Relating Alternating Relations for Conformance and Refinement

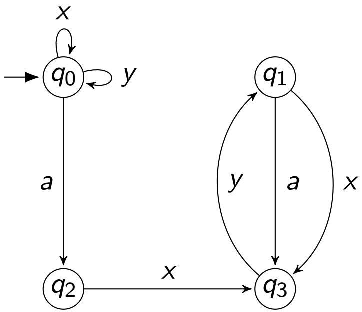
Ramon Janssen Frits Vaandrager Jan Tretmans

Radboud University, Nijmegen, the Netherlands

iFM, Bergen, Norway, December 5, 2019

Interface automata

Labeled transition systems with inputs $a, b, ..$ and outputs $x, y, ..$ (Tretmans '96):



A.k.a. *interface automata* (De Alfaro & Henzinger '01) or *I/O automata* (when input enabled) (Lynch & Tuttle '87).

History

Interface Automata

ioco

Tretmans '96

History

Interface Automata

ioco

Tretmans '96

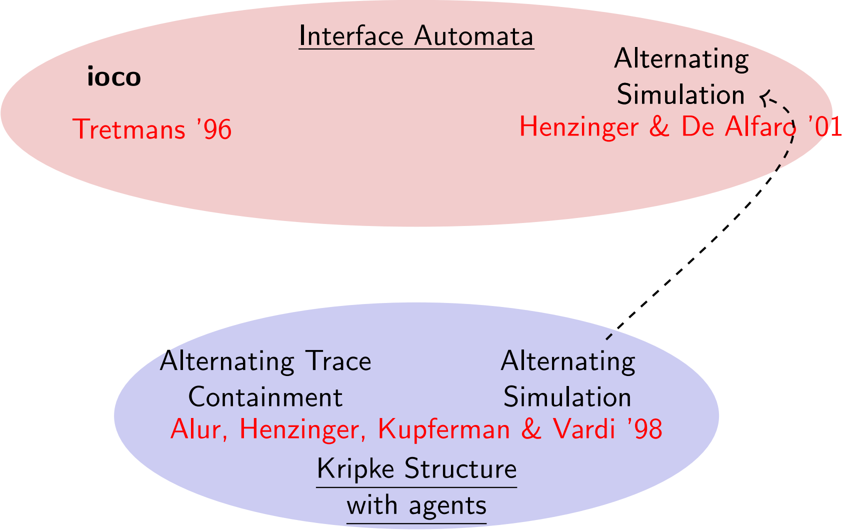
Alternating Trace Containment

Alternating Simulation

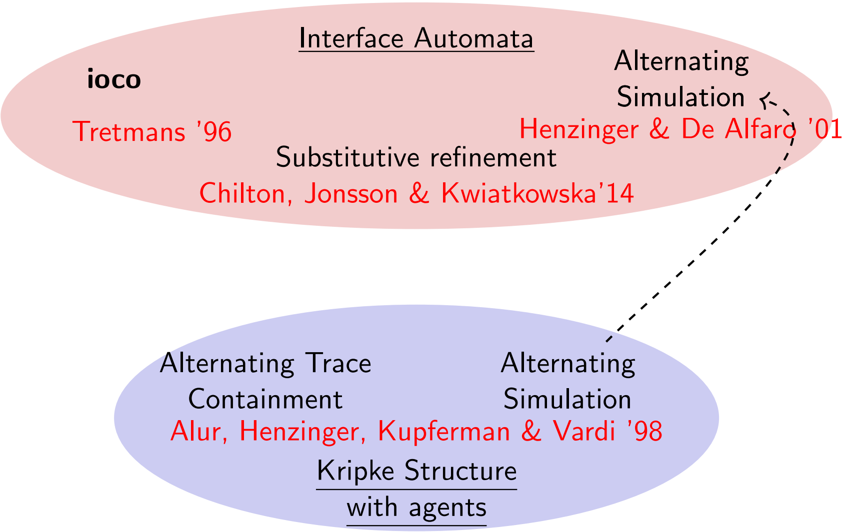
Alur, Henzinger, Kupferman & Vardi '98

Kripke Structure with agents

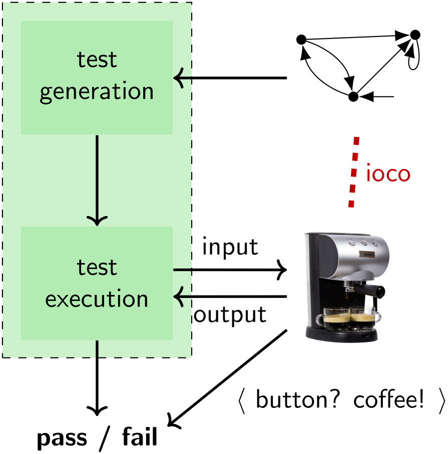
History


Interface Automata
ioco
Tretmans '96
Alternating Simulation
Henzinger & De Alfaro '01
Alternating Trace Containment
Alternating Simulation
Alur, Henzinger, Kupferman & Vardi '98
Kripke Structure with agents

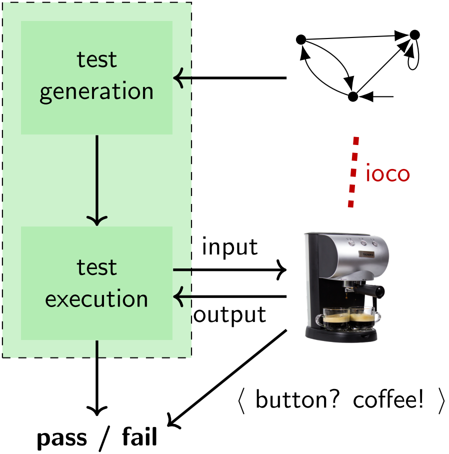
History


Interface Automata
ioco
Tretmans '96
Alternating
Simulation
Henzinger & De Alfaro '01
