# Event structures for the reversible early internal $\pi$-calculus

Eva Graversen, Iain Phillips[0000–0001–5013–5876], and Nobuko Yoshida[0000–0002–3925–8557]

Imperial College London, UK

**Abstract.** The $\pi$-calculus is a widely used process calculus, which models communications between processes and allows the passing of communication links. Various operational semantics of the $\pi$-calculus have been proposed, which can be classified according to whether transitions are unlabelled (so-called reductions) or labelled. With labelled transitions, we can distinguish early and late semantics. The early version allows a process to receive names it already knows from the environment, while the late semantics and reduction semantics do not. All existing reversible versions of the $\pi$-calculus use reduction or late semantics, despite the early semantics of the (forward-only) $\pi$-calculus being more widely used than the late. We define $\pi$IH, the first reversible early $\pi$-calculus, and give it a denotational semantics in terms of reversible bundle event structures. The new calculus is a reversible form of the internal $\pi$-calculus, which is a subset of the $\pi$-calculus where every link sent by an output is private, yielding greater symmetry between inputs and outputs.

## 1 Introduction

The $\pi$-calculus [18] is a widely used process calculus, which models communications between processes using input and output actions, and allows the passing of communication links. Various operational semantics of the $\pi$-calculus have been proposed, which can be classified according to whether transitions are unlabelled or labelled. Unlabelled transitions (so-called reductions) represent completed interactions. As observed in [25] they give us the internal behaviour of complete systems, whereas to reason compositionally about the behaviour of a system in terms of its components we need labelled transitions. With labelled transitions, we can distinguish early and late semantics [19], with the difference being that early semantics allows a process to receive (free) names it already knows from the environment, while the late does not. This creates additional causation in the early case between those inputs and previous output actions making bound names free. All existing reversible versions of the $\pi$-calculus use reduction semantics [14, 26] or late semantics [7, 17]. However the early semantics of the (forward-only) $\pi$-calculus is more widely used than the late, partly because it has a sound correspondence with contextual congruences [13, 20].

We define $\pi$IH, the first reversible early $\pi$-calculus, and give it a denotational semantics in terms of reversible event structures. The new calculus is a reversible form of the internal $\pi$-calculus, or $\pi$I-calculus [24], which is a subset of the $\pi$-calculus where every link sent by an output is bound (private), yielding greater symmetry between inputs and outputs. It has been shown that the asynchronous $\pi$-calculus can be encoded in the asynchronous form of the $\pi$I-calculus [2].

The $\pi$-calculus has two forms of causation. *Structural* causation, as one would find in CCS, comes directly from the structure of the process, e.g. in $a(b).c(d)$ the action $a(b)$ must happen before $c(d)$. *Link* causation, on the other hand, comes from one action making a name available for others to use, e.g. in the process $a(x)|\overline{b}(c)$, the event $a(c)$ will be caused by $\overline{b}(c)$ making $c$ a free name. Note that link causation as in this example is present in the early form of the $\pi$I-calculus though not the late, since it is created by the process receiving one of its free names. Restricting ourselves to the $\pi$I-calculus, rather than the full $\pi$-calculus lets us focus on the link causation created by early semantics, since it removes the other forms of link causation present in the $\pi$-calculus.

We base $\pi$IH on the work of Hildebrandt *et al.* [12], which used extrusion histories and locations to define a stable non-interleaving early operational semantics for the $\pi$-calculus. We extend the extrusion histories so that they contain enough information to reverse the $\pi$I-calculus, storing not only extrusions but also communications. Allowing processes to evolve, while moving past actions to a history separate from the process, is called dynamic reversibility. By contrast, static reversibility, as in CCSK [21], lets processes keep their structure during the computation, and annotations are used to keep track of the current state and how actions may be reversed.

Event structures are a model of concurrency which describe causation, conflict and concurrency between events. They are ‘truly concurrent’ in that they do not reduce concurrency of events to the different possible interleavings. They have been used to model forward-only process calculi [3, 6, 27], including the $\pi$I-calculus [5]. Describing reversible processes as event structures is useful because it gives us a simple representation of the causal relationships between actions and gives us equivalences between processes which generate isomorphic event structures. True concurrency in semantics is particularly important in reversible process calculi, as the order actions can reverse in depends on their causal relations [22].

Event structure semantics of dynamically reversible process calculi have the added complexity of the histories and the actions in the process being separated, obscuring the structural causation. This was an issue for Cristescu *et al.* [8], who used rigid families [4], related to event structures, to describe the semantics of R$\pi$ [7]. Their semantics require a process to first reverse all actions to find the original process, map this process to a rigid family, and then apply each of the reversed memories in order to reach the current state of the process. Aubert and Cristescu [1] used a similar approach to describe the semantics of a subset of RCCS processes as configuration structures. We use a different tactic of first mapping to a statically reversible calculus, $\pi$IK, and then obtaining the event structure. This means that while we do have to reconstruct the original structure of the process, we avoid redoing the actions in the event structure.

