A logical characterisation for input output conformance simulation iocos (Work in Progress)

Luca Aceto **Ignacio Fábregas** Carlos Gregorio-Rodríguez
Anna Ingólsfdóttir

ICE-TCS, School of Computer Science
Reykjavik University, Iceland.

Departamento Sistemas Informáticos y Computación
Universidad Complutense de Madrid, Spain

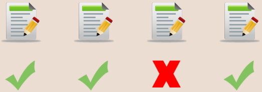
NWPT 2015
Friday, 23^{rd} October

Formal Methods

Formal Methods

Model Based Testing

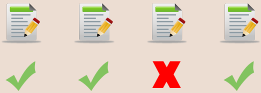
Model



Formal Methods

Model Based Testing

Model

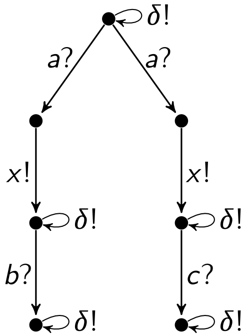


Model Checking

Model		Properties
$\downarrow$		$\downarrow$
Operational description	$\models$	Logic formula
$\downarrow$		
'Minimal' and 'equivalent'	$\models$	Logic formula

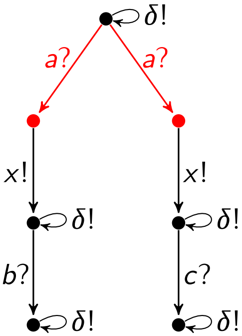
Our Model

- LTS
- Non Determinism
- Inputs
- Outputs
- Explicit quiescence
- Input enabled (not requiered)



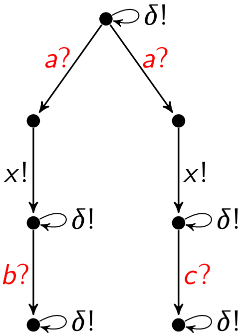
Our Model

- LTS
- Non Determinism
- Inputs
- Outputs
- Explicit quiescence
- Input enabled (not requiered)



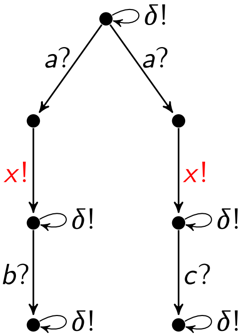
Our Model

- LTS
- Non Determinism
- Inputs
- Outputs
- Explicit quiescence
- Input enabled (not requiered)



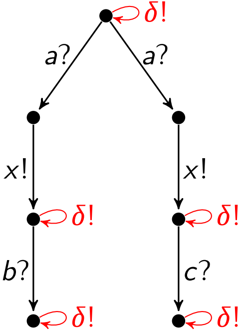
Our Model

- LTS
- Non Determinism
- Inputs
- Outputs
- Explicit quiescence
- Input enabled (not requiered)



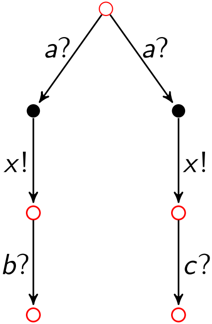
Our Model

- LTS
- Non Determinism
- Inputs
- Outputs
- Explicit quiescence
- Input enabled (not requiered)



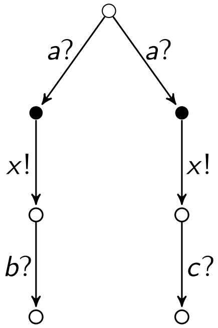
Our Model

- LTS
- Non Determinism
- Inputs
- Outputs
- Explicit quiescence
- Input enabled (not requiered)

