-}closure(v)\}$

Alternating simulation

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. $ExtEn^I_P(s) \subseteq ExtEn^I_Q(r)$;
2. $ExtEn^O_Q(r) \subseteq ExtEn^O_P(s)$;
3. $\forall\ a \in ExtEn^I_P(s) \cup ExtEn^O_Q(r)$ *and* $\forall\ r' \in ExtDest_Q(r, a)$: $\exists\ s' \in ExtDest_P(s, a)$ *such that* $r' \preceq s'$ *and*
    - *if* $a \in ExtEn^I_P(s)$ *then* $Pre_{P,a} \Rightarrow Pre_{Q,a}$ *and* $Post_{Q,a} \Rightarrow Post_{P,a}$.
    - *else if* $a \in ExtEn^O_Q(r)$ *then* $Pre_{P,a} \Leftrightarrow Pre_{Q,a}$ *and* $Post_{P,a} \Leftrightarrow Post_{Q,a}$.

   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

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
u \arrow[r, "\preceq"] \arrow[d, "a?"] \arrow[dr, phantom, "\mathit{Pre}_{Q,a} \Leftarrow \mathit{Pre}_{P,a}", pos=0.3] \arrow[dr, phantom, "\mathit{Post}_{Q,a} \Rightarrow \mathit{Post}_{P,a}", pos=0.7] & v \arrow[d, "a?"] \\
u' \arrow[r, dashed, "\preceq"] & v'
\end{tikzcd}
\qquad
\begin{tikzcd}[column sep=huge, row sep=huge]
u \arrow[r, "\preceq"] \arrow[d, "a!"] \arrow[dr, phantom, "\mathit{Pre}_{Q,a} \Leftrightarrow \mathit{Pre}_{P,a}", pos=0.3] \arrow[dr, phantom, "\mathit{Post}_{Q,a} \Leftrightarrow \mathit{Post}_{P,a}", pos=0.7] & v \arrow[d, "a!"] \\
u' \arrow[r, dashed, "\preceq"] & v'
\end{tikzcd}
$$

Input actions | Input actions

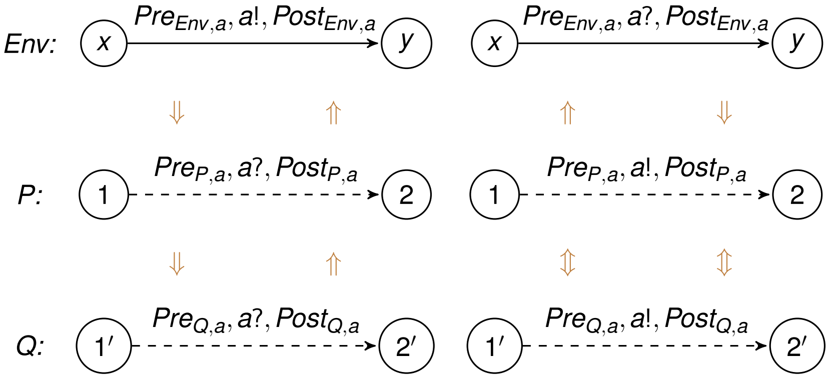
The refinement definition

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* $I_Q \preceq I_P$.

Substitution



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 $Q \| R \preceq P \| R$.