Our $\pi$IK is inspired by CCSK and the statically reversible $\pi$-calculus of [17], which use communication keys to denote past actions. To keep track of link causation, keys are used in a number of different ways in [17]. In our case we can handle link causation by using keys purely to annotate the action which was performed using the key, and any names which were substituted during that action.

Although our two reversible variants of the $\pi$I-calculus have very different syntax and originate from different ideas, we show an operational correspondence between them in Theorem 4.6. We do this despite the extrusion histories containing more information than the keys, since they remember what bound names were before being substituted. The mapping from $\pi$IH to $\pi$IK bears some resemblance to the one presented from RCCS to CCSK in [16], though with some important differences. $\pi$IH uses centralised extrusion histories more similar to rho$\pi$ [15] while RCCS uses distributed memories. Additionally, unlike CCS, $\pi$I has substitution as part of its transitions and memories are handled differently by $\pi$IK and $\pi$IH, and our mapping has to take this into account.

We describe denotational structural event structure semantics of $\pi$IK, partly inspired by [5, 6], using reversible bundle event structures [10]. Reversible event structures [23] allow their events to reverse and include relations describing when events can reverse. Bundle event structures are more expressive than prime event structures, since they allow an event to have multiple possible conflicting causes. This allows us to model parallel composition without having a single action correspond to multiple events. While it would be possible to model $\pi$IK using reversible prime event structures, using bundle event structures not only gives us fewer events, it also lays the foundation for adding rollback to $\pi$IK and $\pi$IH, similarly to [10], which cannot be done using reversible prime event structures.

The structure of the paper is as follows: Section 2 describes $\pi$IH; Section 3 describes $\pi$IK; Section 4 describes the mapping from $\pi$IH to $\pi$IK; Section 5 recalls labelled reversible bundle event structures; and Section 6 gives event structure semantics of $\pi$IK. Proofs of the results presented in this paper can be found in the technical report [11].

## 2 $\pi$I-calculus reversible semantics with extrusion histories

Stable non-interleaving, early operational semantics of the $\pi$-calculus were defined by Hildebrandt *et al.* in [12], using locations and extrusion histories to keep track of link causation. We will in this section use a similar approach to define a reversible variant of the $\pi$I-calculus, $\pi$IH, using the locations and histories to keep track of not just causation, but also past actions. The $\pi$I-calculus is a restricted variant of the $\pi$-calculus wherein output on a channel $a$, $\overline{a}(b)$, binds the name being sent, $b$, corresponding to the $\pi$-calculus process $(\nu b)\overline{a}\langle b\rangle.P$. This creates greater symmetry with the input $a(x)$, where the variable $x$ is also bound. The syntax of $\pi$IH processes is:

$$P ::= \sum_{i\in I} \alpha_i.P_i \mid P_0|P_1 \mid (\nu x)P \quad \alpha ::= \overline{a}(b) \mid a(b)$$

The forward semantics of $\pi$IH can be seen in Table 1 and the reverse semantics can be seen in Table 2. We associate each transition with an action $\mu ::= \alpha \mid \tau$ and a location $u$ (Definition 2.1), describing where the action came from and what changes are made to the process as a result of the action. We store these location and action pairs in extrusion and communication histories associated with processes, so $(\overline{H}, \underline{H}, H) \vdash P$ means that if $(\mu, u)$ is an action and location pair in the output history $\overline{H}$ then $\mu$ is an output action, which $P$ previously performed at location $u$. Similarly $\underline{H}$ contains pairs of input actions and locations and $H$ contains triples of two communicating actions and the location associated with their communication. We use $\mathbf{H}$ as shorthand for $(\overline{H}, \underline{H}, H)$.

**Definition 2.1 (Location [12]).** *A location $u$ of an action $\mu$ is one of the following:*

1. $l[P][P']$ *if $\mu$ is an input or output, where $l \in \{0,1\}^*$ describes the path taken through parallel compositions to get to $\mu$'s origin, $P$ is the subprocess reached by following the path before $\mu$ has been performed, and $P'$ is the result of performing $\mu$ in $P$.*
2. $l\left\langle 0l_0[P_0][P_0'], 1l_1[P_1][P_1'] \right\rangle$ *if $\mu = \tau$, where $l0l_0[P_0][P_0']$ and $l1l_1[P_1][P_1']$ are the locations of the two actions communicating.*

*The path $l$ can be empty if the action did not go through any parallel compositions.*

We also use the operations on extrusion histories from Definition 2.2. These (1) add a branch to the path in every location, (2) isolate the extrusions whose locations begin with a specific branch, (3) isolate the extrusions whose locations begin with a specific branch and then remove the first branch from the locations, and (4) add a pair to the history it belongs in.