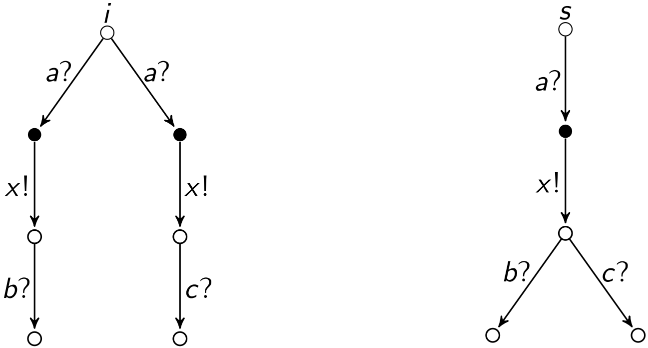


Our Model

- LTS
- Non Determinism
- Inputs
- Outputs
- Explicit quiescence
- Input enabled (not requiered)

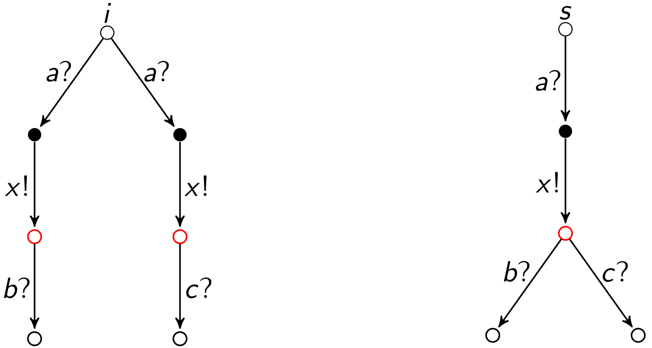


Example 1



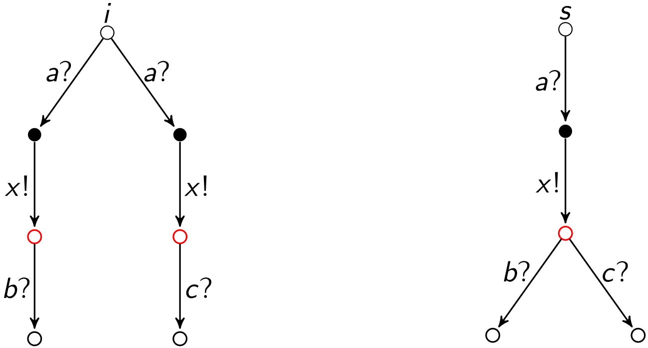
- iocos is a branching semantic.
- *i* io~~c~~os *s*.

Example 1



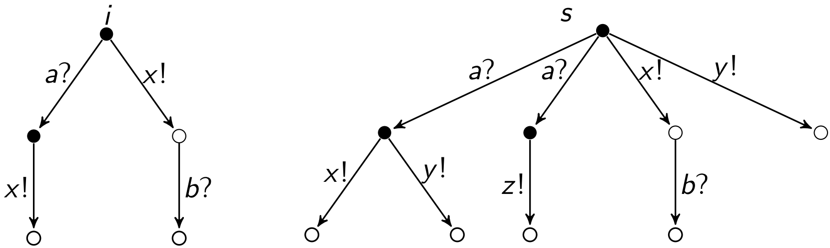
- iocos is a branching semantic.
- i io~~c~~os s.

Example 1



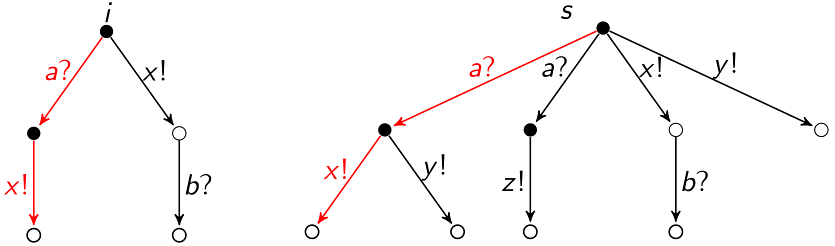
- ioco$\underline{s}$ is a branching semantic.
- i io$\not{c}$o$\underline{s}$ s.

