Building a Concurrent Operational Semantics: the Example of the Orc Language

Matthieu Perrin Claude Jard Achour Mostefaoui

AeLoS / GDD
LINA, University of Nantes

AeLoS meeting

January, 9th 2014

Introduction

Programming in distributing systems:

- how to specify a distributed language ?
- how to prove properties on these programs ?
- how classical solutions handle disorder ?

Plan

Plan

A language for Web orchestration

Philosophy :

- web sites and services already exist
- orchestration of the Web
- programming language *vs* calculus
- true model of concurrent programming

Particularities :

- use of sites
- multiple publications
- original connectors

Sites

Orc sites

- look like functions
- publish 0, 1 or more values
- external / internal

Some examples

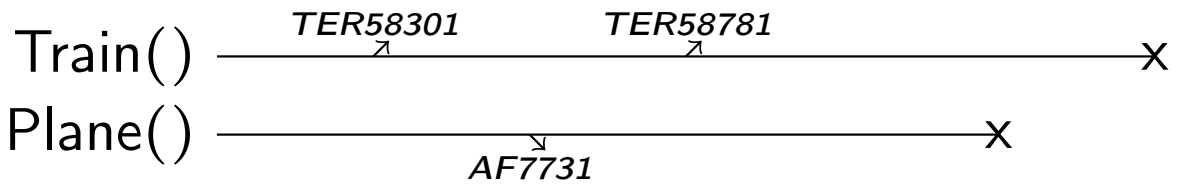
- sncf.com
- **def** Train() = sncf.com("*Nantes*","*Rennes*","09/12/13")
- 1, "hello world", true, signal
- +, max
- Record, r, r.read, r.write,
- ift

Parallel composition

$f|g$

- f and g are started in parallel

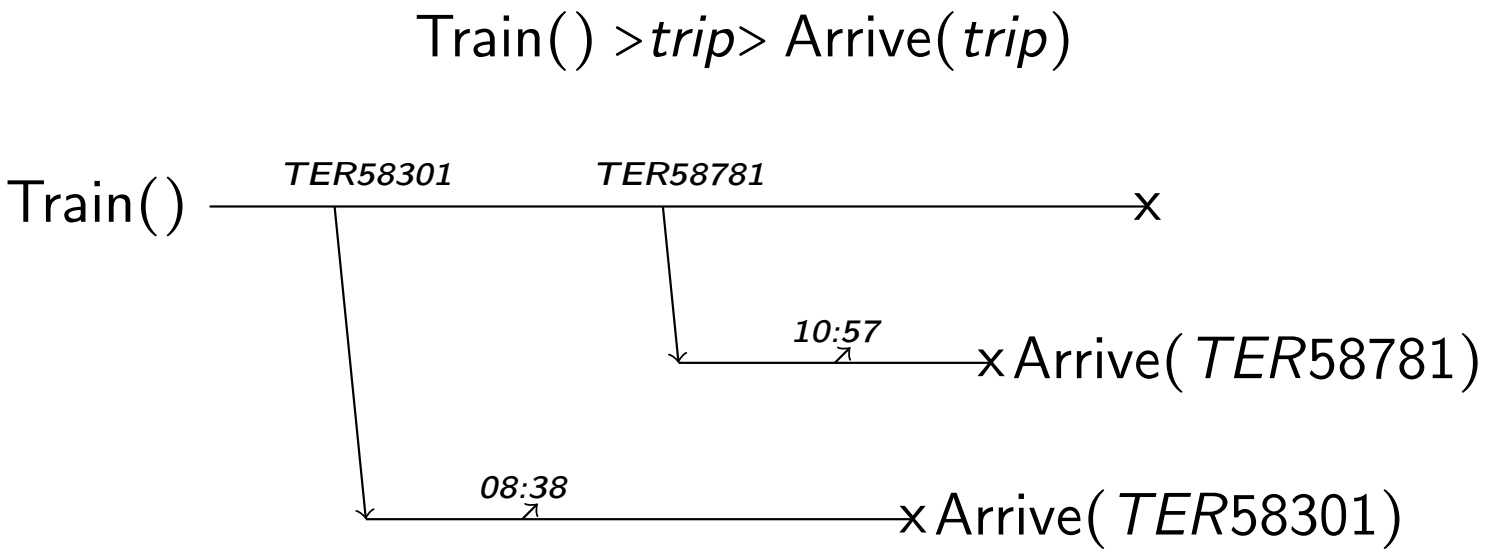
Train()|Plane()