**Definition 2.2 (Operations on extrusion histories [12]).** *Given an extrusion history $(\overline{H}, \underline{H}, H)$, for $H^* \in \{\overline{H}, \underline{H}, H\}$ we have the following operations for $i \in \{0,1\}$:*

1. $iH^* = \{(\mu, iu) \mid (\mu, u) \in H^*\}$
2. $[i]H^* = \{(\mu, iu) \mid (\mu, iu) \in H^*\}$
3. $[\check{i}]H^* = \{(\mu, u) \mid (\mu, iu) \in H^*\}$
4. $$\mathbf{H} + (\mu, u) = \begin{cases} (\overline{H} \cup \{L\}, \underline{H}, H) & \textit{if } (\mu, u) = (\overline{a}(n), u) \\ (\overline{H}, \underline{H} \cup \{L\}, H) & \textit{if } (\mu, u) = (a(x), u) \\ (\overline{H}, \underline{H}, H \cup \{L\}) & \textit{if } (\mu, u) = (a(x), \overline{a}(n), l\langle u_0, u_1 \rangle) \end{cases}$$

The forwards semantics of $\pi$IH have six rules. In [OUT] the action is an output, the location is the process before and after doing the output, and they are added to the output history. The equivalent reverse rule, $[\mathrm{OUT}^{-1}]$, similarly removes the pair from the history and transforms the process from the second part of the location back to the first. The input rule [IN] works similarly, but performs a substitution on the received name and adds the pair to the input history instead. In $[\mathrm{PAR}_i]$ we isolate the parts of the histories whose locations start with $i$ and use those to perform an action in $P_i$, getting $\mathbf{H}_i' \vdash P_i'$. It then replaces the part of the histories parts of the histories whose locations start with $i$ with $\mathbf{H}_i'$ when propagating the action through the parallel. A communication in $[\mathrm{COM}_i]$ adds memory of the communication to the history. The rules [SCOPE] and [STR] are standard and self-explanatory.

The reverse rules use the extrusion histories to find a location $l[P][P']$ such that the current state of the subprocess at $l$ is $P'$, and change it to $P$.

In these semantics structural congruence, consisting only of $\alpha$-conversion together with $!P \equiv\ !P|P$ and $(\nu\, a)(\nu b)P \equiv (\nu\, b)(\nu\, a)P$, is primarily used to create and remove extra copies of a replicated process when reversing the action that happened before the replication. Since we use locations in our extrusion histories, we try to avoid

$$\frac{u = [\sum_{i\in I} \alpha_i.P_i][P_j] \quad \alpha_j = \overline{a}(n) \quad j \in I}{\mathbf{H} \vdash \sum_{i\in I} \alpha_i.P_i \xrightarrow[u]{\alpha_j} (\overline{H} \cup \{(\overline{a}(n), u)\}, \underline{H}, H) \vdash P_j} \;[\mathrm{OUT}]$$

$$\frac{u = [\sum_{i\in I} \alpha_i.P_i][P_j] \quad P'_j = P_j[x := n] \quad \alpha_j = a(x) \quad j \in I}{\mathbf{H} \vdash \sum_{i\in I} \alpha_i.P_i \xrightarrow[u]{a(n)} (\overline{H}, \underline{H} \cup \{(a(n), u)\}, H) \vdash P'_j} \;[\mathrm{IN}]$$

$$\frac{([\check{i}]\overline{H}, [\check{i}]\underline{H}, [\check{i}]H) \vdash P_i \xrightarrow[u]{\mu} \mathbf{H}'_i \vdash P'_i \quad P'_{1-i} = P_{1-i} \quad \text{if } \mu = \overline{a}(n) \text{ then } n \notin \mathsf{fn}(P_{1-i})}{\mathbf{H} \vdash P_0|P_1 \xrightarrow[iu]{\mu} ((\overline{H} \setminus [i]\overline{H}) \cup i\overline{H'_i}, (\underline{H} \setminus [i]\underline{H}) \cup i\underline{H'_i}, (H \setminus [i]H) \cup iH'_i) \vdash P'_0|P'_1} \;[\mathrm{PAR}_i]$$