Example 2



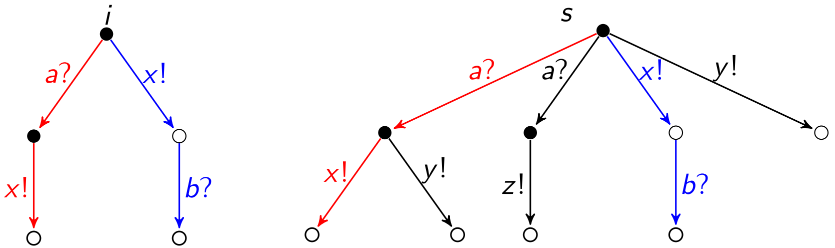
- iocos is a conformance semantic.
- input actions in the specification should be implemented.
- All outputs in the implementation must be allowed by the specification.
- i iocos s.

Example 2



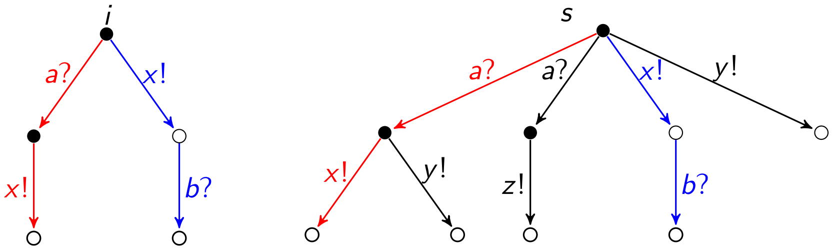
- ioco$\underline{s}$ is a conformance semantic.
- input actions in the specification should be implemented.
- All outputs in the implementation must be allowed by the specification.
- i ioco$\underline{s}$ s.

Example 2



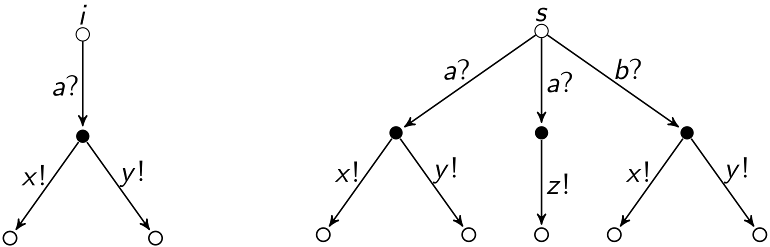
- iocos is a conformance semantic.
- input actions in the specification should be implemented.
- All outputs in the implementation must be allowed by the specification.
- i iocos s.

Example 2



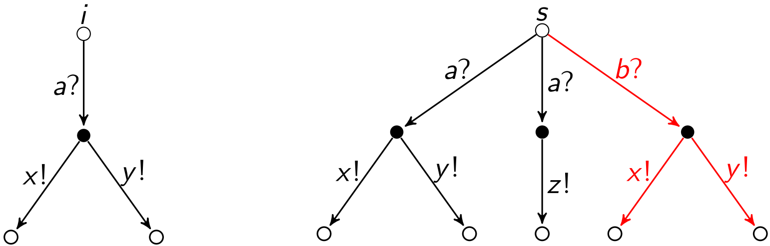
- iocos̲ is a conformance semantic.
- input actions in the specification should be implemented.
- All outputs in the implementation must be allowed by the specification.
- *i* iocos̲ *s*.

Example 3



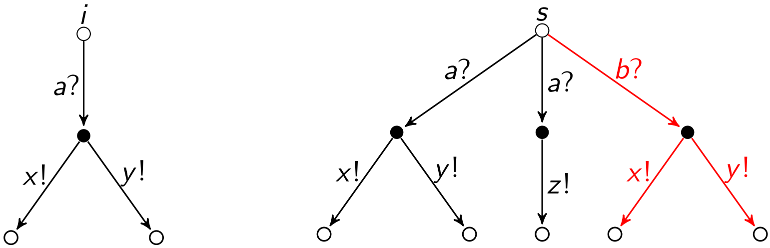
- i must be able to do all the inputs specify by s.
- $i \ \mathrm{io}\cancel{\mathrm{c}}\mathrm{o}\underline{\mathrm{s}} \ s$.

