Proof of the Instrumented Semantics for Orc

Matthieu Perrin, Claude Jard, Achour Mostefaoui

Proof of the Instrumented Semantics for Orc

Matthieu Perrin
LINA, University of Nantes
matthieu.perrin@univ-nantes.fr

Claude Jard
LINA, University of Nantes
claude.jard@univ-nantes.fr

Achour Mostéfaoui
LINA, University of Nantes
achour.mostefaoui@univ-nantes.fr

Abstract

This report shows how the operational semantics of a language like ORC can be instrumented so that the execution of a program produces information on the causal dependencies between events. The concurrent semantics we obtain is based on labeled asymetric event structures. This report contains the complete demonstration of correctness of this approach.

1 The Orc Core calculus

1.1 Syntax

$D \in$ Definition	$::=$	$\mathbf{def}\ y(\overline{x}) = f$
$f, g, h \in$ Expression	$::=$	$p \mid p(\overline{p}) \mid ?k \mid (f \mid g) \mid f >x> g \mid f <x< g \mid f; g \mid D \# f \mid \bot$
$v \in$ Orc Value	$::=$	$V \mid D$
$p \in$ Parameter	$::=$	$v \mid \mathbf{stop} \mid x$
$w \in$ Response	$::=$	$NT(v) \mid T(v) \mid Neg$
$n \in$ Non $-$ publication Label	$::=$	$V_k(\overline{v}) \mid D(\overline{p}) \mid \omega \mid h(\omega) \mid h(!v)$
$l \in$ Label	$::=$	$!v \mid n$

1.2 Semantics

(PUBLISH) $\dfrac{}{v \xrightarrow{!v} \mathbf{stop}}$ where v closed

(STOP) $\dfrac{}{\mathbf{stop} \xrightarrow{\omega} \bot}$

(DEFDECLARE) $\dfrac{[D/y]f \xrightarrow{l} f'}{D \# f \xrightarrow{l} f'}$ where D is $\mathbf{def}\ y(\bar{x}) = g$

(STOPCALL) $\dfrac{}{\mathbf{stop}(\bar{p}) \xrightarrow{\omega} \bot}$

(INTCALL) $\dfrac{}{D(\bar{p}) \xrightarrow{D(\bar{p})} [D/y][\bar{p}/\bar{x}]g}$ where D is $\mathbf{def}\ y(\bar{x}) = g$

(EXTCALL) $\dfrac{}{V(\bar{v}) \xrightarrow{V_k(\bar{v})} ?k}$ where $\bar{v}$ closed and k fresh

(ExtStop) $\dfrac{}{V(\bar{p}) \xrightarrow{\omega} \bot}$ where $\mathbf{stop} \in \bar{p}$

(NtRes) $\dfrac{?k \text{ receives } NT(v)}{?k \xrightarrow{!v} ?k}$

(TRes) $\dfrac{?k \text{ receives } T(v)}{?k \xrightarrow{!v} \mathbf{stop}}$

(NegRes) $\dfrac{?k \text{ receives } Neg}{?k \xrightarrow{\omega} \bot}$

(ParLeftStop) $\dfrac{f \xrightarrow{\omega} \bot}{f|g \xrightarrow{h(\omega)} g}$

(ParRightStop) $\dfrac{g \xrightarrow{\omega} \bot}{f|g \xrightarrow{h(\omega)} f}$

(ParLeft) $\dfrac{f \xrightarrow{l} f'}{f|g \xrightarrow{l} f'|g}$ where $l \neq \omega$

(ParRight) $\dfrac{g \xrightarrow{l} g'}{f|g \xrightarrow{l} f|g'}$ where $l \neq \omega$

(SeqStop) $\dfrac{f \xrightarrow{\omega} \bot}{f >x> g \xrightarrow{\omega} \bot}$

(SeqN) $\dfrac{f \xrightarrow{n} f'}{f >x> g \xrightarrow{n} f' >x> g}$ where $n \neq \omega$

(SeqV) $\dfrac{f \xrightarrow{!v} f'}{f >x> g \xrightarrow{h(!v)} (f' >x> g)|[v/x]g}$

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

(PruneLeft) $\dfrac{f \xrightarrow{l} f'}{f <x< g \xrightarrow{l} f' <x< g}$ where $l \neq \omega$

(PruneN) $\dfrac{g \xrightarrow{n} g'}{f <x< g \xrightarrow{n} f <x< g'}$ where $n \neq \omega$

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

(OtherStop) $\dfrac{f \xrightarrow{\omega} \bot}{f; g \xrightarrow{h(\omega)} g}$

(OtherN) $\dfrac{f \xrightarrow{n} f'}{f; g \xrightarrow{n} f'; g}$

(OTHERV) $\dfrac{f \xrightarrow{!v} f'}{f;g \xrightarrow{!v} f'}$

1.3 The Instrumented semantics

$$\begin{array}{rcl} \langle p, c_l, c_\omega, a_v\rangle.v & = & p.v \\ v.v & = & v \\ \langle p, c_l, c_\omega, a_v\rangle.c & = & p.c \cup c_l \\ v.c & = & \emptyset \\ \langle p, c_l, c_\omega, a_v\rangle.\omega & = & p.c \cup c_l \cup c_\omega \\ v.\omega & = & \emptyset \end{array}$$

(PUBLISH) $\dfrac{}{v \xrightarrow{k,!v,\emptyset,\emptyset} \langle \mathbf{stop}, \emptyset, \{k\}, \emptyset\rangle}$ where v closed and k fresh

(STOP) $\dfrac{}{\mathbf{stop} \xrightarrow{k,\omega,\emptyset,\emptyset} \bot}$ where k fresh

(DEFDECLARE) $\dfrac{[D/y]f \xrightarrow{k,l,c,a} f'}{D\#f \xrightarrow{k,l,c,a} \langle f', \{k\}, \emptyset, \emptyset\rangle}$ where D is **def** $y(\bar{x}) = g$

(STOPCALL) $\dfrac{}{P(\overline{p}) \xrightarrow{k,\omega,P.\omega,P.\omega} \bot}$ where $P.v = \mathbf{stop}$ and k fresh

(INTCALL) $\dfrac{}{P(\overline{p}) \xrightarrow{k,D(\bar{p}),P.c,P.c} \langle [D/y][\bar{p}/\bar{x}]g, \{k\}, \emptyset, \emptyset\rangle}$ where $P.v = D$ is **def** $y(\bar{x}) = g$ and k fresh

(EXTCALL) $\dfrac{}{P(\overline{p}) \xrightarrow{k,V_k(\bar{v}),C,C} \langle ?k, \{k\}, \emptyset, \emptyset\rangle}$ where $\overline{p.v}$ closed, $p.v = V$, $C = P.c \cup \bigcup_{p\in\overline{p}} p.c$ and k fresh

(EXTSTOP) $\dfrac{}{P(\overline{p}) \xrightarrow{k,\omega,C,C} \bot}$ where $\mathbf{stop} \in \overline{p.v}$, k fresh and $x \notin \overline{p.v}$ and $P.v = V$ and $C = \overline{p.\omega} \cup P.\omega$

(NTRES) $\dfrac{?k \text{ receives } NT(v,c,a)}{?k \xrightarrow{j,!v,c,a\cup c} \langle ?k, \emptyset, \{j\}, \emptyset\rangle}$ where j fresh

(TRES) $\dfrac{?k \text{ receives } T(v,c,a)}{?k \xrightarrow{j,!v,c,a\cup c} \langle \mathbf{stop}, \emptyset, \{j\}, \emptyset\rangle}$ where j fresh

(NEGRES) $\dfrac{?k \text{ receives } Neg(c,a)}{?k \xrightarrow{j,\omega,c,a\cup c} \bot}$ where j fresh

(PARLEFTSTOP) $\dfrac{f \xrightarrow{k,\omega,c,a} \bot}{f|g \xrightarrow{k,h(\omega),c,a} \langle g, \emptyset, \{k\}, \emptyset\rangle}$

(PARRIGHTSTOP) $\dfrac{g \xrightarrow{k,\omega,c,a} \bot}{f|g \xrightarrow{k,h(\omega),c,a} \langle f, \emptyset, \{k\}, \emptyset\rangle}$