Sequential composition

$f >x> g$

- f is started alone first
- a new instance of g started at each publication by f

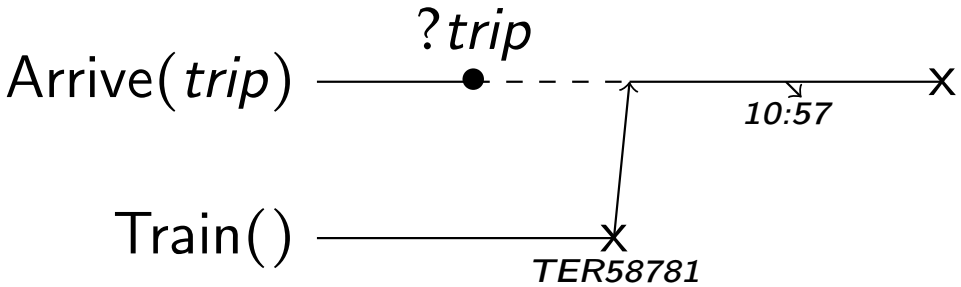


Prunning

$f <x< g$

- f and g are started in parallel
- f is paused when it needs to evaluate x
- g is halted when it publishes a value
- this value is bounded to x in f

$$\text{Arrive}(trip) <trip< \text{Train}()$$



Otherwise operator

$f ; g$

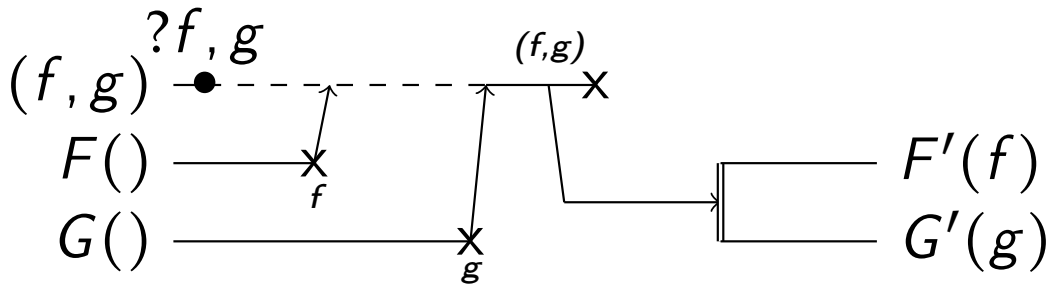
- g is run if and only if f halts without publishing

$$\text{Train}() ; \text{Plane}()$$

Synchronisation point

- run F and G in parallel
- when both have finished, run F' and G' in parallel

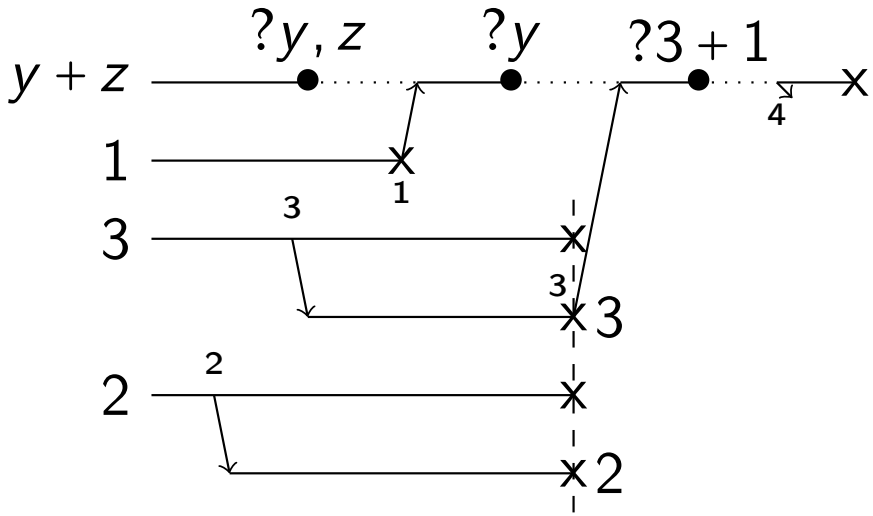
$$((f,g) <f< F() <g< G()) >(f,g)> (F'(f)|G'(g))$$



An example with conflict

- y will be bounded to 2 or 3
- z will be bounded to 1

$$y + z <y< ((2|3) >x> x) <z< 1.$$



How to call Train() or Plane() ?