$$\frac{\begin{array}{c}([\check{i}]\overline{H}, [\check{i}]\underline{H}, [\check{i}]H) \vdash P_i \xrightarrow[v_i]{\alpha_i} \mathbf{H}'_i \vdash P'_i \quad \alpha_i = \overline{a}(n) \quad \alpha_j = a(n) \\ ([\check{j}]\overline{H}, [\check{j}]\underline{H}, [\check{j}]H) \vdash P_j \xrightarrow[v_j]{\alpha_i} \mathbf{H}'_j \vdash P'_j \quad j = 1 - i \quad n \notin \mathsf{fn}(P_j)\end{array}}{\mathbf{H} \vdash P_0|P_1 \xrightarrow[(0v_0, 1v_1)]{\tau} (\overline{H}, \underline{H}, H \cup \{((\alpha_0, \alpha_1, \langle 0v_0, 1v_1\rangle))\}) \vdash (\nu n)(P'_0|P'_1)} \;[\mathrm{COM}_i]$$

$$\frac{\mathbf{H} \vdash P \xrightarrow[u]{\mu} \mathbf{H}' \vdash P' \quad x \notin n(\mu)}{\mathbf{H} \vdash (\nu x)P \xrightarrow[u]{\mu} \mathbf{H}' \vdash (\nu x)P'} \;[\mathrm{SCOPE}] \qquad \frac{P \equiv P' \quad \mathbf{H} \vdash P' \xrightarrow[u]{\mu} \mathbf{H}' \vdash Q' \quad Q' \equiv Q}{\mathbf{H} \vdash P \xrightarrow[u]{\mu} \mathbf{H}' \vdash Q} \;[\mathrm{STR}]$$

**Table 1.** Semantics of $\pi$IH (forwards rules)

using structural congruence any more than necessary. However, not using it for parallel composition would mean that we would need some other way of preventing traces such as $\mathbf{H} \vdash !P \xrightarrow[u]{\mu} \overset{\mu}{\underset{u}{\rightsquigarrow}} \mathbf{H} \vdash !P|P$, which allows a process to reach a state it could not reach via a parabolic trace. Using structural congruence for replication does not cause any problems for the locations, as we can tell past actions originating in each copy of $P$ apart by the path in their location, with actions from the $i$th copy having a path of $i$ 0s followed by a 1.

*Example 2.3.* Consider the process $(a(x).\overline{x}(d)|\overline{a}(c))|b(y)$. If we start with empty histories, each transition adds actions and locations:

$$(\emptyset, \emptyset, \emptyset) \vdash (a(x).\overline{x}(d)|\overline{a}(c))|b(y) \xrightarrow[0\langle 0[a(x).\overline{x}(d)][\overline{c}(d)], 1[\overline{a}(c)][0]\rangle]{\tau}$$

$$(\emptyset, \emptyset, \{(a(c), \overline{a}(c), 0\left\langle 0[a(x).\overline{x}(d)][\overline{c}(d)], 1[\overline{a}(c)][0]\right\rangle\}) \vdash (\nu c)(\overline{c}(d)|0)|b(y) \xrightarrow[00[\overline{c}(d)][0]]{\overline{c}(d)}$$

$$(\{(\overline{c}(d), 00[\overline{c}(d)][0])\}, \emptyset, \{(a(c), \overline{a}(c), 0\left\langle 0[a(x).\overline{x}(d)][\overline{c}(d)], 1[\overline{a}(c)][0]\right\rangle\}) \vdash (\nu c)(0|0)|b(y) \xrightarrow[1[b(y)][0]]{b(d)}$$

$$(\{(\overline{c}(b), 00[\overline{c}(b)][0])\}, \{(b(d), 1[b(y)][0])\}, \{(a(c), \overline{a}(c), 0\left\langle 0[a(x).\overline{x}(d)][\overline{c}(d)], 1[\overline{a}(c)][0]\right\rangle\}) \vdash (0|0)|0$$

We show that our forwards and reverse transitions correspond.

**Proposition 2.4 (Loop).**

1. *Given a $\pi$IH process $P$ and an extrusion history $\mathbf{H}$, if $\mathbf{H} \vdash P \xrightarrow[u]{\alpha} \mathbf{H}' \vdash Q$, then $\mathbf{H}' \vdash Q \overset{\alpha}{\underset{u}{\rightsquigarrow}} \mathbf{H} \vdash P$.*

$$\frac{u = [\sum_{i\in I} \alpha_i.P_i][P_j] \quad \alpha_j = \overline{a}(n) \quad j \in I \quad (\overline{a}(n), u) \in \overline{H}}{\mathbf{H} \vdash P_j \overset{\alpha_j}{\underset{u}{\rightsquigarrow}} (\overline{H} \setminus \left\{(\overline{a}(n),u)\right\}, \underline{H}, H) \vdash \sum_{i\in I}\alpha_i.P_i} \text{ [OUT}^{-1}]$$

$$\frac{u = [\sum_{i\in I} \alpha_i.P_i][P_j] \quad P'_j = P_j[x := n] \quad \alpha_j = a(x) \quad j \in I \quad (a(n), u) \in \underline{H}}{\mathbf{H} \vdash P'_j \overset{a(n)}{\underset{u}{\rightsquigarrow}} (\overline{H}, \underline{H} \setminus \{(a(n),u)\}, H) \vdash \sum_{i\in I}\alpha_i.P_i} \text{ [IN}^{-1}]$$