(PARLEFT) $\dfrac{f \xrightarrow{k,l,c,a} f'}{f|g \xrightarrow{k,l,c,a} f'|g}$ where $l \neq \omega$

$$\text{(ParRight)}\ \frac{g \xrightarrow{k,l,c,a} g'}{f|g \xrightarrow{k,l,c,a} f|g'} \text{ where } l \neq \omega$$

$$\text{(SeqStop)}\ \frac{f \xrightarrow{k,\omega,c,a} \bot}{f >x> g \xrightarrow{k,\omega,c,a} \bot}$$

$$\text{(SeqN)}\ \frac{f \xrightarrow{k,n,c,a} f'}{f >x> g \xrightarrow{k,n,c,a} f' >x> g} \text{ where } n \neq \omega$$

$$\text{(SeqV)}\ \frac{f \xrightarrow{k,!v,c,a} f'}{f >x> g \xrightarrow{k,h(!v),c,a} (f' >x> g)|\langle [v/x]g, \{k\}, \emptyset, \emptyset\rangle}$$

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

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

$$\text{(PruneN)}\ \frac{g \xrightarrow{k,n,c,a} g'}{f <x< g \xrightarrow{k,n,c,a} f <x< \langle g', \emptyset, \emptyset, \{k\}\rangle} \text{ where } n \neq \omega$$

$$\text{(PruneV)}\ \frac{g \xrightarrow{k,!v,c,a} g'}{f <x< g \xrightarrow{k,h(!v),c,a} \langle [\langle v, \{k\}, \emptyset, \emptyset\rangle/x]f, \emptyset, \{k\}, \emptyset\rangle}$$

$$\text{(OtherStop)}\ \frac{f \xrightarrow{k,\omega,c,a} \bot}{f; g \xrightarrow{k,h(\omega),c,a} \langle g, \{k\}, \emptyset, \emptyset\rangle}$$

$$\text{(OtherN)}\ \frac{f \xrightarrow{k,n,c,a} f'}{f; g \xrightarrow{k,n,c,a} f'; g}$$

$$\text{(OtherV)}\ \frac{f \xrightarrow{k,!v,c,a} f'}{f; g \xrightarrow{k,!v,c,a} f'}$$

$$\text{(CausalStop)}\ \frac{f \xrightarrow{k,\omega,c,a} \bot}{\langle f, c_l, c_\omega, a_v\rangle \xrightarrow{k,\omega,c\cup c_l\cup c_\omega,a\cup c_l\cup c_\omega} \bot}$$

$$\text{(CausalN)}\ \frac{f \xrightarrow{k,n,c,a} f'}{\langle f, c_l, c_\omega, a_v\rangle \xrightarrow{k,n,c\cup c_l,a\cup c_l} \langle f', c_l, c_\omega, a_v\rangle} \text{ where } n \neq \omega$$

$$\text{(CausalV)}\ \frac{f \xrightarrow{k,!v,c,a} f'}{\langle f, c_l, c_\omega, a_v\rangle \xrightarrow{k,!v,c\cup c_l,a\cup c_l\cup a_v} \langle f', c_l, c_\omega, a_v\rangle}$$

2 Concurrent executions

In all the remaining, we fix f_0 to be an orc program.

Definition 1 (Labelled asymmetric event structure)**.** *A* labelled asymmetric event structure *is a tuple* $(E, L, \leq, \nearrow, \Lambda)$ *where*

- E *is a set of* events*,*
- L *is a set of* labels*,*
- $\leq$*, the* causality *is a partial order on* E*,*

- $\nearrow$, *the* weak causality *is a binary relation on* E,
- $\Lambda : E \mapsto L$ *is the labelling function*
- *each event* $e \in E$ *has a finite* causal history $[e] = \{e' \in E | e' \leq e\}$,
- *for all events* $e < e' \in E, e \nearrow e'$, *where* $<$ *is the irreflexive restriction of* $\leq$,
- *for all* $e \in E$, $\nearrow \cap [e]^2$ *is acyclic.*

If $(E, L, \leq, \nearrow, \Lambda)$ *is an asymmetric event structure, for* $e, e' \in E$, *we say that:*

- e *is* prehempted *by* e', *denoted* $e \rightsquigarrow e'$, *if* $e \nearrow e'$ *and* $e \not\leq e'$,
- e *and* e' *are* concurrent, *denoted* $e||e'$, *if* $\neg e \nearrow e'$ *and* $\neg e' \nearrow e$.

We also define an induced conflict *relation* $\#_a$ *as the smallest set of finite parts of* E *such that, for* $E' \subset E$ *and* $e_0, e_1, ..., e_n \in E$,

- *if* $e_0 \nearrow e_1 \nearrow \cdots \nearrow e_n \nearrow e_0$ *then* $\{e_0, e_1, \cdots, e_n\} \in \#_a$
- *if* $E' \cup \{e_0\} \in \#_a$ *and* $e_0 \leq e_1$ *then* $E' \cup \{e_1\} \in \#_a$.

Definition 2 (Linearization)**.** *Let* $(E, L, \leq, \nearrow, \Lambda)$ *be a labelled asymmetric event system. A* finite linearization *of* $(E, L, \leq, \nearrow, \Lambda)$ *is a word* $w = \Lambda(e_0)...\Lambda(e_n)$ *where the* $e_i \in E$ *are pairwise distincts such that:*

- *it is left-closed for causality, ie* $\forall e \in E, \forall e' \in \{e_0, ..., e_n\}, e \leq e' \Rightarrow e \in \{e_0, ..., e_n\}$,
- *the weak causality is respected, ie* $\forall e_i, e_j \in \{e_0, ..., e_n\}, e_i \nearrow e_j \Rightarrow i < j$

We denote $Lin(E, L, \leq, \nearrow, \Lambda)$ *as the set of all the linearizations of* $(E, L, \leq, \nearrow, \Lambda)$.

Definition 3 (Event)**.** *The transitions of the instrumented semantics are labelled by tuples* $e = (k, l, c, a, b)$ *where* k *is the identifier of the transition taken in a countable set* K, $l \in L$ *is a label of the original semantics and* c, a *and* b *are finite sets of identifiers.*
$\mathcal{E} = K \times L \times \mathbf{2}^K_{fin} \times \mathbf{2}^K_{fin} \times \mathbf{2}^K_{fin}$ *is the set of all the events.*
We introduce the notations $e_k = k$, $e_l = l$, $e_c = c$, $e_a = a$ *and* $e_b = b$.

Definition 4 (Sequential execution)**.** *A* sequential execution *is a sequence of labels given by the instrumented semantics. More formally, an execution is a sequence* $e^1...e^n$ *such that there is* $f_1...f_n$ *such that* $f_0 \xrightarrow{e^1} f_1 \xrightarrow{e^2} ... \xrightarrow{e^n} f_n$.

Definition 5 (Concurrent execution)**.** *We considere a sequential execution* $e^0...e^n$. *We define its* concurrent execution *as the triple* $(E, \leq, \nearrow)$ *where:*

- $E = \{e^0, ..., e^n\}$,
- $\leq_E$ *is the transitive and reflexive closure of* $\{(e, e') \in E^2 | e_k \in e'_c\}$,
- $\nearrow_E$ *is the transitive closure of* $\{(e, e') \in E^2 | e_k \in e'_a\}$.

Property 1. *A concurrent execution* $(E, \leq, \nearrow)$ *is an asymmetric event structure.*

Proof. Let $e^0...e^n$ be a sequential execution and $e^i \mapsto e^j$ denote the fact that $i \leq j$. Let $(E, \leq, \nearrow)$ be the sequential execution obtained from it.

Let us first remark that in all the rules, the c or a part of a label is created as a union of the same parts in the premise and some arguments of the $\langle f, c_l, c_w, a_v \rangle$ construction. As f_0 is not instrumented, this construction is only built with the identifier of the the rule that made it up, or left unchanged, so c and a contain only identifiers of previous steps. Moreover, anything that is added into c is also added into a. It remains true for the transitive closures, so $< \subset \nearrow \subset \mapsto$.

It follows that $\leq$ is antisymmetric. As it is also reflexive and transitive by construction, it is a partial order. Moreover, if $e' \leq e$, then $e' \mapsto e$. As there are a finite such e', $[e]$ is finite.

At last, $\nearrow$ is also a sub-relation of $\mapsto$, which is acyclic, so $\nearrow$ is acyclic and $\nearrow_{[e]}$ is acyclic for all e.