- draw a random value true or false
- test it
- call the correct site

$$(\text{ift}(x) >\_> \text{Train}())|(\text{iff}(x) >\_> \text{Plane}()) <x< (\textbf{true}|\textbf{false})$$

Plan

Structural operational semantics

Rules of inference

$$\frac{f_1 \xrightarrow{l_1} f_1' \quad \dots \quad f_n \xrightarrow{l_n} f_n'}{f \xrightarrow{l} f'}$$

- if the premises are possible, then the conclusion is possible
- defines a transition system

Semantics of f

- $l_1 \dots l_n \in [\![f]\!]$ if there are $f_1, \dots, f_n$ such that $f \xrightarrow{l_1} f_1 \dots \xrightarrow{l_n} f_n$

The semantics of Orc

Semantics of the prunning operator

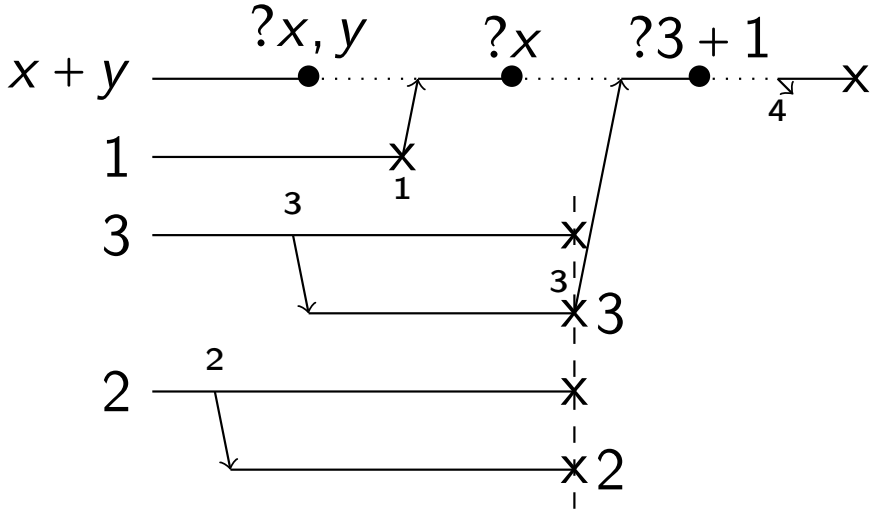
$$(\text{PruneLeft})\ \frac{f \xrightarrow{l} f'}{f \mathbin{<x<} g \xrightarrow{l} f' \mathbin{<x<} g}\ l \neq \omega$$

$$(\text{PruneN})\ \frac{g \xrightarrow{n} g'}{f \mathbin{<x<} g \xrightarrow{n} f \mathbin{<x<} g'}$$

$$(\text{PruneV})\ \frac{g \xrightarrow{!v} g'}{f \mathbin{<x<} g \xrightarrow{h(!v)} [v/x]f}$$

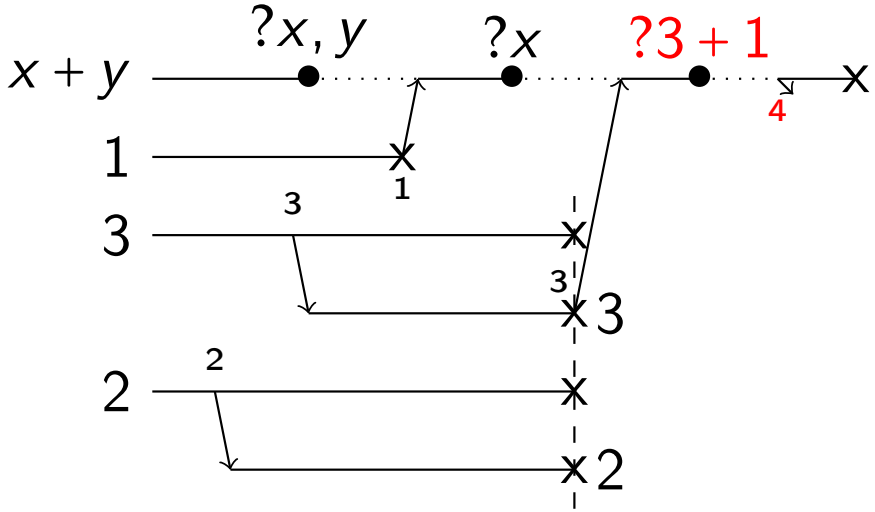
$$(\text{PruneStop})\ \frac{g \xrightarrow{\omega} \bot}{f \mathbin{<x<} g \xrightarrow{h(\omega)} [\mathbf{stop}/x]f}$$