$$\frac{([\check{i}]\overline{H}, [\check{i}]\underline{H}, [\check{i}]H) \vdash P_i \overset{\alpha}{\underset{u}{\rightsquigarrow}} \mathbf{H}'_i \vdash P'_i \quad P'_{1-i} = P_{1-i} \text{ if } \alpha = \overline{a}(n) \text{ then } n \notin \mathsf{fn}(P_{1-i})}{\mathbf{H} \vdash P_0|P_1 \overset{\alpha}{\underset{iu}{\rightsquigarrow}} ((\overline{H} \setminus [i]\overline{H}) \cup i\overline{H'_i}, (\underline{H} \setminus [i]\underline{H}) \cup i\underline{H'_i}, (H \setminus [i]H) \cup iH'_i) \vdash P'_0|P'_1} \text{ [PAR}_i^{-1}]$$

$$\frac{\begin{array}{c}([\check{i}]\overline{H} \cup \{(\overline{a}(n), v_i)\}, [\check{i}]\underline{H}, [\check{i}]H) \vdash P_i \overset{\overline{a}(n)}{\underset{v_i}{\rightsquigarrow}} \mathbf{H}'_i \vdash P'_i \quad \alpha_i = \overline{a}(n) \quad \alpha_j = a(n) \\ ([\check{j}]\overline{H}, [\check{j}]\underline{H} \cup \{(a(n), v_j)\}, [\check{j}]H) \vdash P_j \overset{a(n)}{\underset{v_j}{\rightsquigarrow}} \mathbf{H}'_j \vdash P'_j \quad j = 1 - i \quad n \notin \mathsf{fn}(P_j)\end{array}}{\mathbf{H} \vdash (\nu n)(P_0|P_1) \underset{(0v_0, 1v_1)}{\overset{\tau}{\rightsquigarrow}} (\overline{H}, \underline{H}, H \setminus \{((\alpha_0, \alpha_1, \langle 0v_0, 1v_1\rangle))\} \vdash P'_0|P'_1} \text{ [COM}_i^{-1}]$$

$$\frac{\mathbf{H} \vdash P \overset{\mu}{\underset{u}{\rightsquigarrow}} \mathbf{H}' \vdash P' \quad x \notin n(\alpha)}{\mathbf{H} \vdash (\nu x)P \overset{\mu}{\underset{u}{\rightsquigarrow}} \mathbf{H}' \vdash (\nu x)P'} \text{ [SCOPE}^{-1}] \qquad \frac{P \equiv P' \quad \mathbf{H} \vdash P' \overset{\alpha}{\underset{u}{\rightsquigarrow}} \mathbf{H}' \vdash Q' \quad Q' \equiv Q}{\mathbf{H} \vdash P \overset{\alpha}{\underset{u}{\rightsquigarrow}} \mathbf{H}' \vdash Q} \text{ [STR}^{-1}]$$

**Table 2.** Semantics of reversible $\pi$IH (reverse rules)

2. *Given a forwards-reachable $\pi$IH process $P$ and an extrusion history* $\mathbf{H}$*, if* $\mathbf{H} \vdash P \overset{\alpha}{\underset{u}{\rightsquigarrow}} \mathbf{H}' \vdash Q$*, then* $\mathbf{H}' \vdash Q \xrightarrow[u]{\alpha} \mathbf{H} \vdash P$.

## 3 $\pi$I-calculus reversible semantics with annotations

In order to define event structure semantics of $\pi$IH, we first map from $\pi$IH to a statically reversible variant of $\pi$I-calculus, called $\pi$IK. $\pi$IK is based on previous statically reversible calculi $\pi$K [17] and CCSK [21]. Both of these use *communication keys* to denote past actions and which other actions they have interacted with, so $a(x)|\overline{a}(b) \xrightarrow{\tau[n]} a(b)[n]|\overline{a}(b)[n]$ means a communication with the key $n$ has taken place between the two actions. We apply this idea to define early semantics of $\pi$IK, which has the following syntax:

$$P ::= \alpha.P \mid \alpha[n].P \mid P_0 + P_1 \mid P_0|P_1 \mid (\nu x)P \quad \alpha ::= \overline{a}(b) \mid a(b)$$

The primary difference between applying communication keys to CCS and the $\pi$I-calculus is the need to deal with substitution. We need to keep track of not only which actions have communicated with each other, but also which names were substituted when. We do this by giving the substituted names a key, $a_{[n]}$, but otherwise treating them the same as those without the key, except when undoing the input associated with $n$.