Finally, $(E, \leq, \nearrow)$ is an asymmetric event structure. □

Remark 1. *The fact that* $\nearrow$ *is acyclic means that we will never detect conflicts, i.e* $\#_a = \emptyset$. *This is due to the fact that we only considere one sequential execution of the program, so all the events we considere occure together. For exemple, the program* $x = x = 1|2$, *can generate two concurrent executions, one containing the publication* !1 *and one containing* !2. *As they are not in the same execution, they cannot be in conflict.*

3 Instrumentation

Remark 2. *We omit the proofs for the external site calls because it is missing formalisation.*

It is actually much more complicated if we want to deal with sites properly, because an expression may not allow the same transitions depending on an unspecified context. In other words, the semantics does not define a transitions system.

In particular, lemmas 4 and **??** *are not sufficient to prove the theorem. Considere the expression*

$$write(0); (read() <x< (write(1)|1)).$$

The second call to $write$ *is prehempted by the internal publication of* 1*, but after this publication, the program is* $read()$*. However, the next transition can be* !0 *or* !1 *depending on what happened before.*

Property 2 (Conformity to the original semantics)**.** *There is a sequential execution $e^1...e^n$ in the instrumented semantics if and only if $e^1_l...e^n_l$ is an execution in the original semantics.*

Proof. For an instrumented orc program F, we define $[\![F]\!]$ as the same expression in which the angle brackets are removed, i.e :

$$\begin{array}{rcl}
[\![\langle f, c_l, c_\omega, a_v\rangle]\!] & = & [\![f]\!] \\
[\![\mathbf{def}\ y(\overline{x}) = f]\!] & = & \mathbf{def}\ y(\overline{x}) = [\![f]\!] \\
[\![p]\!] & = & p \\
[\![p(\overline{q})]\!] & = & [\![p]\!](\overline{[\![q]\!]}) \\
[\![?k]\!] & = & ?k \\
[\![f|g]\!] & = & [\![f]\!]|[\![g]\!] \\
[\![f >x> g]\!] & = & [\![f]\!] >x> [\![g]\!] \\
[\![f <x< g]\!] & = & [\![f]\!] <x< [\![g]\!] \\
[\![f; g]\!] & = & [\![f]\!]; [\![g]\!] \\
[\![D\#f]\!] & = & [\![D]\!]\#[\![f]\!] \\
[\![\bot]\!] & = & \bot.
\end{array}$$

One step. Let F be an instrumented program. We show that there is an instrumented program F' and an event e such that $F \xrightarrow{e} F'$ if and only if $[\![F]\!] \xrightarrow{e_l} [\![F']\!]$. We show the implication and its converse by induction on the derivation trees.

Implication. Suppose $F \xrightarrow{e} F'$.

If the step was defined by an axiom, then an axiom of the same name exists in the original semantics and its application is $[\![F]\!] \xrightarrow{e_l} [\![F']\!]$. For example, if the rule is PUBLISH, the step is $v \xrightarrow{k:!v,\emptyset,\emptyset} \langle \mathbf{stop}, \emptyset, \{k\}\rangle$ and $v \xrightarrow{!v} \mathbf{stop}$ is a valid step in the original semantics.

Otherwise, if F is not of the form $\langle f, c_l, c_\omega, a_v\rangle$, the step was generated by a rule with a unique premise $f \xrightarrow{e'} f'$. By induction, $[\![f]\!] \xrightarrow{e'_l} [\![f']\!]$ is a valid step in the original semantics, so the rule with the same name can be applied in the original semantics, and $[\![F]\!] \xrightarrow{e_l} [\![F']\!]$.

The last case is if $F = \langle f, c_l, c_\omega, a_v\rangle$, so the rule is one of CAUSALSTOP, CAUSALN or CAUSALV. As above, the premise corresponds to a valid step $[\![f]\!] \xrightarrow{e'_l} [\![f']\!]$ in the original semantics. As $[\![F]\!] = [\![f]\!]$, $[\![F']\!] = [\![f']\!]$ and $e_l = e'_l$, we have $[\![F]\!] \xrightarrow{e_l} [\![F']\!]$.

Converse. Suppose $[\![F]\!] \xrightarrow{l} f'$.

There is a finite $m \in \mathbb{N}$ such that $F = \langle\langle...\langle f, c^m_l, c^m_\omega\rangle...\rangle, c^1_l, c^1_\omega\rangle$ where f is not of the form $\langle g, c_l, c_\omega\rangle$. Moreover, $[\![f]\!] = [\![F]\!] \xrightarrow{l} f'$.

We now considere the step $[\![f]\!] \xrightarrow{l} f'$. If it was generated by an axiom, then we can apply the axiom with the same name applies on f to get $f \xrightarrow{e} F'$ with $e_l = l$ and $[\![F']\!] = f'$.

Otherwise, the rule used for the step has a unique premise $g \xrightarrow{l'} g'$. Moreover, the left hand side of the premise of the rule with the same name is G with $[\![G]\!] = g \xrightarrow{l'} g'$. By induction, there are e' and G' with $e'_l = l'$ and $[\![g']\!] = g'$ such that $G \xrightarrow{e'} G'$, so we can apply the rule in the instrumented semantics, which gives us the expected e and F'.

Finally, we can apply m times the rule CAUSALSTOP, CAUSALN or CAUSALV depending on the label, and get $F \xrightarrow{e''} H = \langle\langle...\langle F', c^m_l, c^m_\omega\rangle...\rangle, c^1_l, c^1_\omega\rangle$, with $e''_l = l$ and $[\![H]\!] = [\![F']\!] = f'$.

Whole execution. It remains to show the property on whole executions. We will prove not only that the labels are the same, but also that the intermediate expressions correspond. We will prove both the induction and its converse by induction on n, the length of the execution.

Implication. Suppose there are instrumented expressions $F_1, ..., F_n$ and events $e^1, ..., e^n$ such that $f_0 \xrightarrow{e^1} F_1 ... \xrightarrow{e^n} F_n$. We show by induction on n that $f_0 = [\![f_0]\!] \xrightarrow{e_l^1} [\![F_1]\!] ... \xrightarrow{e_l^n} [\![F_n]\!]$.

It is clear for $n = 0$. Just notice that f_0 is not instrumented, so $f_0 = [\![f_0]\!]$. Suppose the property is true for n and $f_0 \xrightarrow{e^1 ... e^n} F_n \xrightarrow{e^{n+1}} F_{n+1}$. By induction, $f_0 \xrightarrow{e_l^1 ... e_l^n} [\![F_n]\!]$ and by what precedes, $[\![F_n]\!] \xrightarrow{e_l^{n+1}} [\![F_{n+1}]\!]$, so $f_0 \xrightarrow{e_l^1 ... e_l^{n+1}} [\![F_{n+1}]\!]$.

Converse. Suppose there are orc expressions $f_1, ..., f_n$ and labels $l^1, ..., l^n$ such that $f_0 \xrightarrow{l^1} f_1 ... \xrightarrow{l^n} f_n$. We show by induction on n that there are instrumented expressions $F_1, ..., F_n$ and events $e^1, ..., e^n$ with $[\![F_i]\!] = f_i$ and $e_l^i = l^i$ for all i such that $f_0 \xrightarrow{e^1} F_1 ... \xrightarrow{e^n} F_n$.

It is clear for $n = 0$ as $f_0 = [\![f_0]\!]$. Suppose the property is true for n and $f_0 \xrightarrow{l^1 ... l^{n+1}} l_{n+1}$. By induction, $f_0 \xrightarrow{e^1 ... e^n} F_n$ and by what precedes, $F_n \xrightarrow{e^{n+1}} F_{n+1}$, so $f_0 \xrightarrow{e^1 ... e^{n+1}} F_{n+1}$ with for all i, $[\![F_i]\!] = f_i$ and $e_l^i = l^i$.

□

4 Correctness

Definition 6 (Conform linearization)**.** *Let $(E, \leq, \nearrow)$ be an asymmetric event system. A linearization $(E', \mapsto)$ is said to be* conform *to E if:*

- *E' is left-closed for causality, ie $\forall e \in E, \forall e' \in E', e \leq e' \Rightarrow e \in E'$,*
- *the weak causality is respected, ie $\forall e, e' \in E', e \nearrow e' \Rightarrow e \mapsto e'$.*

Definition 7 (Feasible linearization)**.** *Let $(E, \leq, \nearrow)$ be an asymmetric event system.*

Let E be a concurrent execution and $(E', \mapsto)$ be a linearization of E.