Substitutive refinement
Chilton, Jonsson & Kwiatkowska'14
Alternating Trace
Containment
Alternating
Simulation
Alur, Henzinger, Kupferman & Vardi '98
Kripke Structure
with agents

Model-Based Testing

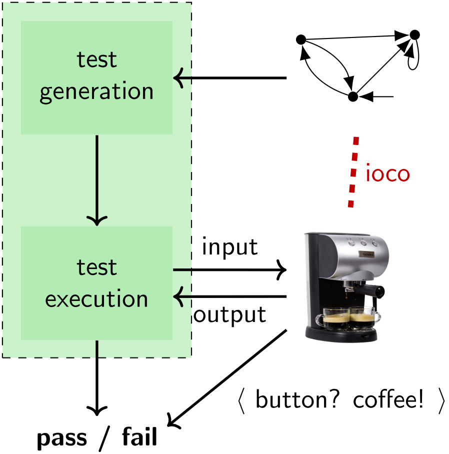


Model-Based Testing



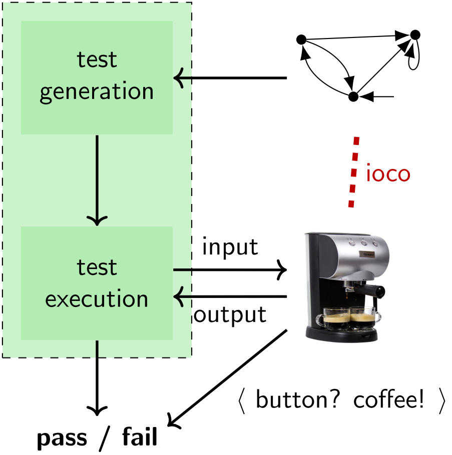
- ioco $\subseteq IA_{i.e.} \times IA$
- Assume SUT can be modeled as *input enabled* IA

Model-Based Testing



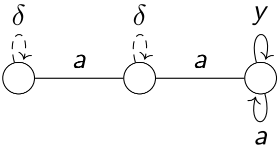
- ioco $\subseteq IA_{i.e.} \times IA$
- Assume SUT can be modeled as *input enabled* IA
- **uioco** fixes problem with compositionality (vd Bijl, Rensink & Tretmans '03)

Model-Based Testing



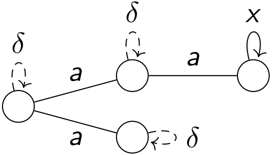
- ioco $\subseteq IA_{i.e.} \times IA$
- Assume SUT can be modeled as *input enabled* IA
- **uioco** fixes problem with compositionality (vd Bijl, Rensink & Tretmans '03)
- **uioco** generalized to preorder on IA's (Volpato & Tretmans '13)
- **ioco** used as conformance relation in many tools

Difference between ioco and uioco



uioco

~~ioco~~



Tester versus bug

There are fundamental connections between model-based testing and 2-player concurrent games.