Example 3



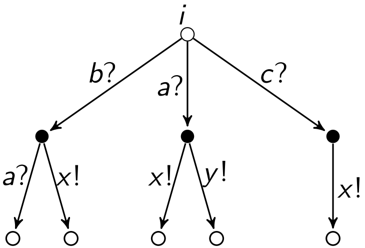
- i must be able to do all the inputs specify by s.
- i io~~c~~o$\underline{s}$ s.

Example 3



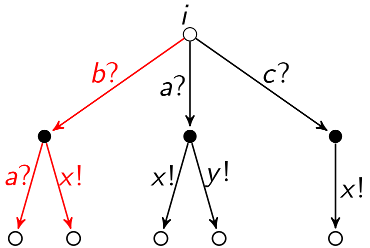
- i must be able to do all the inputs specify by s.
- i io~~c~~o$\underline{s}$ s.

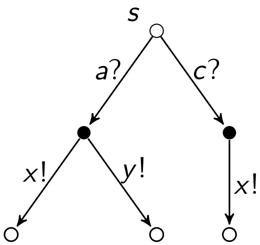
Example 4



- The implementation can add new behaviours for the inputs.
- *i* iocos *s*.

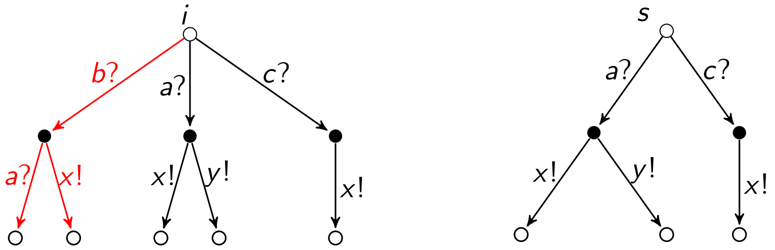
Example 4





- The implementation can add new behaviours for the inputs.
- i iocos s.

Example 4



- The implementation can add new behaviours for the inputs.
- i ioco$\underline{s}$ s.

Input-Output Conformance relation

An ioco$\underline{s}$ relation

R is *an* ioco$\underline{s}$*-relation* iff for any $(i, s) \in R$:

1. $\text{ins}(s) \subseteq \text{ins}(i)$
2. $a? \in \text{ins}(s) \quad i \xrightarrow{a?} i' \quad \text{then} \quad s \xrightarrow{a?} s', \ (i', s') \in R$
3. $x! \in \text{outs}(i) \quad i \xrightarrow{x!} i' \quad \text{then} \quad s \xrightarrow{x!} s', \ (i', s') \in R$

ioco$\underline{s}$

$\text{ioco}\underline{s} = \bigcup\{R \mid R \text{ is a ioco}\underline{s}\text{-relation}\}$ $\qquad (i, s) \in \text{ioco}\underline{s} \leftrightarrow i \ \text{ioco}\underline{s} \ s$

Offline testing

Soundness

p pass T for any $T \in \mathcal{T}(p)$

Completeness

$\forall T \in \mathcal{T}(s)$ i pass T iff i iocos̲ s

"C. Gregorio-Rodríguez, L. Llana, R. Martínez-Torres: Input Output Conformance Simulation (iocos̲) for Model Based Testing. FORTE 2013"