E' is not left-closed regarding to weak causality, in the sens that there can be $e \in E \setminus E'$ and $e' \in E'$ such that $e \nearrow e'$. We introduce $\pi^a_{E'}$, as the projection that remove the weak causes of events that are not in E': $\pi^a_{E'}(k, l, c, a) = (k, l, c, \{k' \in a | \exists e' \in E', e'_k = k'\})$.

$(E', \mapsto)$ is said to be feasible *if there is a sequential execution $\pi^a_{E'}(e^0)...\pi^a_{E'}(e^n)$ such that $E' = \{e^0, ..., e^n\}$ with $e^0 \mapsto ... \mapsto e^n$.*

We will now prove that the linearisations of E that are conform to E are exactly those that are feasible. We need a few other definitions and lemma in order to prove this theorem (theorem 1).

Definition 8 (Equivalence of programs)**.** *Let F and F' be two instrumented orc programs. They are equivalent, denoted $F \equiv F'$, if the have the same sequential executions regardless the values of the variables, ie for all $x_1, ..., x_m$ and $f_1, ..., f_m$, there is $F_1, ..., F_n$ and $e_1, ...e_n$ such that $[f_1/x_1]...[f_m/x_m]F \xrightarrow{e_1} F_1 ... \xrightarrow{e_n} F_n$ if and only if there is $F'_1, ..., F'_n$ such that $[f_1/x_1]...[f_m/x_m]F' \xrightarrow{e_1} F'_1 ... \xrightarrow{e_n} F'_n$.*

Lemma 1. *We prove some properties on the equivalence relation that will be usefull later:*

- *for all programs f, g, h, $f|g \equiv g|f$ and $(f|g)|h \equiv f|(g|h)$*
- *for all program f and all sets of identifiers $c_l, c'_l, c_\omega, c'_\omega, a_v, a'_v$, $\langle\langle f, c'_l, c'_\omega, a'_v\rangle, c_l, c_\omega, a_v\rangle \equiv \langle\langle f, c_l, c_\omega, a_v\rangle, c'_l, c'_\omega, a'_v\rangle$*
- *if $f \equiv f'$ and g are instrumented orc programs, then $f|g \equiv f'|g$, $f >x> g \equiv f' >x> g$, $f <x< g \equiv f' <x< g$ and $f; g \equiv f'; g$*

Proof. Let $x_1, ..., x_m$ and $f_1, ..., f_m$ be variables and programs. For an orc program f, let $[]f$ denote $[f_1/x_1]...[f_m/x_m]F$.

Let F, F' be two instrumented orc programs, and let $P_n(F, F')$ denote, for all n, that there are n labels $e_1, ..., e_n \in \mathcal{E}$ and n instrumented orc programs $F_1, ..., F_n$ such that $[]F \xrightarrow{e_1} F_1 \xrightarrow{e_2} ... \xrightarrow{e_n} F_n$ if and only if there are $F'_1, ..., F'_n$ such that $[]F' \xrightarrow{e_1} F'_1 \xrightarrow{e_2} ... \xrightarrow{e_n} F'_n$.

For all F, F', $P_0(F, F')$ is true.

Suppose, for all f and g, $P_n(f|g, g|f)$ for a given $n \geq 0$. Let f, g be two instrumented orc programs such that $[](f|g) = []f|[]g \xrightarrow{e} F'$. Only four rules are able to generate this step

PARLEFT we have $[]f \xrightarrow{e} f'$ and $[]f|[]g \xrightarrow{e} f'|[]g$. So by PARRIGHT, $[]g|[]f \xrightarrow{e} []g|f'$. As $P_n(f'|g, f|g')$, we have $P_{n+1}(f|g, f|g)$.

PARLEFTSTOP we have $[]f \xrightarrow{e'=(e_k,\omega,e_c,e_a)} \bot$ and $[]f|[]g \xrightarrow{e} \langle []g, \emptyset, \{e_k\}, \emptyset = F$. So by PARRIGHTSTOP, $[]g|[]f \xrightarrow{e} F$, and $P_{n+1}(f|g, g|f)$.

PARRIGHT and PARRIGHTSTOP are symmetric.

The converse is true by symmetry

The proofs for $P_{n+1}((f|g)|h, f|(g|h))$ and $P_{n+1}(\langle\langle f, c'_l, c'_\omega, a'_v\rangle, c_l, c_\omega, a_v\rangle, \langle\langle f, c_l, c_\omega, a_v\rangle, c'_l, c'_\omega, a'_v\rangle$, as well as there converses are very similar, which proves the first two points.

Suppose there is an n such that for all instrumented orc programs $f \equiv f'$ and g, $P_n(f|g, f'|g)$. Let $f \equiv f'$ and g be instrumented orc programs such that $[](f|g) \xrightarrow{e} F$. There are four possibilities for this transition:

PARLEFT we have $[]f \xrightarrow{e} h$, so $[]f' \xrightarrow{e} h'$ with $h \equiv h'$, and by PARLEFT, $[](f'|g) \xrightarrow{e} h'|[]g$. As, by induction, $P_n(h|g, h'|g)$, we have $P_{n+1}(f|g, f'|g)$.

PARLEFTSTOP we have $[]f \xrightarrow{e'=(e_k,\omega,e_c,e_a)} \bot$, so $[]f' \xrightarrow{e'=(e_k,\omega,e_c,e_a)} \bot$, so PARLEFTSTOP gives $[](f'|g) \xrightarrow{e} []g$ and $P_n(h|g, h'|g)$

PARRIGHT we have $[]g \xrightarrow{e} g'$, so by PARLEFT, $[](f'|g) \xrightarrow{e} []f'|g'$ and the end by induction.

PARRIGHTSTOP we have $[]g \xrightarrow{e'} \bot$, so by PARRIGHTSTOP, $[](f'|g) \xrightarrow{e} []f'$ and the end by induction.

The converse and the proofs for the sequence and otherwise are very similar. We can prove similarly that $\langle f, c_l, c_\omega, a_v\rangle \equiv \langle f', c_l, c_\omega, a_v\rangle$.

For $f < x < g \equiv f' < x < g$, the arguments are the same for rules PRUNEN and PRUNELEFT. For rule PRUNEV, we have $[]g \xrightarrow{e'} g'$, so by rule PRUNEV, $[](f' < x < g) \xrightarrow{e} \langle[\langle v, \{e_k\}, \emptyset, \emptyset\rangle][]f', \emptyset, \{e_k\}, \emptyset\rangle$. We have $[\langle v, \{e_k\}, \emptyset, \emptyset\rangle][]f' \equiv [\langle v, \{e_k\}, \emptyset, \emptyset\rangle][]f$ by definition of the equivalence, so $\langle[\langle v, \{e_k\}, \emptyset, \emptyset\rangle][]f', \emptyset, \{e_k\}, \emptyset\rangle \equiv \langle[\langle v, \{e_k\}, \emptyset, \emptyset\rangle][]f, \emptyset, \{e_k\}, \emptyset\rangle$, and $P_{n+1}(f < x < g, f' <x< g)$. It is similar for rule PRUNESTOP.

□

Lemma 2. *For all instrumented Orc programs F, F', F'' and for all events $A = (k, !v, c, a)$ and $B = (k', \omega, c', a')$ such that $F \xrightarrow{A} F' \xrightarrow{B} F''$, we have $A \leq B$.*

Proof. We show this lemma by induction on the syntax of F. For the base cases, the only possibilities to publish a value are the rules PUBLISH, DEFDECLARE, NTRES and TRES. All this rules instrument F' with $\{k\}$, so any further step that produces ω must be generated by CAUSALSTOP, so we have $A \leq B$.

Here is the list of the rules that can publish a value, and thus stand for A, for the induction cases:

PARLEFT, PARRIGHT: the following step can only be produced by PARLEFT, PARRIGHT, PARLEFTSTOP or PARRIGHTSTOP, none of which being able to produce B.

PRUNELEFT: the following step can only be produced by PRUNESTOP, PRUNELEFT, PRUNEN or PRUNEV, none of which being able to produce B.