Tester versus bug

There are fundamental connections between model-based testing and 2-player concurrent games.

Van den Bos & Stoelinga '18 show that in a deterministic setting:

- specifications are game arenas

Tester versus bug

There are fundamental connections between model-based testing and 2-player concurrent games.

Van den Bos & Stoelinga '18 show that in a deterministic setting:

- specifications are game arenas
- test cases are game strategies

Tester versus bug

There are fundamental connections between model-based testing and 2-player concurrent games.

Van den Bos & Stoelinga '18 show that in a deterministic setting:

- specifications are game arenas
- test cases are game strategies
- test case derivation is strategy synthesis

Tester versus bug

There are fundamental connections between model-based testing and 2-player concurrent games.

Van den Bos & Stoelinga '18 show that in a deterministic setting:

- specifications are game arenas
- test cases are game strategies
- test case derivation is strategy synthesis
- conformance is alternating-trace containment

Tester versus bug

There are fundamental connections between model-based testing and 2-player concurrent games.

Van den Bos & Stoelinga '18 show that in a deterministic setting:

- specifications are game arenas
- test cases are game strategies
- test case derivation is strategy synthesis
- conformance is alternating-trace containment

Can we lift these connections to general, nondeterministic settting?

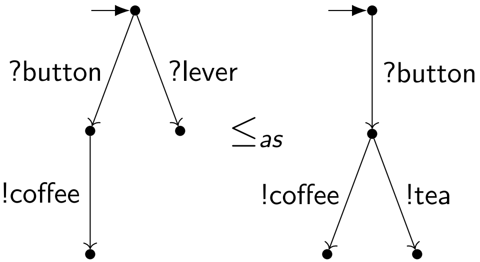
Alternating Refinements

- Kripke structures with *collaborative* and *adversarial* agents
- Refinement restrictse behaviour of collaborative agents, without restricting the adversarial agents
- Two refinements
  - alternating *simulation*
  - alternating *trace containment*

Alternating Simulation

Adaptation to interface automata:

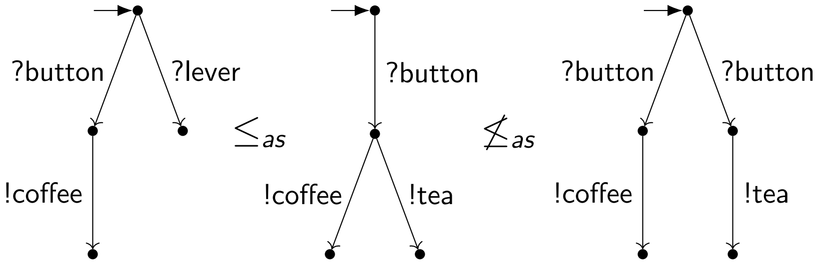
- Input-choices are adversarial
- Output-choices are collaborative



Alternating Simulation

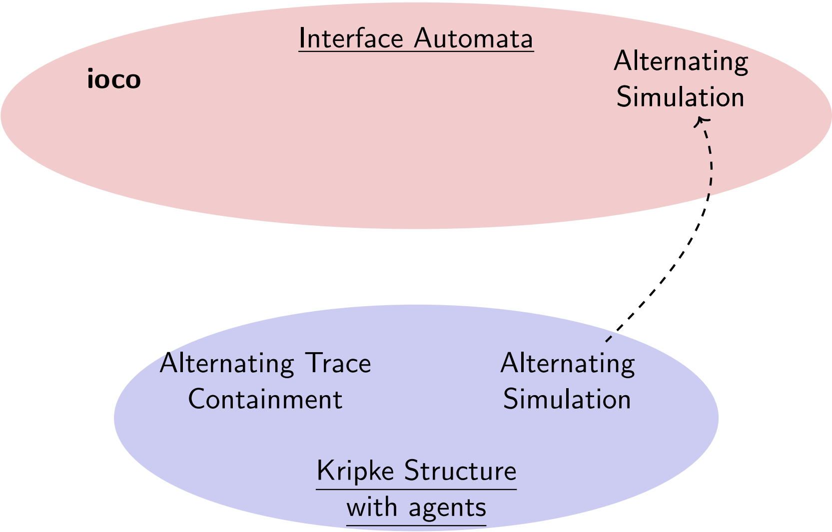
Adaptation to interface automata:

- Input-choices are adversarial
- Output-choices are collaborative

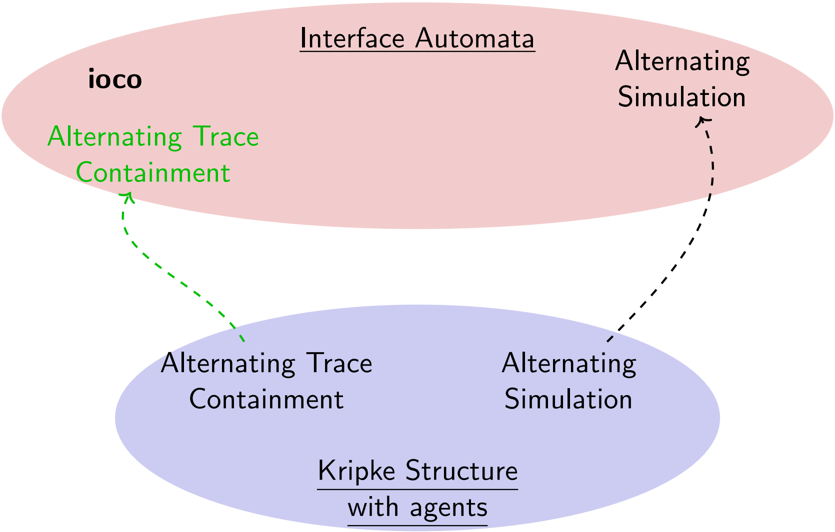


There is no (reasonable) testing scenario for alternating simulation!

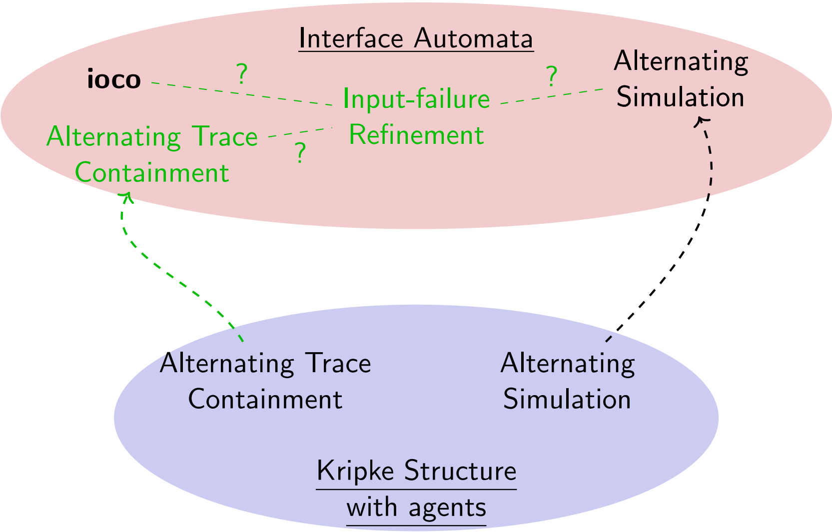
Contributions



Contributions

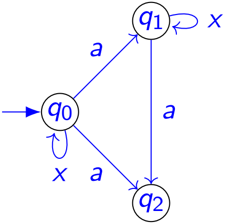


Contributions



Input-universal and output-existential

Let s be an IA with states Q inputs I and output O. Then



$$\mathrm{out}(Q) = \{x \in O \mid \exists q \in Q : q \xrightarrow{x}\}$$

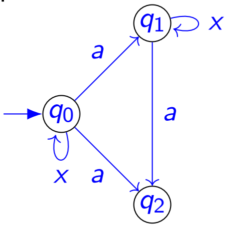
$$\mathrm{in}(Q) = \{a \in I \mid \forall q \in Q : q \xrightarrow{a}\}$$