$$
\frac{\mathsf{std}(P) \quad P' = P[x := b_{[n]}]}{a(x).P \xrightarrow{a(b)[n]} a(b)[n].P'}
\qquad
\frac{\mathsf{std}(P)}{\overline{a}(b).P \xrightarrow{\overline{a}(b)[n]} \overline{a}(b)[n].P}
$$

$$
\frac{P \xrightarrow{\mu[m]} P' \quad m \neq n \quad \text{if } \mu = \overline{a}(x) \text{ then } x \notin n(\alpha)}{\alpha[n].P \xrightarrow{\mu[m]} \alpha[n].P'}
\qquad
\frac{P_0 \xrightarrow{\mu[n]} P_0' \quad \mathsf{std}(P_1)}{P_0 + P_1 \xrightarrow{\mu[n]} P_0' + P_1}
$$

$$
\frac{P_0 \xrightarrow{\mu[n]} P_0' \quad \mathsf{fsh}[n](P_1) \quad \text{if } \mu = \overline{a}(b) \text{ then } b \notin \mathsf{fn}(P_1)}{P_0 | P_1 \xrightarrow{\mu[n]} P_0' | P_1}
\qquad
\frac{P_0 \xrightarrow{a(b)[n]} P_0' \quad P_1 \xrightarrow{\overline{a}(b)[n]} P_1'}{P_0 | P_1 \xrightarrow{\tau[n]} (\nu b)(P_0' | P_1')}
$$

$$
\frac{P \xrightarrow{\mu[m]} P' \quad a \notin n(\mu)}{(\nu a)P \xrightarrow{\mu[m]} (\nu a)P'}
\qquad
\frac{P \equiv Q \xrightarrow{\mu[n]} Q' \equiv P'}{P \xrightarrow{\mu[n]} P'}
$$

**Table 3.** $\pi$IK forward semantics

$$
\frac{\mathsf{std}(P) \quad x \notin n(P) \quad P' = P[b_{[m]} := x]}{a(b)[m].P \overset{a(b)[m]}{\rightsquigarrow} a(x).P'}
\qquad
\frac{\mathsf{std}(P)}{\overline{a}(b)[n].P \overset{\overline{a}(b)[n]}{\rightsquigarrow} \overline{a}(b).P}
$$

$$
\frac{P \overset{\mu[m]}{\rightsquigarrow} P' \quad m \neq n}{\alpha[n].P \overset{\mu[m]}{\rightsquigarrow} \alpha[n].P'}
\qquad
\frac{P_0 \overset{\mu[n]}{\rightsquigarrow} P_0' \quad \mathsf{std}(P_1)}{P_0 + P_1 \overset{\mu[n]}{\rightsquigarrow} P_0' + P_1}
$$

$$
\frac{P_0 \overset{\mu[n]}{\rightsquigarrow} P_0' \quad \mathsf{fsh}[n](P_1) \quad \text{if } \mu = \overline{a}(b) \text{ then } b \notin \mathsf{fn}(P_1)}{P_0 | P_1 \overset{\mu[n]}{\rightsquigarrow} P_0' | P_1}
\qquad
\frac{P_0 \overset{a(b)[n]}{\rightsquigarrow} P_0' \quad P_1 \overset{\overline{a}(b)[n]}{\rightsquigarrow} P_1'}{(\nu b)(P_0 | P_1) \overset{\tau[n]}{\rightsquigarrow} P_0' | P_1'}
$$

$$
\frac{P \overset{\mu[m]}{\rightsquigarrow} P' \quad a \notin n(\mu)}{(\nu a)P \overset{\mu[m]}{\rightsquigarrow} (\nu a)P'}
\qquad
\frac{P \equiv Q \overset{\mu[n]}{\rightsquigarrow} Q' \equiv P'}{P \overset{\mu[n]}{\rightsquigarrow} P'}
$$

**Table 4.** $\pi$IK reverse semantics

Table 3 shows the forward semantics of $\pi$IK. The reverse semantics can be seen in Table 4. We use $\alpha$ to range over input and output actions and $\mu$ over input, output, and $\tau$. We use $\mathsf{std}(P)$ denote that $P$ is a *standard process*, meaning it does not contain any past actions (actions annotated with a key), and $\mathsf{fsh}[n](P)$ to denote that a key $n$ is fresh for $P$. Names in past actions are always free. Our semantics very much resemble those of CCSK, with the exceptions of substitution and ensuring that any name being output does not appear elsewhere in the process. The semantics use structural congruence as defined in Table 5.

We again show a correspondence between forward and reverse transitions.