OTHERV: if $F = f; g \xrightarrow{A} F'$ was generated by OTHERV, then the premise $f \xrightarrow{A} F'$ is true, and we have $f \xrightarrow{A} F' \xrightarrow{B} F''$. By induction, we have $A \leq B$.

CAUSAL: the last possibility is $F = \langle f, c_l, c_\omega\rangle \xrightarrow{A} F' = \langle f', c_l, c_\omega\rangle \xrightarrow{B} F'' = \langle f'', c_l, c_\omega\rangle$. The premises give $f \xrightarrow{A'=(k,!v,c_0,a)} f' \xrightarrow{B'=(k',\omega,c'_0,a')} f''$ with $c = c_0 \cup c_l$ and $c' = c'_0 \cup c_l \cup c_\omega$. By induction, we have $A' \leq B'$, so $A \leq B$.

□

Lemma 3. *Let F, F' be instrumented program, p be the parameter v or* **stop**, *x be a variable, c be a set of identifiers and e be an event. For a program f, let $f[p] = [\langle p, c, \emptyset, \emptyset\rangle/x]f$.*

- *If $F \xrightarrow{e} F'$ then $F[p] \xrightarrow{e} F'[p]$.*
- *If $F[p] \xrightarrow{e} F'[p]$ then $F \xrightarrow{e} F'$ or $c \subset e_c$.*

Proof. Suppose $F \xrightarrow{e} F'$. We will prove by induction on the derivation tree of the step that $F[p] \xrightarrow{e} F'[p]$.

PUBLISH, STOP, NTRES, TRES, NEGRES: $F = F[p]$, so the property is true.

DEFDECLARE: $F = D\#f$ and $[D/y]f \xrightarrow{e} f'$, so by induction $([D/y]f)[p] \xrightarrow{e} f'[p]$. As y is bound in f, we have $[D[p]/y]f[p] = ([D/y]f)[p]$, so $F[p] \xrightarrow{e} \langle f'[p], \{e_k\}, \emptyset, \emptyset\rangle = F'[p]$.

other cases: In all the other cases, it is easy to see by induction that the same step can be achieved.

Conversely, suppose $F \xrightarrow{e} F'$. We will prove by induction on the derivation tree of the step that $F[p] \xrightarrow{e} F'[p]$.

□

Lemma 4. *Let F, F' and F'' be instrumented orc programs and $e||e'$ be two events of the instrumented semantics such that $F \xrightarrow{e} F' \xrightarrow{e'} F''$. Then there are two instrumented program G, G' such that $F \xrightarrow{e'} G \xrightarrow{e} G' \equiv F''$.*

Proof. Let F, F' and F'' be instrumented orc programs and $e||e'$ be two events of the instrumented semantics such that $F \xrightarrow{e} F' \xrightarrow{e'} F''$.

We will prove this lemma by induction on F. The principle of this proof is to enumerate all the possibilities for the pair of rules that where able to generate e and e'. As this is an inductive proof, we should differentiate the base cases and the induction cases. However, by sake of clarity, we give a unique list of possibilities. The first-level items are for rules that can generate e and the second-level items are related to e'. Of course, we never use the induction hypothesis on case bases.

PUBLISH, DEFDECLARE, INTCALL, EXTCALL, OTHERSTOP: the following step must be CAUSALN or CAUSALSTOP, but then $e \leq e'$, which is not possible.

STOP, STOPCALL, EXTSTOP, SEQSTOP, CAUSALSTOP: there is no possible following step either.

PARLEFTSTOP (resp. PARRIGHTSTOP): $F = f|g$ (resp. $F = g|f$), $F' = \langle g, \emptyset, \{k\}, \emptyset\rangle$ and $e_l = h(\omega)$. As $\neg e \nearrow e'$, e' can only be an application of rule CAUSALN, with premise $g \xrightarrow{e'=(k',n',c',a')} g'$ where $n' \neq \omega$. Thus, rule PARRIGHT (resp. PARLEFT) can be applied, giving $F \xrightarrow{e'} G = f|g'$ (resp $g'|f$), then rule PARLEFTSTOP (resp. PARRIGHTSTOP), giving $G \xrightarrow{e} F'' = \langle g', \emptyset, \{k\}, \emptyset\rangle$.

PARLEFT (the same arguments hold for PARRIGHT): $F = f|g \xrightarrow{e} f'|g = F'$ with $e_l \neq \omega$. The second step can be generated by four different rules:

- **PARRIGHT:** we have $g \xrightarrow{e'} g'$, so $F = f|g \xrightarrow{e'} f|g' \xrightarrow{e} F'' = f'|g'$.
- **PARLEFT:** we have $f \xrightarrow{e} f' \xrightarrow{e'} f''$ with $e||e'$, so by induction, there are $h, h' \equiv f''$ such that $f \xrightarrow{e'} h \xrightarrow{e} h'$. We can apply PARLEFT twice to obtain $F = f|g \xrightarrow{e'} h|g \xrightarrow{e} h'|g \equiv F''$.
- **PARRIGHTSTOP:** we have $g \xrightarrow{e'} \bot$ and $e'_l = \omega$, so by PARRIGHTSTOP and PARLEFT, $F = f|g \xrightarrow{e'} \langle f, \emptyset, \{e'_k\}, \emptyset\rangle \xrightarrow{e} F'' = \langle f', \emptyset, \{e'_k\}, \emptyset\rangle$.
- **PARLEFTSTOP:** we have $f \xrightarrow{e} f' \xrightarrow{e''=(e'_k,\omega,e'_c,e'_a)} \bot$ with $e||e''$, so by induction, there is h such that $f \xrightarrow{e''} h \xrightarrow{e} \bot$. As $e''_l = \omega$, $h = \bot$, which is not possible.

SEQN: $F = f >x> g \xrightarrow{e} f' >x> g = F'$. The second step can be generated by three different rules:

- **SEQSTOP:** we have $f \xrightarrow{e} f' \xrightarrow{e'} \bot$ with $e'_l = \omega$. As above, $f \xrightarrow{e'} \bot$, which is impossible.
- **SEQN:** we have $f \xrightarrow{e} f' \xrightarrow{e'} f''$ with $e||e'$, so by induction, there are $h, h' \equiv f''$ such that $f \xrightarrow{e'} h \xrightarrow{e} h'$. We can apply SEQN twice to obtain $F = f >x> g \xrightarrow{e'} h >x> g \xrightarrow{e} h' >x> g \equiv F''$.
- **SEQV:** we have $f \xrightarrow{e} f' \xrightarrow{e''=(e'_k,!v,e'_c,e'_a)} f''$ with $e||e''$ and $e'_l = h(v)$. By induction, there are $h, h' \equiv f''$ such that $f \xrightarrow{e''} h \xrightarrow{e} h'$. We can apply SEQV and then PARRIGHT and SEQN, so that $F = f >x> g \xrightarrow{e'} h >x> g|\langle [v/x]g, \{k\}, \emptyset, \emptyset\rangle \xrightarrow{e} h' >x> g|\langle [v/x]g, \{k\}, \emptyset, \emptyset\rangle \equiv F''$.

SEQV: $F = f >x> g \xrightarrow{e} f' >x> g|\langle [v/x]g, \{e_k\}, \emptyset, \emptyset\rangle = F'$. As $e \not\leq e'$, it is not possible to generate the second step from the right hand side of F', so there are four possibilities:

- **PARLEFTSTOP:** $e'_l = h(\omega)$, so the premise is $f' >x> g \xrightarrow{e''=(e'_k,\omega,e'_c,e'_a)} \bot$. As above, it is not compatible with the fact that $e||e'$.
- **PARLEFT and SEQSTOP:** we have $e'_l = \omega$, which is incompatible with the fact that $e||e'$.
- **PARLEFT and SEQN:** we have $f \xrightarrow{e''=(e_k,!v,e_c,e_a)} f' \xrightarrow{e'} f''$ with $e''|e'$. By induction, there are $h, h' \equiv f''$ such that $f \xrightarrow{e'} h \xrightarrow{e} h'$, so we can apply SEQN and then SEQV, so $F = f >x> g \xrightarrow{e'} h >x> g \xrightarrow{x} h' >x> g|\langle [v/x]g, \{k\}, \emptyset, \emptyset\rangle \equiv F''$.