$$\mathrm{out}(\{q_1, q_2\}) = \{x\}$$

$$\mathrm{in}(\{q_1, q_2\}) = \emptyset$$

Input-universal and output-existential

Let s be an IA with states Q inputs I and output O. Then



$$\mathrm{out}(Q) = \{x \in O \mid \exists q \in Q : q \xrightarrow{x}\}$$

$$\mathrm{in}(Q) = \{a \in I \mid \forall q \in Q : q \xrightarrow{a}\}$$

$$\mathrm{out}(\{q_1, q_2\}) = \{x\}$$

$$\mathrm{in}(\{q_1, q_2\}) = \emptyset$$

Let $L = I \cup O$ and $\sigma \in L^*$. Then

σ is *s-output-existential* if $\forall j \in 1 \ldots n : \ell^j \in \mathrm{out}(s \text{ after } \ell^1 \ldots \ell^{j-1}) \cup I$

σ is *s-input-universal* if $\forall j \in 1 \ldots n : \ell^j \in \mathrm{in}(s \text{ after } \ell^1 \ldots \ell^{j-1}) \cup O$

$a\ a$ is output-existential but not input-universal

$a\ x\ a$ is output-existential and input-universal

Input-Universal and Output-Existential refinement

Definition

Let $\mathrm{OE}(s)$ denote the set of s-output-existential words, and $\mathrm{IU}(s)$ the set of s-input-universal words. Then

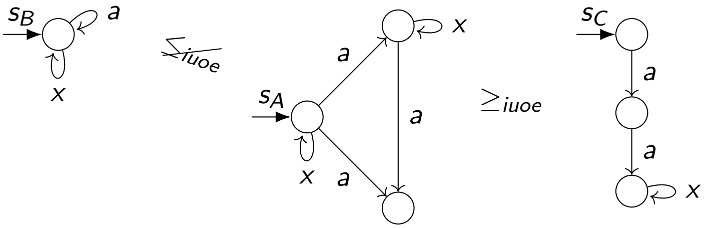
$$s_1 \leq_{iuoe} s_2 \iff \mathrm{OE}(s_1) \cap \mathrm{IU}(s_2) \subseteq \mathrm{IU}(s_1) \cap \mathrm{OE}(s_2)$$

Input-Universal and Output-Existential refinement

Definition

Let OE(s) denote the set of s-output-existential words, and IU(s) the set of s-input-universal words. Then

$$s_1 \leq_{iuoe} s_2 \iff \mathrm{OE}(s_1) \cap \mathrm{IU}(s_2) \subseteq \mathrm{IU}(s_1) \cap \mathrm{OE}(s_2)$$

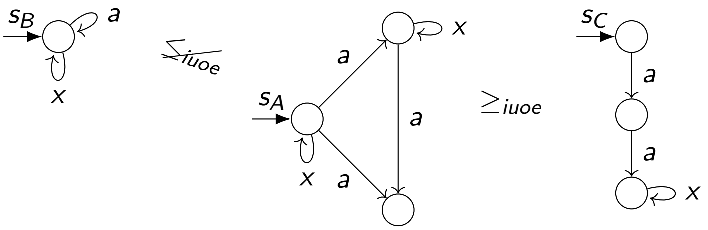


Input-Universal and Output-Existential refinement

Definition

Let OE(s) denote the set of s-output-existential words, and IU(s) the set of s-input-universal words. Then

$$s_1 \leq_{iuoe} s_2 \iff \mathrm{OE}(s_1) \cap \mathrm{IU}(s_2) \subseteq \mathrm{IU}(s_1) \cap \mathrm{OE}(s_2)$$



$axax \in \mathrm{OE}(s_B) \cap \mathrm{IU}(s_A)$ and $axax \notin \mathrm{OE}(s_A)$

Relating uioco, $\leq_{if}$ and $\leq_{iuoe}$

If s is an interface automaton then $\Delta(s)$ is the interface automaton obtained by adding a self-loop with output label δ to every state of s that does not enable an output.

Theorem

s_1 **uioco** $s_2 \quad \Leftrightarrow \quad \Delta(s_1) \leq_{if} \Delta(s_2)$

Theorem

$s_1 \leq_{if} s_2 \quad \Leftrightarrow \quad s_1 \leq_{iuoe} s_2$

Here $\leq_{if}$ is the substitutive refinement of Chilton, Jonsson & Kwiatkowska '14.