Online testing

Algorithm 1 Online Testing Algorithm for ioco$\underline{s}$

```
function TE(s, iut, maxIter)
    continue ← ✓
    numIter ← maxIter
    while numIter > 0 ∧ continue == ✓ do
        continue, numIter ← TE_REC(s, iut, numIter)
        if continue == ✓ then
            reset iut
    return continue
function TE_REC(s, iut, numIter)
    if numIter = 0 then
        return ✓, numIter
    else
        choice
            case action do                        ▷ Offers an input to the implementation
                choice a ∈ ins(s)
                if a? is not enabled in iut then
                    return ✗, numIter
                send a? to iut
                iut_0 ← copy(iut)
                for s' ∈ s after a? do
                    iut ← copy(iut_0)
                    continue, numIter ← TE_REC(s', iut, numIter − 1)
                    if continue == ✓ then
                        return ✓, numIter
                return ✗, numIter
            case wait do                          ▷ Waits for an output from the implementation
                wait o! from iut
                if s after o! = ∅ then
                    return ✗, T
                iut_0 ← copy(iut)
                for s' ∈ s after o! do
                    iut ← copy(iut_0)
                    continue, numIter ← TE_REC(s', iut, numIter − 1)
                    if continue == ✓ then
                        return ✓, numIter
                return ✗, numIter
            case reset do                         ▷ Resets implementation and restart
                return ✓, maxIter
```

“C. Gregorio-Rodríguez, L. Llana, R. Martínez-Torres: Effectiveness for Input Output Conformance Simulation ioco$\underline{s}$. FORTE 2014”

Implementations

General Coarser Partition Problem (GCPP)

- Can be effectively computed using the GCPP algorithm.
- This allows to perform iocos-minimisation.
  - Given process p, compute q s.t. q $\text{iocos}_{\equiv}$ p and q has a minimal LTS.

mCRL2 tool (Jan Friso Groote, TU Eindhoven (CWI, Twente...))

- Implementation of iocos in **mCRL2**.

"C. Gregorio-Rodríguez, L. Llana, R. Martínez-Torres: Extending mCRL2 with ready simulation and iocos input-output conformance simulation. SAC 2015"

Logic for ioco$\underline{s}$

Model Checking

$$\begin{array}{ccc} \text{Model} & & \text{Properties} \\ \downarrow & & \downarrow \\ \text{Operational description} & \models & \text{Logic formula} \\ \downarrow & & \\ \text{'Minimal' and 'equivalent'} & \models & \text{Logic formula} \end{array}$$

- We present a logic that characterizes ioco$\underline{s}$.
  - Both, preorder and equivalence.
- Is a subset of the Hennessy-Milner Logic.

Definitions

Syntax of $\mathcal{L}_{\text{ioco}\underline{\text{s}}}$

$$\phi ::= \mathrm{tt} \mid \mathrm{ff} \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle\!| a? |\!\rangle \phi \mid \langle x! \rangle \phi.$$

Semantics of $\mathcal{L}_{\text{ioco}\underline{\text{s}}}$

- Standard interpretations for tt, ff, $\wedge$ and $\vee$.
- $p \models \langle x! \rangle \phi$ iff $p' \models \phi$ for some $p \xrightarrow{x!} p'$.
- $p \models \langle\!| a? |\!\rangle \phi$ iff $p \stackrel{a?}{\not\longrightarrow}$ or $p' \models \phi$ for some $p \xrightarrow{a?} p'$.
- $\langle\!| a? |\!\rangle \phi$ is logically equivalent to $[\,a?\,]\mathrm{ff} \vee \langle a? \rangle \phi$.

Definitions

Syntax of $\mathcal{L}_{\mathrm{ioco\underline{s}}}$

$$\phi ::= \mathrm{tt} \mid \mathrm{ff} \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle\!| a? |\!\rangle \phi \mid \langle x! \rangle \phi.$$

Semantics of $\mathcal{L}_{\mathrm{ioco\underline{s}}}$

- Standard interpretations for tt, ff, $\wedge$ and $\vee$.
- $p \models \langle x! \rangle \phi$ iff $p' \models \phi$ for some $p \xrightarrow{x!} p'$.
- $p \models \langle\!| a? |\!\rangle \phi$ iff $p \overset{a?}{\not\longrightarrow}$ or $p' \models \phi$ for some $p \xrightarrow{a?} p'$.
- $\langle\!| a? |\!\rangle \phi$ is logically equivalent to $[\, a? \,]\mathrm{ff} \vee \langle a? \rangle \phi$.