PARLEFT and SEQV: we have $f \xrightarrow{e''=(e_k,!v,e_c,e_a)} f' \xrightarrow{e'''=(e'_k,!w,e'_c,e'_a)} f''$. By induction, there are $h, h' \equiv f''$ such that $f \xrightarrow{e'''} h \xrightarrow{e''} h'$. We can apply the same rules, so $F = f >x> g \xrightarrow{e'} h > x> g|\langle[w/x]g, \{e'_k\}, \emptyset, \emptyset\rangle \xrightarrow{e} h' >x> (g|\langle[v/x]g, \{e_k\}, \emptyset, \emptyset\rangle)|\langle[w/x]g, \{e'_k\}, \emptyset, \emptyset\rangle \equiv F''$.

PRUNELEFT: $F = f <x< g \xrightarrow{e} f' <x< g = F'$. There are four possibilities for the second step:

PRUNESTOP: we have $g \xrightarrow{e''=(e'_k,\omega,e'_c,e'_a)} \perp$, so we can apply PRUNESTOP on F, which gives $F = f <x< g \xrightarrow{e'} \langle[\langle \mathbf{stop}, \emptyset, c, \emptyset\rangle/x]f, \emptyset, \{k\}, \emptyset\rangle = G$. The premise of the first step is $f \xrightarrow{e} f'$. By lemma 3, $G \xrightarrow{e} F''$.

PRUNELEFT: we have $f \xrightarrow{e} f' \xrightarrow{e'} f''$ and $e||e'$, so by induction there are $h, h' \equiv f''$ such that $f \xrightarrow{e'} h \xrightarrow{e} h'$. We can apply PRUNELEFT twice, so that $F = f <x< g \xrightarrow{e'} h <x< g \xrightarrow{e} h' < x< g \equiv F''$.

PRUNEN: we have $f \xrightarrow{e} f'$ and $g \xrightarrow{e'} g'$. We can apply PRUNEN and then PRUNELEFT, so $F = f <x< g \xrightarrow{e'} f <x< \langle g', \emptyset, \emptyset, \{e'_k\}\rangle \xrightarrow{e} f' <x< \langle g', \{e'_k\}, \emptyset, \emptyset\rangle = F''$.

PRUNEV: we have $g \xrightarrow{e''=(e'_k,!v,e'_c,e'_a)} g'$, so we can apply PRUNEV on F, which gives $F = f < x< g \xrightarrow{e'} \langle[\langle v, \{e'_k\}, \emptyset, \emptyset\rangle/x]f, \emptyset, \{e'_k\}, \emptyset\rangle = G$. The premise of the first step is $f \xrightarrow{e} f'$. By lemma 3, $G \xrightarrow{e} F''$.

PRUNEN: $F = f <x< g \xrightarrow{e} f <x< \langle g', \emptyset, \emptyset, \{e_k\}\rangle = F'$. As $e||e'$, PRUNEV is impossible for the next step, that has to be:

PRUNESTOP: we have $g \xrightarrow{e} g' \xrightarrow{e''=(e'_k,\omega,e'_c,e'_a)} g''$. It is impossible as $e||e'$.

PRUNELEFT: we have $f \xrightarrow{e'} f'$. We can apply PRUNELEFT and then PRUNEN, so $F = f <x< g \xrightarrow{e'} f' <x< g \xrightarrow{e} f' <x< \langle g', \{e_k\}, \emptyset, \emptyset\rangle = F''$.

PRUNEN and CAUSALN: we have $g \xrightarrow{e} g' \xrightarrow{e'} g''$. By induction, there are $h, h' \equiv g''$ such that $g \xrightarrow{e'} h \xrightarrow{e} h'$. We can apply PRUNEN followed by PRUNEN and CAUSALN, so $F = f <x< g \xrightarrow{e'} f <x< \langle h, \{e'_k\}, \emptyset, \emptyset\rangle \xrightarrow{e} f' <x< \langle\langle h', \{e'_k\}, \emptyset, \emptyset\rangle, \{e_k\}, \emptyset, \emptyset\rangle \equiv F''$.

PRUNEV (the same arguments hold for PRUNESTOP): $F = f < x < g \xrightarrow{e} \langle[\langle v, \{e_k\}, \emptyset, \emptyset\rangle/x]f, \emptyset, \{e_k\}, \emptyset\rangle = F'$. As $e||e'$, the next step is given by either CAUSALN or causal CAUSALV, and the premise is $[\langle v, \{e_k\}, \emptyset, \emptyset\rangle/x]f \xrightarrow{e'} f'$, as $c_l = a_v = \emptyset$. By lemma 3, as $e||e'$, $f \xrightarrow{e'} f''$, with $[\langle v, \{e_k\}, \emptyset, \emptyset\rangle/x]f'' = f'$. We can apply PRUNELEFT and then PRUNEV, so $F = f <x< g \xrightarrow{e'} f' <x< g \xrightarrow{e} \langle[\langle v, \{e_k\}, \emptyset, \emptyset\rangle/x]f', \emptyset, \{e_k\}, \emptyset\rangle = F'$.

OTHERN: $F = f; g \xrightarrow{e} f'; g$. The next step can be generated by three rules :

OTHERN: we have $f \xrightarrow{e} f' \xrightarrow{e'} f''$, with $e||e'$. By induction, there are $h, h' \equiv f''$ such that $f \xrightarrow{e'} h \xrightarrow{e} h'$. We can apply OTHERN twice, so that $F = f; g \xrightarrow{e'} h; g \xrightarrow{e} h'; g \equiv F''$.

OTHERV: we have $f \xrightarrow{e} f' \xrightarrow{e''=(e'_k,!v,e'_c,e'_a)} F''$, with $e||e'$. By induction, there are $h, h' \equiv F''$ such that $f \xrightarrow{e'} h \xrightarrow{e} h'$. We can apply OTHERV on the first step, so that $f \xrightarrow{e'} h \xrightarrow{e} h' \equiv F''$.

OTHERSTOP: $f \xrightarrow{e} f' \xrightarrow{e'} \perp$. Once again, it is not possible as $e||e'$.

OTHERV: $F = f; g$. As $e||e'$, by lemma 2, $e'_l \neq \omega$. It leaves a lot of possibilities for the second rule, that we will reduce to two cases :

- $e'_l = !w$: we have $f \xrightarrow{e} f' \xrightarrow{e'} f''$. By induction, there are $h, h' \equiv F''$ such that $f \xrightarrow{e'} h \xrightarrow{e} h'$. We can apply OTHERV on the first step, and $f; g \xrightarrow{e'} h \xrightarrow{e} h' \equiv F''$.
- $e'_l = n$: we have $f \xrightarrow{e} f' \xrightarrow{e'} f''$. By induction, there are $h, h' \equiv F''$ such that $f \xrightarrow{e'} h \xrightarrow{e} h'$. We can apply OTHERN on the first step, and OTHERV on the second step, so $f; g \xrightarrow{e'} h; g \xrightarrow{e} h' \equiv F''$.

CAUSALN or CAUSALV: $F = \langle f, c_l, c_\omega\rangle \xrightarrow{e} \langle f', c_l, c_\omega\rangle = F'$. The next step cannot be CAUSALSTOP because $e||e'$, so it is generated by CAUSALN or CAUSALV. We have $f \xrightarrow{e} f' \xrightarrow{e'} f''$, so by induction there are $h, h' \equiv f''$ such that $f \xrightarrow{e'} h \xrightarrow{e} h'$. We can apply the same rulesin the reverse order, so $F = \langle f, c_l, c_\omega\rangle \xrightarrow{e'} \langle h, c_l, c_\omega\rangle \xrightarrow{e} \langle h', c_l, c_\omega\rangle \equiv F''$.