$$y + z \,{<}y{<}\, ((2|3) \,{>}x{>}\, x) \,{<}z{<}\, 1$$



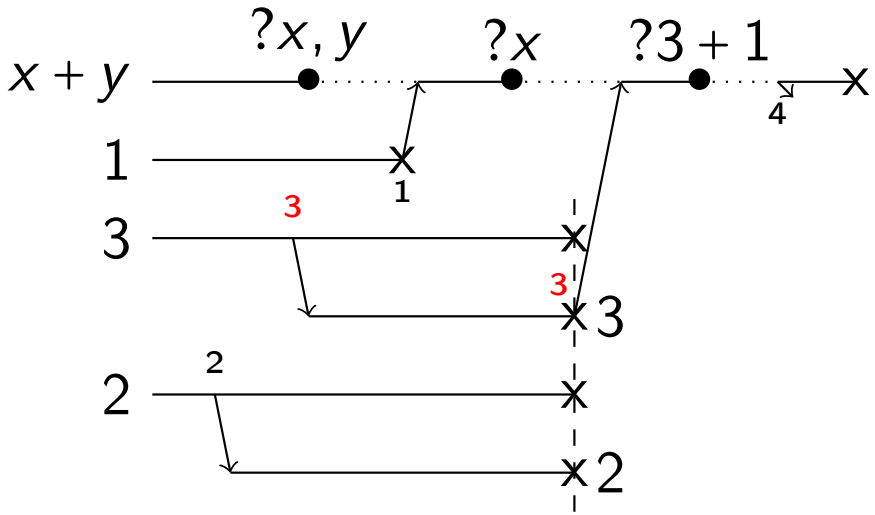
$$\left\{ \begin{array}{c} h(!1).h(!3).h(!3).? + (3,1).!4.\omega, \\ h(!3).h(!1).h(!3).? + (3,1).!4.\omega, \\ h(!3).h(!3).h(!1).? + (3,1).!4.\omega, \\ h(!2).h(!1).h(!3).h(!3).? + (3,1).!4.\omega, \\ h(!1).h(!2).h(!3).h(!2).? + (2,1).!3.\omega, \\ h(!2).h(!1).h(!3).h(\omega).h(!3).? + (3,1).!4.\omega \\ \ldots \end{array} \right\} \subset [\![f]\!]$$