Some results

$\mathcal{L}_{\text{ioco}\underline{s}}$ characterizes the preorder

i ioco$\underline{s}$ s iff $(\forall \phi \in \mathcal{L}_{\text{ioco}\underline{s}} \quad i \models \phi$ then $s \models \phi)$.

$\mathcal{L}_{\text{ioco}\underline{s}}$ characterizes the induced equivalence

i ioco$\underline{s}_{\equiv}$ s iff $(\forall \phi \in \mathcal{L}_{\text{ioco}\underline{s}} \quad i \models \phi$ iff $s \models \phi)$.

Corollary

For all ϕ in $\mathcal{L}_{\text{ioco}\underline{s}}$ if we want to check $p \models \phi$, it is equivalent to minimise p to q (using GCPP) and solve $q \models \phi$

An Alternative logic

- $\mathcal{L}_{\text{ioco}\underline{\text{s}}}$ follows a standard approach to the characterisation of simulation semantics.
- However, ioco$\underline{\text{s}}$ was originated in the model based testing environment.
  - The natural reading for a logical characterisation would be *"every formula produced by the specification should be also proved correct in the implementation"*.

$\widetilde{\mathcal{L}}_{\text{ioco}\underline{\text{s}}}$

Syntax

$$\phi ::= \mathsf{tt} \mid \mathsf{ff} \mid \phi \wedge \phi \mid \phi \vee \phi \mid [\![a?]\!]\phi \mid [\,x!\,]\phi.$$

Semantics

- Standard interpretations for tt, ff, $\wedge$ and $\vee$.
- $p \models [\,x!\,]\phi$ iff $p' \models \phi$ for each $p \xrightarrow{x!} p'$.
- $p \models [\![a?]\!]\phi$ iff $p \xrightarrow{a?}$ and $p' \models \phi$ for each $p \xrightarrow{a?} p'$.
- $[\![a?]\!]\phi$ is logically equivalent to $\langle a? \rangle \mathsf{tt} \wedge [\,a?\,]\phi$.

$\widetilde{\mathcal{L}}_{\text{ioco}\underline{s}}$

Syntax

$$\phi ::= \mathrm{tt} \mid \mathrm{ff} \mid \phi \wedge \phi \mid \phi \vee \phi \mid [\![a?]\!]\phi \mid [\,x!\,]\phi.$$

Semantics

- Standard interpretations for tt, ff, $\wedge$ and $\vee$.
- $p \models [\,x!\,]\phi$ iff $p' \models \phi$ for each $p \xrightarrow{x!} p'$.
- $p \models [\![a?]\!]\phi$ iff $p \xrightarrow{a?}$ and $p' \models \phi$ for each $p \xrightarrow{a?} p'$.
- $[\![a?]\!]\phi$ is logically equivalent to $\langle a?\rangle \mathrm{tt} \wedge [\,a?\,]\phi$.

Some results

$\widetilde{\mathcal{L}}_{\text{ioco}\underline{s}}$ characterizes the preorder

i ioco$\underline{s}$ s iff ($\forall \phi \in \widetilde{\mathcal{L}}_{\text{ioco}\underline{s}} \quad s \models \phi$ then $i \models \phi$).

$\widetilde{\mathcal{L}}_{\text{ioco}\underline{s}}$ characterizes the induced equivalence

i ioco$\underline{s}_{\equiv}$ s iff ($\forall \phi \in \mathcal{L}_{\text{ioco}\underline{s}} \quad s \models \phi$ iff $i \models \phi$).