Agents and Strategies

- Alternating trace containment presupposes a set of agents, which are either collaborative or adversarial.
- Agents may restrict possible transitions following an initial path by choosing a strategy.
- Once every agent has chosen a strategy, we obtain a unique path in the interface automaton.
- Collaborative agent choose at most one input action, and adversarial agents choose at most one output action, a determinization strategy, and a race condition strategy.
- Domains of strategies denoted $\Sigma_i(s)$, $\Sigma_o(s)$, $\Sigma_d(s)$ and $\Sigma_r(s)$, respectively.

Two player game

Game of alternating trace containment played by two players, the protagonist and the antagonist, on IAs s_1 and s_2:

1. Antagonist chooses strategy for collaborative agents s_1
2. Protagonist chooses strategy for collaborative agents s_2
3. Antagonist chooses strategy for adversarial agents s_2
4. Protogonist chooses strategy for adversarial agents s_1

Protogonist wins if traces of runs in s_1 and s_2 are the same.

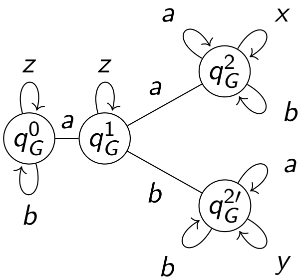
Alternating trace containment

Definition

Let s_1, s_2 be interface automata.
Then s_1 is alternating-trace contained in s_2, denoted $s_1 \leq_{atc} s_2$, if

$$\forall f_o^1 \in \Sigma_o(s_1), \forall f_d^1 \in \Sigma_d(s_1), \forall_r^1 \in \Sigma_d(s_1),$$
$$\exists_o^2 \in \Sigma_o(s_2), \exists f_d^2 \in \Sigma_d(s_2), \exists f_r^2 \in \Sigma_r(s_2),$$
$$\forall f_i^2 \in \Sigma_i(s_2), \exists f_i^1 \in \Sigma_i(s_1) :$$
$$\mathrm{trace}(\mathrm{outcome}(f_i^1, f_o^1, f_d^1, f_r^1)) = \mathrm{trace}(\mathrm{outcome}(f_i^2, f_o^2, f_d^2, f_r^2))$$

Counterexample



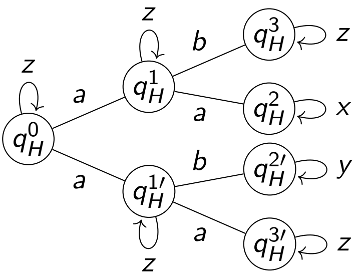
ioco

uioco

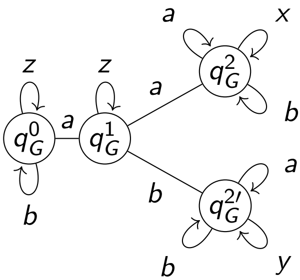
$\leq_{if}$

$\leq_{iuoe}$

$\not\leq_{atc}$



Counterexample



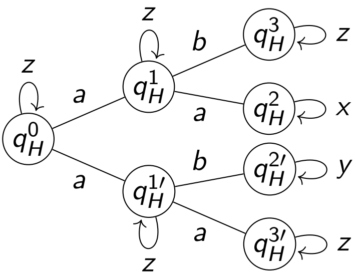
ioco

uioco

$\leq_{if}$

$\leq_{iuoe}$

$\not\leq_{atc}$



Problems in definition $\leq_{atc}$:

(1) order in which strategies are selected,

(2) input strategies not trace based.