$$y + z \,{<}y{<}\, ((2|3) \,{>}x{>}\, x) \,{<}z{<}\, 1$$



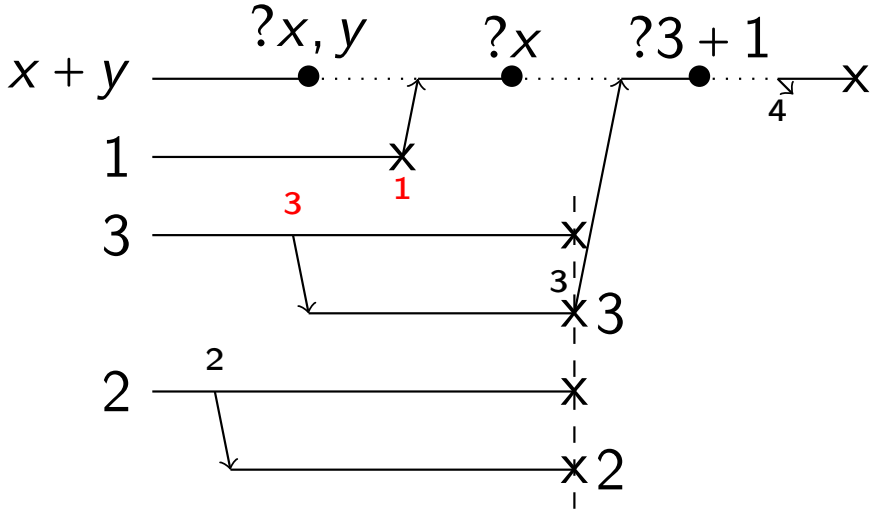
$$\left\{ \begin{array}{c} h(!1).h(!3).h(!3).?+(3,1).!4.\omega, \\ h(!3).h(!1).h(!3).?+(3,1).!4.\omega, \\ h(!3).h(!3).h(!1).?+(3,1).!4.\omega, \\ h(!2).h(!1).h(!3).h(!3).?+(3,1).!4.\omega, \\ h(!1).h(!2).h(!3).h(!2).?+(2,1).!3.\omega, \\ h(!2).h(!1).h(!3).h(\omega).h(!3).?+(3,1).!4.\omega \\ \ldots \end{array} \right\} \subset [\![f]\!]$$

$$y + z \,{<}y{<}\, ((2|3) \,{>}x{>}\, x) \,{<}z{<}\, 1$$



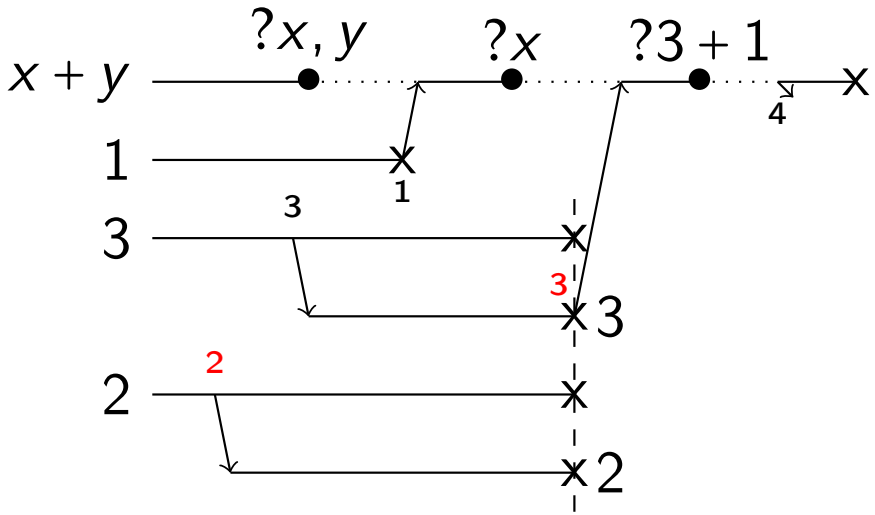
$$\left\{ \begin{array}{c} h(!1).h(!3).h(!3).? + (3,1).!4.\omega, \\ h(!3).h(!1).h(!3).? + (3,1).!4.\omega, \\ h(!3).h(!3).h(!1).? + (3,1).!4.\omega, \\ h(!2).h(!1).h(!3).h(!3).? + (3,1).!4.\omega, \\ h(!1).h(!2).h(!3).h(!2).? + (2,1).!3.\omega, \\ h(!2).h(!1).h(!3).h(\omega).h(!3).? + (3,1).!4.\omega \\ \ldots \end{array} \right\} \subset [\![f]\!]$$

$$y + z \,{<}y{<}\, ((2|3) \,{>}x{>}\, x) \,{<}z{<}\, 1$$



$$\left\{ \begin{array}{c} h(!1).h(!3).h(!3).? + (3,1).!4.\omega, \\ h(!3).h(!1).h(!3).? + (3,1).!4.\omega, \\ h(!3).h(!3).h(!1).? + (3,1).!4.\omega, \\ h(!2).h(!1).h(!3).h(!3).? + (3,1).!4.\omega, \\ h(!1).h(!2).h(!3).h(!2).? + (2,1).!3.\omega, \\ h(!2).h(!1).h(!3).h(\omega).h(!3).? + (3,1).!4.\omega \\ \ldots \end{array} \right\} \subset [\![f]\!]$$