**Proposition 3.1 (Loop).**

1. *Given a process $P$, if $P \xrightarrow{\mu[n]} Q$ then $Q \overset{\mu[n]}{\rightsquigarrow} P$.*
2. *Given a forwards reachable process $P$, if $P \overset{\mu[n]}{\rightsquigarrow} Q$ then $Q \xrightarrow{\mu[n]} P$.*

| | | |
|---|---|---|
| $P\|0 \equiv P$ | $P_0\|P_1 \equiv P_1\|P_0$ | $P_0\|(P_1\|P_2) \equiv (P_0\|P_1)\|P_2$ |
| $P + 0 \equiv P$ | $P_0 + P_1 \equiv P_1 + P_0$ | $P_0 + (P_1 + P_2) \equiv (P_0 + P_1) + P_2$ |
| $!P \equiv !P\|P$ | $(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$ | $(\nu a)(P_0\|P_1) \equiv ((\nu a)P_0\|P_1)$ if $a \notin n(P_1)$ |

**Table 5.** Structural congruence

## 4 Mapping from $\pi$IH to $\pi$IK

We will now define a mapping from $\pi$IH to $\pi$IK and show that we have an operational correspondence in Theorem 4.6. The extrusion histories store more information than the keys, as they keep track of which names were substituted, as illustrated by Example 4.1. This means we lose some information in our mapping, but not information we need.

*Example 4.1.* Consider the processes $(\emptyset, \{(a(b), [a(x)][0])\}, \emptyset) \vdash 0$ and $a(b)[n]$. These are the result of $a(x)$ receiving $b$ in the two different semantics. We can see that the extrusion history remembers that the input name was $x$ before $b$ was received, but the keys do not remember, and when reversing the action could use any name as the input name. This does not make a great deal of difference, as after reversing $a(b)$, the process with the extrusion history can also $\alpha$-convert $x$ to any name.

Since we intend to define a mapping from processes with extrusion histories to processes with keys, we first describe how to add keys to substituted names in a process in Definition 4.2. We have a function, $S$, which takes a process, $P_1$, in which we wish to add the key $[n]$ to all those names which were $x$ in a previous state of the process, $P_2$, before being substituted for some other name in an input action with the key $[n]$.

**Definition 4.2 (Substituting in $\pi$IK-process to correspond with processes with extrusion histories).** *Given a $\pi$IK process $P_1$, a $\pi$I-calculus process without keys, $P_2$, a key $n$, and a name $x$, we can add the key $n$ to any names which $x$ has been substituted with, by applying $S(P_1, P_2, [n], x)$, defined as:*

1. $S(0, 0, [n], x) = 0$
2. $S\left(\sum_{i\in I} P_{i1}, \sum_{i\in I} P_{i2}, [n], x\right) = \sum_{i\in I} S\left(P_{i1}, P_{i2}, [n], x\right)$
3. $S\left(P_1|Q_1, P_2|Q_2, [n], x\right) = S\left(P_1, P_2, [n], x\right)|S\left(Q_1, Q_2, [n], x\right)$
4. $S\left((\nu a)P_1, (\nu b)P_2, [n], x\right) = P_1'$ *where:*
   *if* $x = b$ *then* $P_1' = P_1$ *and otherwise* $P_1' = (\nu a)S\left(P_1, P_2, [n], x\right)$.
5. $S\left(\alpha_1.P_1, \alpha_2.P_2, [n], x\right) = \alpha_1'.P_1'$ *where:*
   *if* $\alpha_2 \in \{x(c), \overline{x}(c)\}$ *then* $\alpha_1' = \alpha_{1_{[n]}}$ *and otherwise* $\alpha_1' = \alpha_1$*;*
   *if* $\alpha_2 \in \{c(x), \overline{c}(x)\}$ *then* $P_1' = P_1$ *and otherwise* $P_1' = S\left(P_1, P_2, [n], x\right)$.
6. $S\left(\alpha_1[m].P_1, \alpha_2.P_2, [n], x\right) = \alpha_1'[m].P_1'$ *where:*
   *if* $\alpha_2 \in \{x(c), \overline{x}(c)\}$ *then* $\alpha_1' = \alpha_{1_{[n]}}$ *and otherwise* $\alpha_1' = \alpha_1$*;*
   *if* $\alpha_2 \in \{c(x), \overline{c}(x)\}$ *then* $P_1' = P_1$ *and otherwise* $P_1' = S\left(P_1, P_2, [n], x\right)$.