Corollary

For all ϕ in $\widetilde{\mathcal{L}}_{\text{ioco}\underline{s}}$ if we want to check $p \models \phi$, it is equivalent to minimise p to q (using GCPP) and solve $q \models \phi$

Relation between $\mathcal{L}_{\text{ioco}\underline{\text{s}}}$ & $\widetilde{\mathcal{L}}_{\text{ioco}\underline{\text{s}}}$

Bijection $\mathcal{T} : \mathcal{L}_{\text{ioco}\underline{\text{s}}} \rightarrow \widetilde{\mathcal{L}}_{\text{ioco}\underline{\text{s}}}$:

- $\mathcal{T}(\text{tt}) = \text{ff}$.
- $\mathcal{T}(\text{ff}) = \text{tt}$.
- $\mathcal{T}(\phi_1 \wedge \phi_2) = \mathcal{T}(\phi_1) \vee \mathcal{T}(\phi_2)$.
- $\mathcal{T}(\phi_1 \vee \phi_2) = \mathcal{T}(\phi_1) \wedge \mathcal{T}(\phi_2)$.
- $\mathcal{T}(\langle\!|a?|\!\rangle\phi) = [\![a?]\!]\mathcal{T}(\phi)$.
- $\mathcal{T}(\langle x!\rangle\phi) = [\,x!\,]\mathcal{T}(\phi)$.

The inverse function $\mathcal{T}^{-1} : \widetilde{\mathcal{L}}_{\text{ioco}\underline{\text{s}}} \rightarrow \mathcal{L}_{\text{ioco}\underline{\text{s}}}$ is defined in the obvious way.

Characteristic formula

Definition

A formula ϕ is *characteristic* for s iff

- $s \models \phi$ and
- for all i it holds that $i \models \phi$ if and only if i ioco$\underline{s}$ s.

Bisimulation

$$\chi(p) = \bigwedge_{a, p \xrightarrow{a} p'} \langle a \rangle \chi(p') \wedge \bigwedge_{a \in A} [\, a\,] \bigvee_{p \xrightarrow{a} p'} \chi(p')$$

Characteristic formula

$$\chi(p) = \bigwedge_{a? \in \mathrm{ins}(p)} [\![a?]\!] \bigvee_{p \xrightarrow{a?} p'} \chi(p') \wedge \bigwedge_{x! \in O} [\,x!\,] \bigvee_{p \xrightarrow{x!} p'} \chi(p')$$

Theorem

- Applying a result in "L. Aceto, A. Ingólfsdóttir, and J. Sack. Characteristic formulae for fixed-point semantics: A general framework. EXPRESS 2009".

Characteristic formula

$$\chi(p) = \bigwedge_{a? \in \mathrm{ins}(p)} [\![a?]\!] \bigvee_{p \xrightarrow{a?} p'} \chi(p') \wedge \bigwedge_{x! \in O} [\,x!\,] \bigvee_{p \xrightarrow{x!} p'} \chi(p')$$

Theorem

- Applying a result in "L. Aceto, A. Ingólfsdóttir, and J. Sack. Characteristic formulae for fixed-point semantics: A general framework. EXPRESS 2009".

Characteristic formula

$$\chi(p) = \bigwedge_{a? \in \mathrm{ins}(p)} [\![a?]\!] \bigvee_{p \xrightarrow{a?} p'} \chi(p') \wedge \bigwedge_{x! \in O} [\, x! \,] \bigvee_{p \xrightarrow{x!} p'} \chi(p')$$

Theorem

- Applying a result in "L. Aceto, A. Ingólfsdóttir, and J. Sack. Characteristic formulae for fixed-point semantics: A general framework. EXPRESS 2009".

Characteristic formula

$$\chi(p) = \bigwedge_{a? \in \mathrm{ins}(p)} [\![a?]\!] \bigvee_{p \xrightarrow{a?} p'} \chi(p') \wedge \bigwedge_{x! \in O} [\, x! \,] \bigvee_{p \xrightarrow{x!} p'} \chi(p')$$

Theorem

- Applying a result in "L. Aceto, A. Ingólfsdóttir, and J. Sack. Characteristic formulae for fixed-point semantics: A general framework. EXPRESS 2009".

Future Work

- Relation of $\mathcal{L}_{\mathrm{ioco}\underline{\mathrm{s}}}$ with other logics in the literature
  - Ready simulation logic, covariant-contravariant simulation logic and μ-calculus.
- Expressive logic for ioco$\underline{\text{s}}$
  - ACTL