$$y + z \,{<}y{<}\, ((2|3) \,{>}x{>}\, x) \,{<}z{<}\, 1$$



$$\left\{ \begin{array}{c} h(!1).h(!3).h(!3).? + (3,1).!4.\omega, \\ h(!3).h(!1).h(!3).? + (3,1).!4.\omega, \\ h(!3).h(!3).h(!1).? + (3,1).!4.\omega, \\ h(!2).h(!1).h(!3).h(!3).? + (3,1).!4.\omega, \\ h(!1).h(!2).h(!3).h(!2).? + (2,1).!3.\omega, \\ h(!2).h(!1).h(!3).h(\omega).h(!3).? + (3,1).!4.\omega \\ \ldots \end{array} \right\} \subset [\![f]\!]$$

What we need

Limits of SOS

- two many executions
- not enough information

Solution

- sequences not well suited
- causality is a partial order
- information on preemption

Labelled Asymmetric Event Structures

$(E, L, \leq, \nearrow, \Lambda)$

- E: set of *events*
- L: set of *labels*
- $\leq \in E^2$: *causality* (partial order)
- $\nearrow \in E^2$: *weak causality*
- $\Lambda : E \mapsto L$: *labelling function*
- $[e] = \{e' \in E | e' \leq e\}$ finite
- $e < e' \Rightarrow e \nearrow e'$
- $e \in E$, $\nearrow \cap [e]^2$ acyclic

- $e \leq e'$: e *always* happens *before* e'
- $e \nearrow e'$: e *never* happens *after* e'
- $e \| e'$: $\neg(e \nearrow e' \vee e' \nearrow e)$
- $e \rightsquigarrow e'$: $e \nearrow e' \wedge \neg(e \leq e')$
- $\#\{e_1, \ldots, e_n\}$: cycle in $\nearrow$

Plan

Instrumented executions

Labels on transitions

$\sigma \in [\![f]\!]_i, \sigma_i = (k_i, l_i, c_i, a_i)$
$\overline{\sigma} = (E, L, \leq, \nearrow, \Lambda)$

- k_i: unique identifier

$$E = \{k_0, \ldots, k_n\}$$

- l_i : label

$$L = \{l_0, \ldots, l_n\} \quad \Lambda(k_i) = l_i$$

- c_i : causes

$$k_i \leq k_j \Leftrightarrow k_i \in c_j$$

- a_i : weak causes

$$k_i \nearrow k_j \Leftrightarrow k_i \in a_j$$

Properties

Instrumentation

We only add information on the existing executions:

- $(\llbracket f \rrbracket_i)\,|_I = \{\sigma_1.I...\sigma_n.I | \sigma \in \llbracket f \rrbracket_i\}$
- $\forall f, (\llbracket f \rrbracket_i)\,|_I = \llbracket f \rrbracket$.

Correctness

Only correct behaviors can be infered from an execution

$$\forall f, \forall \sigma \in \llbracket f \rrbracket_i, \mathsf{Lin}(\overline{\sigma}) \subset \llbracket f \rrbracket.$$

Linearization $\Lambda(e_1)...\Lambda(e_n)$:

- left closed for causality
- respects weak-causality

The causality is not structural

$x <x< 1|2$

- $|$ encodes concurrency
- $<x<$ encodes preemption
- $x <x< 1|2$ encodes conflict

The causality is not structural

$x \,{<}x{<}\, 1|2$

- $|$ encodes concurrency
- ${<}x{<}$ encodes preemption
- $x \,{<}x{<}\, 1|2$ encodes conflict

stop ${<}x{<}\, 1?$

- $x+1 \,{<}x{<}\, 1$: $h(!1) \leq !2$

The causality is not structural

$x <x< 1|2$

- $|$ encodes concurrency
- $<x<$ encodes preemption
- $x <x< 1|2$ encodes conflict

stop $<x< 1$?

- $x+1 <x< 1$: $h(!1) \leq !2$
- $2 <x< 1$: $h(!1) \| !2$

The causality is not structural

$x <x< 1|2$

- $|$ encodes concurrency
- $<x<$ encodes preemption
- $x <x< 1|2$ encodes conflict

stop $<x< 1?$

- $x+1 <x< 1:\ h(!1) \leq !2$
- $2 <x< 1:\ h(!1) \| !2$
- **stop** $<x< 1$?