□

Definition 9.

$$
\begin{array}{rcll}
\pi_k(\langle f, c_l, c_\omega, a_v\rangle) & = & \pi_k(f) & \text{if } c_l \cup c_\omega \cup a_v = \{k\} \\
\pi_k(\langle f, c_l, c_\omega, a_v\rangle) & = & \langle \pi_k(f), c_l \setminus \{k\}, c_\omega \setminus \{k\}, a_v \setminus \{k\}\rangle & \text{otherwise} \\
\pi_k(\mathbf{def}\ y(\overline{x}) = f) & = & \mathbf{def}\ y(\overline{x}) = \pi_k(f) & \\
\pi_k(p) & = & p & \\
\pi_k(p(\overline{q})) & = & \pi_k(p)(\overline{\pi_k(q)}) & \\
\pi_k(?k') & = & ?k' & \\
\pi_k(f|g) & = & \pi_k(f)|\pi_k(g) & \\
\pi_k(f >x> g) & = & \pi_k(f) >x> \pi_k(g) & \\
\pi_k(f <x< g) & = & \pi_k(f) <x< \pi_k(g) & \\
\pi_k(f; g) & = & \pi_k(f); \pi_k(g) & \\
\pi_k(D\#f) & = & \pi_k(D)\#\pi_k(f) & \\
\pi_k(\bot) & = & \bot. &
\end{array}
$$

Lemma 5. *Let F, F' and F'' be instrumented orc programs and $e \rightsquigarrow e'$ be two events of the instrumented semantics such that $F \xrightarrow{e} F' \xrightarrow{e'} F''$. Then $F \xrightarrow{\pi(e')=(e'_k, e'_l, e'_c, e'_a \setminus \{e_k\})} \pi_{e_k}(F'')$.*

Proof. Let F, F' and F'' be instrumented orc programs and $e \rightsquigarrow e'$ be two events such that $F \xrightarrow{e} F' \xrightarrow{e'} F''$.

Let ${e'}^0, ..., {e'}^n$ be the sequence of events that appear in the derivation tree that produced e', such that ${e'}^0$ is produced by an axiom, for all $i < n$, ${e'}^i$ and ${e'}^{i+1}$ are the labels of the premise and the conclusion of a rule, and ${e'}^n = e'$.

Let $u_i = {e'}^i_a \setminus {e'}^i_c$ for all i. The sequence $(u_i)_{i<n}$ is growing for set inclusion, $u_0 = \emptyset$ and $e_k \in u_n$ as $e \rightsquigarrow e'$. There is a unique j such that for all $i < j$, $e_k \notin u_i$ and for all $i \geq j$, $e_k \in u_i$.

The only possibility to add an identifier in a and not in c is to use the rule CAUSALV, so it was instanciated to build ${e'}^j$, and there are f, f' and $c_l, c'\omega, a_v$ such that $e_k \in a_v$, $f \xrightarrow{{e'}^{j-1}} f'$ and $\langle f, c_l, c_\omega, a_v\rangle \xrightarrow{{e'}^j} \langle f', c_l, c_\omega, a_v\rangle$.

As $e_k \in a_v$, and f_0 is not instrumented, $\langle f, c_l, c_\omega, a_v\rangle$ was built at step e with rule PRUNEN. There is a subterm $H = \langle h, \emptyset, \emptyset, \{e_k\}\rangle$ in F' that is created during the first step.

We know that the rule PRUNEN is called in the derivation tree of the first step, and that it produces a label n. All the rules that have a label $n \neq \omega$ in premise just copy it in the conclusion, so $e_l = n$ and we will only considere the rules that have a label n in there premises.

Similarly, the rule PRUNEV is called in the derivation tree of the first step, and it produces a label $h(!v)$. For the same reason, apart from PRUNEV, we only considere the rules that have a label $h(!v)$ in there premises for the second step.

The cases are, for e. For many rules, more than one argument holds, but we just give one of them:

DEFDECLARE : it is not possible as $e \not\leq e'$.

PARLEFT (the same arguments hold for PARRIGHT, SEQN, PRUNELEFT, OTHERN and CAUSALN): $F = f|g \xrightarrow{e} f'|g = F'$ and $f \xrightarrow{e} f'$. H is contained in f', so the second step must be performed by PARLEFT (because PARLEFTSTOP has ω in its premise) with the premise $f' \xrightarrow{e'} f''$. By induction, $f \xrightarrow{\pi(e')} \pi(f'')$, so by PARLEFT, $F \xrightarrow{\pi(e')} \pi(F'')$.

PRUNEN: $F = f <x< g \xrightarrow{e} f <x< \langle g', \emptyset, \emptyset, \{e_k\} = F'$ and $g \xrightarrow{e} g'$. A subterm corresponding to H is build in the conclusion of the rule, but it does not implie that it CAUSALV was applied on it in the second step, as g' was also build in the first step. Two choices are possible for the second step:

- **PRUNEN:** gives $g' \xrightarrow{e'} g''$, so by induction, $g \xrightarrow{\pi(e')} g''$, and by rule PRUNEN, $F = f <x< g \xrightarrow{\pi(e')} f <x< \langle g', \emptyset, \emptyset, \{e'_k\}\rangle = \pi(F'')$
- **PRUNEN:** we have $g \xrightarrow{e} g' \xrightarrow{e''=(e'_k, !v, e'_c, e'_a \setminus \{e_k\})} g''$, so by induction $g \xrightarrow{\pi(e'')} pi(g'')$. We can apply PRUNEV, so $g \xrightarrow{\pi(e')=(e'_k, e'_l, e'_c, e'_a \setminus \{e_k\})} pi(g'')$.

□

Lemma 6. *If $F \xrightarrow{e} F'$ and $k \notin e_a$, then $\pi_k(F) \xrightarrow{e} \pi_k(F')$.*

Suppose $F \xrightarrow{e} F'$, $k \neq e_k$ and $k \notin e_a$, then $\pi_k(F) \xrightarrow{e} \pi_k(F')$.

Proof. Suppose that $F \xrightarrow{e} F'$, $k \neq e_k$ and $k \notin e_a$.

Let us considere the rule that was used to generate e:

PUBLISH (the same argument hold for STOP, NTRES, TRES and NEGRES): we have $F = v = \pi_k(F)$ and $F' = \langle \mathbf{stop}, \emptyset, \{e_k\}, \emptyset \rangle = \pi_k(F')$, so $\pi_k(F) \xrightarrow{e} \pi_k(F')$.

STOPCALL, INTCALL, EXTCALL, EXTSTOP: If $F \neq \pi_k(F)$, we would have $k \in e_a$. So $F = \pi_k(F)$, $F' = \pi_k(F')$ and $\pi_k(F) \xrightarrow{e} \pi_k(F')$.

□

Theorem 1. *Let $(E, \leq, \nearrow)$ be a concurrent execution and $(E_1, \mapsto_1)$ be a linearization of E. Then $(E_1, \mapsto_1)$ is feasible if and only if it is conform to E.*

Proof. Let $e^0...e^n$ be a sequential execution and let $(E, \leq, \nearrow)$ be the concurrent execution obtained from it. Let $(E_1, \mapsto_1)$ be a linearization of E.

Implication. Suppose $(E_1, \mapsto_1)$ is feasible. There is a sequential execution $\pi^a_{E_1}(e_0)...\pi^a_{E_1}(e_n)$ such that $E_1 = \{e_0, ..., e_n\}$ and $e_0 \mapsto_1 ... \mapsto_1 e_n$. Let $(E_\pi, \leq_\pi, \nearrow_\pi)$ be the concurrent execution associated with $\pi^a_{E_1}(e_0)...\pi^a_{E_1}(e_n)$.

Let $\pi^a_{E_1}(e) \mapsto_\pi \pi^a_{E_1}(e')$ denote $e \mapsto_1 e'$. As above, $(E_\pi, \mapsto_\pi)$ is conform to E_π. We have $e \in E_1$ if and oonly if $\pi^a_{E_1}(e) \in E_\pi$, $e \leq e'$ if and only if $\pi^a_{E_1}(e) \leq_\pi \pi^a_{E_1}(e')$ and $e \nearrow e'$ if and only if $\pi^a_{E_1}(e) \nearrow_\pi \pi^a_{E_1}(e')$. So $(E_1, \mapsto_1)$ is conform to E.

Converse. Suppose $(E_1, \mapsto_1)$ is conform to E.

Let W be the set of linearizations $(E_2, \mapsto_2)$ of E that are conform to E and feasible such that $E_1 \subset E_2$. For $(E_2, \mapsto_2) \in W$, we pose

$$L(E_2, \mapsto_2) = \sum_{e \in E_2 \setminus E_1} |\{e' \in E_2, e \mapsto_2 e'\}| + |\{(e, e') \in E_1^2 | e \mapsto_1 e' \wedge e' \mapsto_2 e\}|.$$

Let $e^i \mapsto_0 e^j$ denote the fact that $i \leq j$. The first part of the proof of property 1 actually claims that $(E, \mapsto_0)$ is conform to E. It is also feasible by construction, and it contains E_1, so W is not empty, and $\min_{w \in W} L(w) \geq 0$.