Changing the rules of the game

Definition

Let s_1, s_2 be interface automata.
Then $s_1 \leq^{tb}_{\forall\forall\exists\exists} s_2$, if

$$\forall f_i^2 \in \Sigma_{i,tb}(s_2), \forall f_o^1 \in \Sigma_o(s_1), \forall f_d^1 \in \Sigma_d(s_1), \forall_r^1 \in \Sigma_d(s_1),$$
$$\exists f_i^1 \in \Sigma_{i,tb}(s_1), \exists_o^2 \in \Sigma_o(s_2), \exists f_d^2 \in \Sigma_d(s_2), \exists f_r^2 \in \Sigma_r(s_2) :$$
$$\mathrm{trace}(\mathrm{outcome}(f_i^1, f_o^1, f_d^1, f_r^1)) = \mathrm{trace}(\mathrm{outcome}(f_i^2, f_o^2, f_d^2, f_r^2))$$

Theorem

$s_1 \leq_{as} s_2 \implies s_1 \leq_{atc} s_2.$

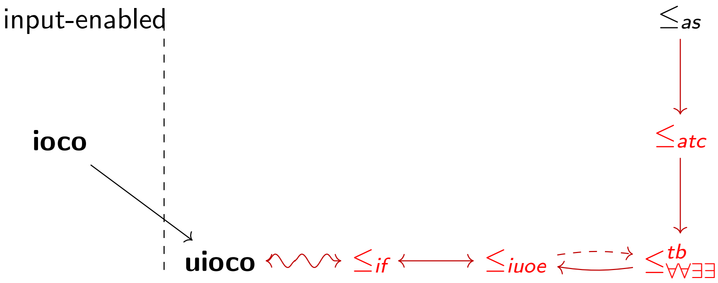
Theorem

$s_1 \leq_{atc} s_2 \implies s_1 \leq^{tb}_{\forall\forall\exists\exists} s_2.$

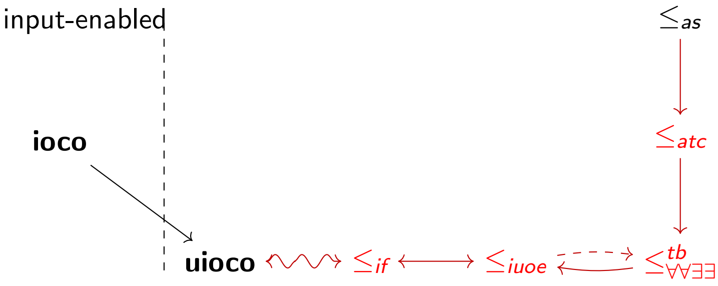
Theorem

$s_1 \leq^{tb}_{\forall\forall\exists\exists} s_2 \implies s_1 \leq_{iuoe} s_2.$
Furthermore, if s_2 is image-finite, then
$s_1 \leq^{tb}_{\forall\forall\exists\exists} s_2 \impliedby s_1 \leq_{iuoe} s_2.$

Conclusions

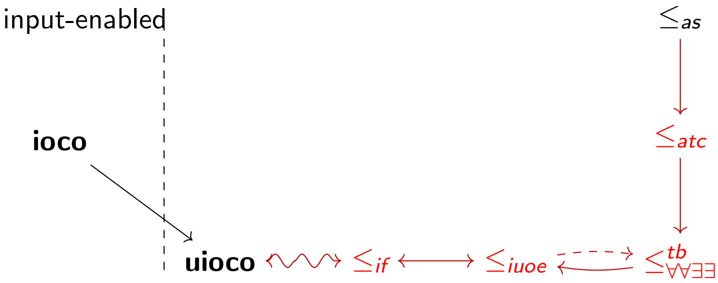


Conclusions



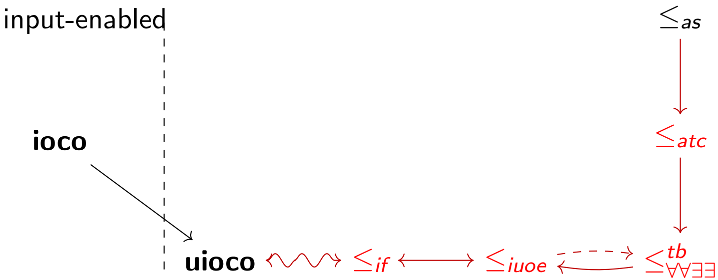
1. use $\leq_{if}$ rather than $\leq_{as}$ as refinement relation

Conclusions



1. use $\leq_{if}$ rather than $\leq_{as}$ as refinement relation
2. use **uioco** rather than **ioco** as conformance relation

Conclusions



1. use $\leq_{if}$ rather than $\leq_{as}$ as refinement relation
2. use **uioco** rather than **ioco** as conformance relation
3. $\leq_{atc}$ not a sensible notion of trace containment