The causality is not structural

$x <x< 1|2$

- $|$ encodes concurrency
- $<x<$ encodes preemption
- $x <x< 1|2$ encodes conflict

stop $<x< 1?$

- $x+1 <x< 1:\ h(!1) \leq !2$
- $2 <x< 1:\ h(!1) \| !2$
- **stop** $<x< 1\ ?$
- $(\textbf{stop} <x< 1); 2:\ h(!1) \leq !2$

The causality is not structural

$x <x< 1|2$

- $|$ encodes concurrency
- $<x<$ encodes preemption
- $x <x< 1|2$ encodes conflict

stop $<x< 1?$

- $x+1 <x< 1:\ h(!1) \leq !2$
- $2 <x< 1:\ h(!1) || !2$
- **stop** $<x< 1$?
- $(\textbf{stop} <x< 1); 2:\ h(!1) \leq !2$
- $(\textbf{stop}; 2) <x< 1:\ h(!1) || !2$

The causal operator

$\langle f, c, a \rangle_L$

- f: a program
- c: its causes
- a: its weak causes
- L: a type of labels

$$\text{(CauseYes)}\ \frac{f \xrightarrow{k,l,c,a}_i f'}{\langle f, c', a' \rangle_L \xrightarrow{k,l,c \cup c', a \cup a' \cup c'}_i \langle f', c', a' \rangle_L}\ l \in L$$

$$\text{(CauseNo)}\ \frac{f \xrightarrow{k,l,c,a}_i f'}{\langle f, c', a' \rangle_L \xrightarrow{k,l,c,a}_i \langle f', c', a' \rangle_L}\ l \notin L$$

Semantics of the pruning operator

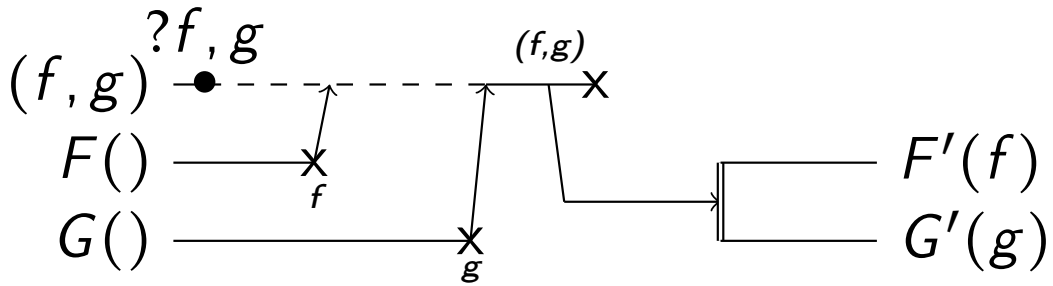
$$\text{(PruneLeft)}\ \frac{f \xrightarrow{k,l,c,a}_i f'}{f <x< g \xrightarrow{k,l,c,a}_i f' <x< g}\ l \neq \omega$$

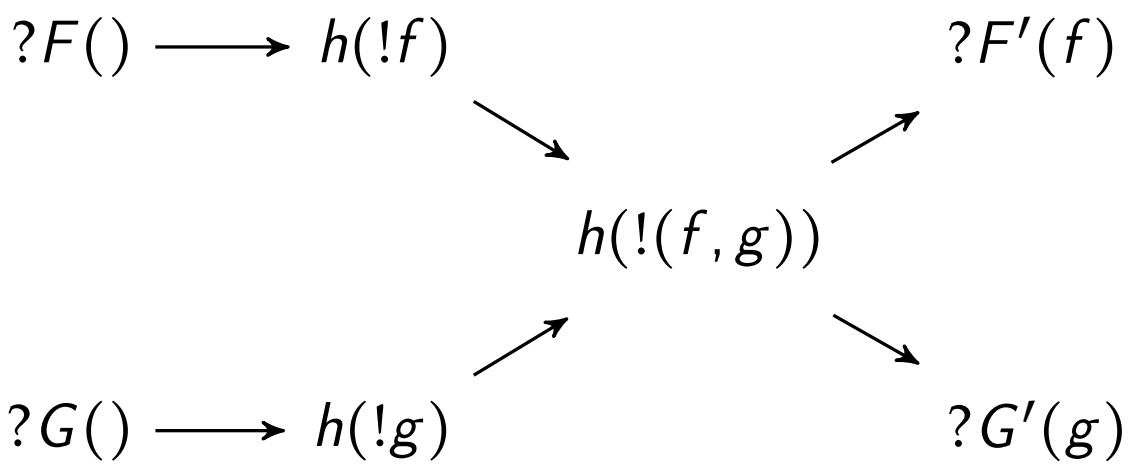
$$\text{(PruneN)}\ \frac{g \xrightarrow{k,n,c,a}_i g'}{f <x< g \xrightarrow{k,n,c,a}_i f <x< \langle g', \varnothing, \{k\}\rangle_{!v}}$$