Let $(E_2, \mapsto_2) \in W$ be a linearization such that $L(E_2, \mapsto_2) = \min_{w \in W} L(w)$. Suppose (for contradiction) that $L(E_2, \mapsto_2) > 0$.

Case 1. Suppose $e = \max_{\mapsto_2} E_2 \notin E_1$. Let us considere $E_3 = E_2 \setminus \{e\}$ and $\mapsto_3 = \mapsto_2 \cap E_3^2$. As $(E_2, \mapsto_2)$ is conform to E, E_3 is left-closed for causality and $\mapsto_3$, that coincides with $\mapsto_2$ on E_3, is compatible with the weak-causality, so $(E_3, \mapsto_3)$ is conform to E. As $e' \notin E_1$, $E_1 \subset E_3$.

$(E_2, \mapsto_2)$ is feasible, so there is a sequential execution $f_0 \xrightarrow{\pi^a_{E_2}(e_2^1)} F_1...F_{n-2} \xrightarrow{\pi^a_{E_2}(e_2^{n-1})} F_{n-1} \xrightarrow{\pi^a_{E_2}(e)} F_n$ is the instrumented semantics. As $(E_2, \mapsto_2)$ is conform, e is also maximal for weak causality, and $\pi^a_{E_3}(e^i) = \pi^a_{E_2}(e^i)$ for all i, so $f_0 \xrightarrow{\pi^a_{E_3}(e_2^1)} F_1...F_{n-2} \xrightarrow{\pi^a_{E_3}(e_2^{n-1})} F_{n-1}$ is a sequential execution, and $(E_3, \mapsto_3)$ is feasible.

$(E_3, \mapsto_3) \in W$, but $L(E_3, \mapsto_3) = L(E_2, \mapsto_2) - 1 - |E_3 \setminus E_1| < L(E_2, \mapsto_2)$, which is absurd.

Case 2. Suppose there is $e \in E_2 \setminus E_1$ that has a successor e' according to $\mapsto_2$, with $e||e'$.

Let $E_3 = E_2$ and $\mapsto_3 = \mapsto_2 \setminus \{(e, e')\} \cup \{(e', e)\}$. $(E_3, \mapsto_3)$ is still conform to E and $E_1 \subset E_3$. As $(E_2, \mapsto_2)$ is feasible, there is a sequence $f_0 \xrightarrow{\pi^a_{E_2}(e_2^1)} ... \xrightarrow{\pi^a_{E_2}(e_2^k)} F \xrightarrow{\pi^a_{E_2}(e)} F' \xrightarrow{\pi^a_{E_2}(e')} F'' \xrightarrow{\pi^a_{E_2}(e_2^{k+3})} ... \xrightarrow{\pi^a_{E_2}(e_2^n)} F'''$ in the concurrent semantics. By lemma 4, there is G, G' such that $G' \equiv F''$ and $f_0 \xrightarrow{\pi^a_{E_2}(e_2^1)} ... \xrightarrow{\pi^a_{E_2}(e_2^k)} F \xrightarrow{\pi^a_{E_2}(e')} G \xrightarrow{\pi^a_{E_2}(e)} G' \xrightarrow{\pi^a_{E_2}(e_2^{k+3})} ... \xrightarrow{\pi^a_{E_2}(e_2^n)} F'''$ also defines a valid sequential execution, and $\pi^a_{E_2} = \pi^a_{E_3}$ so $(E_3, \mapsto_3)$ is feasible.

However, $L(E_3, \mapsto_3) = L(E_2, \mapsto_2) - 1 < L(E_2, \mapsto_2)$, which is absurd.

Case 3. Suppose $E_1 \neq E_2$ and cases 1 and 2 do not apply. Take $e \in E_2 \subset E_1$ that is maximal for $\leq$ and e' its successor for $\mapsto_2$. We have $e \nearrow e'$ as case 2 does not apply and $e \not\leq e'$ as e is maximal for causality. So $e \rightsquigarrow e'$.

Let us considere $E_3 = E_2 \setminus \{e\}$ and $\mapsto_3 = \mapsto_2 \cap E_3^2$. As $(E_2, \mapsto_2)$ is conform to E and e is maximal, E_3 is left-closed for causality and $\mapsto_3$, that coincides with $\mapsto_2$ on E_3, is compatible with the weak-causality, so $(E_3, \mapsto_3)$ is conform to E. Moreover, $E_1 \subset E3$.

As $(E_2, \mapsto_2)$ is feasible, there is a sequence $f_0 \xrightarrow{\pi^a_{E_2}(e_2^1)} ... \xrightarrow{\pi^a_{E_2}(e_2^k)} F \xrightarrow{\pi^a_{E_2}(e)} F' \xrightarrow{\pi^a_{E_2}(e')} F'' \xrightarrow{\pi^a_{E_2}(e_2^{k+3})} ... \xrightarrow{\pi^a_{E_2}(e_2^n)} F'''$ in the concurrent semantics. By lemma **??** (certes, mais ce lemme est faux), if $e'' = (e'_k, e'_l, e'_c, e'_a \setminus \{e_k\})$, $f_0 \xrightarrow{\pi^a_{E_2}(e_2^1)} ... \xrightarrow{\pi^a_{E_2}(e_2^k)} F \xrightarrow{\pi^a_{E_2}(e'')} F'' \xrightarrow{\pi^a_{E_2}(e_2^{k+3})} ... \xrightarrow{\pi^a_{E_2}(e_2^n)} F'''$ also defines a valid sequential execution. Moreover, for $e_2^i \notin \{e, e'\}$, $\pi^a_{E_3}(e_2^i) = \pi^a_{E_2}(e_2^i)$, and $\pi^a_{E_3}(e') = \pi^a_{E_2}(e'')$, so $(E_3, \mapsto_3)$ is feasible. and $(E_3, \mapsto_3) \in W$.

However, $L(E_3, \mapsto_3) = L(E_2, \mapsto_2) - 1 < L(E_2, \mapsto_2)$, which is absurd.

Case 4. Otherwise, $E_1 = E_2$ and $U = \{(e, e') \in E_2^2 | e \mapsto_2 e' \wedge e' \mapsto_1 e\}$, the set of events that are unordered, is not empty. For $(e, e') \in U$, let $between(e, e') = \{e'' \in E_2, e \mapsto_2 e'' \mapsto_2 e'\}$ and suppose (e, e') minimizes $|between(e, e')|$. If there is $e'' \in between(e, e')$, then either $e'' \mapsto_1 e'$ or $e \mapsto_1 e''$, so $|between(e, e')|$ is not minimal. Then $between(e, e') = \emptyset$. As $(E_1, \mapsto_1)$ and $(E_2, \mapsto_2)$ are both conform to E, $e||e'$.

Let $E_3 = E_2$ and $\mapsto_3 = \mapsto_2 \setminus \{(e, e')\} \cup \{(e', e)\}$. This case is similar to case 2, where we saw that we could exchange to consecutive concurrent events. We have $(E_3, \mapsto_3) \in W$ and $L(E_3, \mapsto_3) = L(E_2, \mapsto_2) - 1 < L(E_2, \mapsto_2)$, which is absurd.

Conclusion. All this cases are impossible, so $L(E_2, \mapsto_2) = 0$ If there is $e \in E_2 \setminus E_1$, then $e \mapsto_2 e$ and $\sum_{e \in E_2 \setminus E_1} |\{e' \in E_2, e \mapsto_2 e'\}| > 0$, which is impossible, so $E_1 = E_2$. Moreover, $\mapsto_1 = \mapsto_2$ as $|\{(e, e') \in E_1^2 | e \mapsto_1 e' \wedge e' \mapsto_2 e\}| = 0$. Finally, $(E_2, \mapsto_2) = (E_1, \mapsto_1)$, so $(E_1, \mapsto_1) \in W$ and $(E_1, \mapsto_1)$ is feasible.

Finally, $(E_1, \mapsto_1)$ is feasible if and only if it is conform to E. □