Theorems

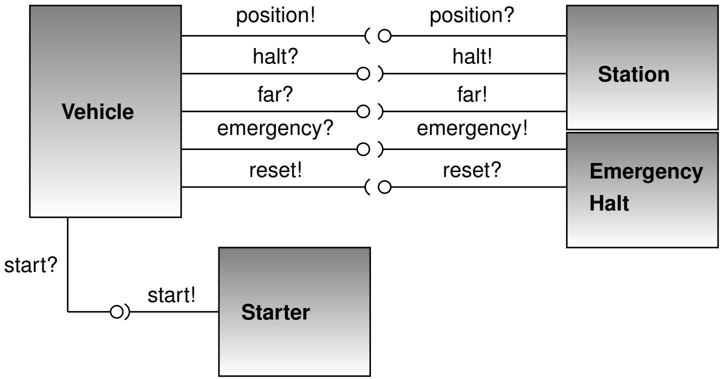
Corollary

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

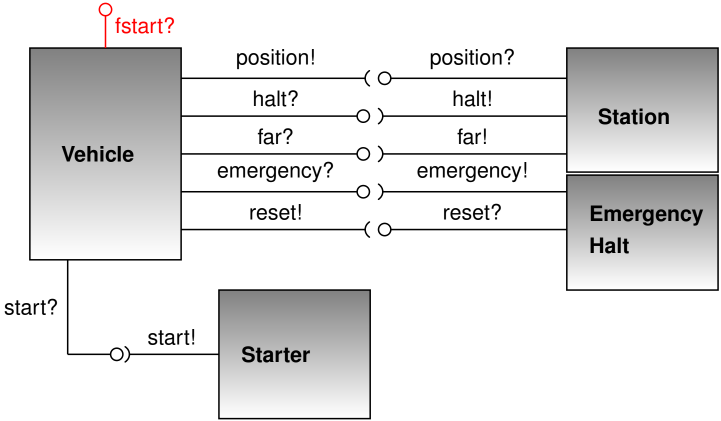
- *Q and R are composable;*
- $\Sigma_I^Q \cap \Sigma_O^R \subseteq \Sigma_I^P \cap \Sigma_O^R$;
- *S and Q are composable;*
- $\Sigma_I^S \cap \Sigma_O^Q \subseteq \Sigma_I^R \cap \Sigma_O^Q$;

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 $Q \| S \preceq P \| R$.

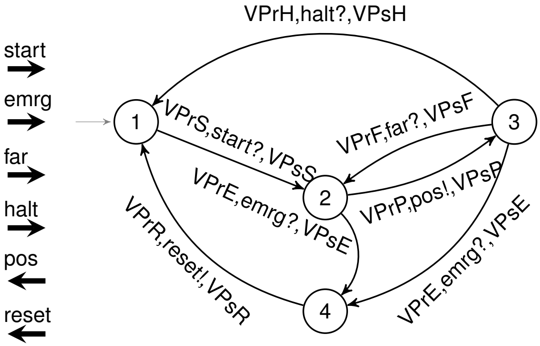
Components

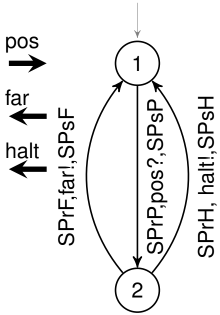

Vehicle
Station
Emergency Halt
Starter
position!
position?
halt?
halt!
far?
far!
emergency?
emergency!
reset!
reset?
start?
start!

Components

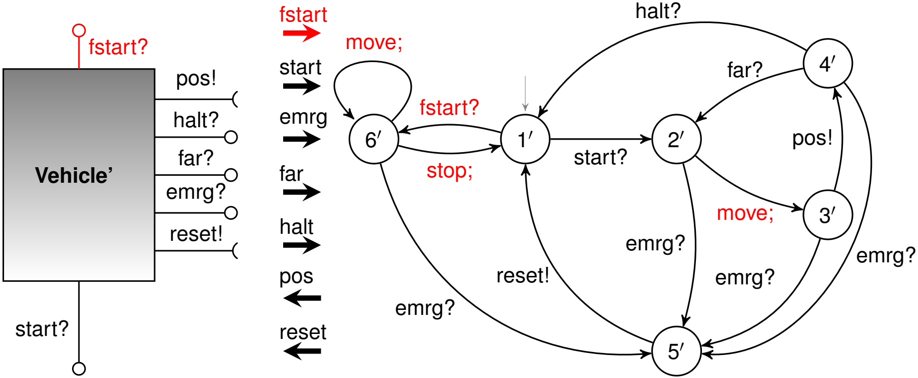

fstart?
position!
position?
halt?
halt!
far?
far!
emergency?
emergency!
reset!
reset?
Vehicle
Station
Emergency Halt
start?
start!
Starter

The Vehicle and Station interfaces





Refinement of the Vehicle interface



That’s all, thanks for your attention

References I

L. Alfaro and T. Henzinger.
Interface automata.
ACM Press, 2001.

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.
INRIA technical report, Avril 1999.