$$\text{(PruneV)}\ \frac{g \xrightarrow{k,!v,c,a}_i g'}{f <x< g \xrightarrow{k,h(!v),c,a}_i \langle [\langle v, c \cup \{k\}, a\rangle_l / x] f, c \cup \{k\}, a\rangle_\omega}$$

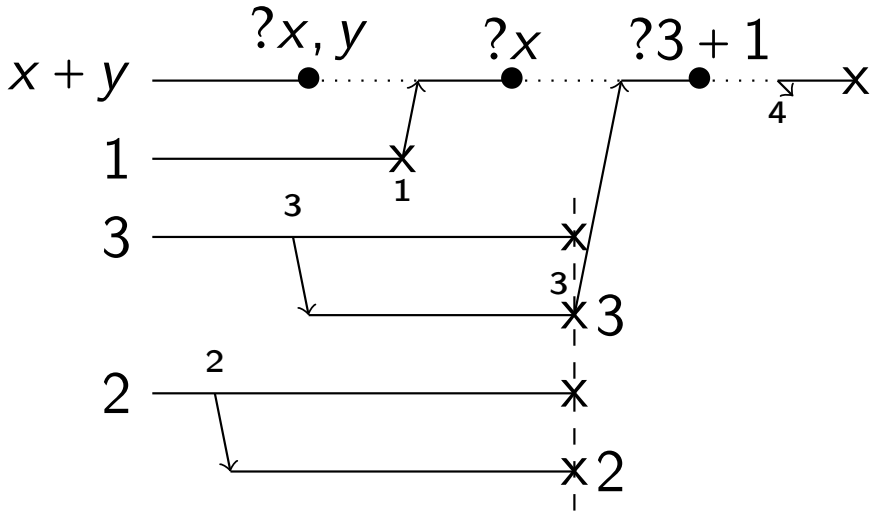
$$\text{(PruneStop)}\ \frac{g \xrightarrow{k,\omega,c,a}_i \bot}{f <x< g \xrightarrow{k,h(\omega),c,a}_i \langle [\langle \mathbf{stop}, c \cup \{k\}, a\rangle_l / x] f, c \cup \{k\}, a\rangle_\omega}$$

$$((f,g) < f < F() < g < G()) > (f,g) > (F'(f)|G'(g))$$





$y + z \,{<}y{<}\, ((2|3) \,{>}x{>}\, x) \,{<}z{<}\, 1$



$$h(!3) \rightarrow h(!3) \leftsquigarrow h(!2)$$

$$\downarrow$$

$$h(!1) \triangleright ?+(3,1) \triangleright \; !4 \longrightarrow \omega$$

Plan

The concurrent semantics

Goal

- unicity of the execution:

$$\forall \sigma_1, \sigma_2 \in [\![f]\!]_{ic}, \overline{\overline{\sigma_1}} \equiv \overline{\overline{\sigma_2}}$$

- correctness and completeness:

$$\forall \sigma \in [\![f]\!]_{ic}, \mathrm{Lin}(\overline{\overline{\sigma_1}}) = [\![f]\!]$$

Additional difficulties

- generation of the conflictual events
- a consequence can be followed by its weak causes

Conclusion

The Orc language

- too expressive : consensus needed
- semantics and implementation of the sites not defined
- hides the notion of process / place

The solution can be applied to other languages

Building a Concurrent Operational Semantics: the Example of the Orc Language

Matthieu Perrin Claude Jard Achour Mostefaoui

AeLoS / GDD
LINA, University of Nantes

AeLoS meeting

January, 9th 2014