7. $S\left(!P_1, !P_2, [n], x\right) = !S\left(P_1, P_2, [n], x\right)$
8. $S\left(P_1|P_1', !P_2, [n], x\right) = S\left(P_1, !P_2, [n], x\right) | S\left(P_1', P_2, [n], x\right)$
9. $S\left(!P_1, P_2|P_2', [n], x\right) = S\left(!P_1, P_2, [n], x\right) | S\left(P_1, P_2', [n], x\right)$

*where* $a(b)_{[n]} = a_{[n]}(b)$ *and* $\overline{a}(b)_{[n]} = \overline{a_{[n]}}(b)$

Being able to annotate our names with keys, we can define a mapping, $E$, from extrusion histories to keys in Definition 4.4. $E$ iterates over the extrusions, having one process which builds $\pi$IK-process, and another that keeps track of which state of the original $\pi$IH process has been reached. When turning an extrusion into a keyed action, we use the locations as key and also give each extrusion an extra copy of its location to use for determining where the action came from. This way we can use one copy to iteratively go through the process, removing splits from the path as we go through them, while still having another intact copy of the location to use as the final key. In $E(\mathbf{H} \vdash P, P')$, $\mathbf{H}$ is a history of extrusions which need to be turned into keyed actions, $P$ is the process these keyed actions should be added to, and $P'$ is the state the process would have reached, had the added extrusions been reversed instead of turned into keyed actions.

If $E$ encounters a parallel composition in $P$ (case 2), it splits its extrusion histories in three. One part, $\mathbf{H}_{\mathsf{shared}}$ contains the locations which have an empty path, and therefore belong to actions from before the processes split. Another part contains the locations beginning with 0, and goes to the first part of the process. And finally the third part contains the locations beginning with 1, and goes to the second part of the process.

$E$ can add an action – and the choices not picked when that action was performed – to $P$ (cases 3,4) when the associated location has an empty path and has $P'$ as its result process. When turning an input memory from the history into a past input action in the process (case 4), we use $S$ (Definition 4.2) to add keys to the substituted names. When $E$ encounters a restriction (case 5), it moves a memory that can be used inside the restriction inside. It does this iteratively until there are no such memories left in the extrusion histories. We apply $E$ to a process in Example 4.5.

**Definition 4.3.** *The function* $\mathsf{lcopy}$ *gives each member of an extrusion history an extra copy of its location:*

$\mathsf{lcopy}(H^*) = \{(\mu, u, u) \mid (\mu, u) \in H^*\}$

$\mathsf{lcopy}(\overline{H}, \underline{H}, H) = (\mathsf{lcopy}(\overline{H}), \mathsf{lcopy}(\underline{H}), \mathsf{lcopy}(H))$

**Definition 4.4.** *Given a $\pi$IH process,* $\mathbf{H} \vdash P$*, we can create an equivalent $\pi$IK process,* $E(\mathsf{lcopy}(\mathbf{H}) \vdash P, P) = P'$ *defined as*

1. $E((\emptyset, \emptyset, \emptyset) \vdash P, P') = P$
2. $E(\mathbf{H} \vdash P_0|P_1, P_0'|P_1') = E(\mathbf{H}_{\mathsf{shared}} \vdash P_0''|P_1'', P_0'''|P_1''')$ *where:*
   $\mathbf{H}_{\mathsf{shared}} = (\{(\alpha, u, u') \mid (\alpha, u, u') \in \overline{H} \text{ and } u \neq iu''\}, \{(\alpha, u, u') \mid (\alpha, u, u') \in \underline{H} \text{ and } u \neq iu''\}, \emptyset)$
   $P_0'' = E((\overline{H_0}, \underline{H_0}, H_0) \vdash P_0, P_0')$ *where:*
   $\overline{H_0} = \{(\overline{a}(b), u_0, u_0') \mid (\overline{a}(b), 0u_0, u_0') \in \overline{H} \text{ or } (\overline{a}(b), \alpha_1, \langle 0u_0, 1u_1\rangle, u_0') \in H\}$

$\underline{H_0} = \{(a(b), u_0, u'_0) \mid (a(b), 0u_0, u'_0) \in \underline{H} \text{ or } (a(b), \alpha_1, \langle 0u_0, 1u_1\rangle, u'_0) \in H\}$
$\overline{H_0} = \{(\alpha, \alpha', u, u') \mid (\alpha, \alpha', 0u, u') \in H\}$
$P''_1 = E((\overline{H_1}, \underline{H_1}, H_1) \vdash P_1, P'_1))$ *where:*
$\overline{H_1} = \{(\overline{a}(b), u_1, u'_1) \mid (\overline{a}(b), 1u_1, u'_1) \in \overline{H} \text{ or } (\alpha_0, \overline{a}(b), \langle 0u_0, 1u_1\rangle, u'_1) \in H\}$
$\underline{H_1} = \{(a(b), u_1, u'_1) \mid (a(b), 1u_1, u'_1) \in \underline{H} \text{ or } (\alpha_0, a(b), \langle 0u_0, 1u_1\rangle, u'_1) \in H\}$
$\overline{H_1} = \{(\alpha, \alpha', u, u') \mid (\alpha, \alpha', 1u, u') \in H\}$
$\mathbf{H_i} \vdash P'_i \overset{\alpha_{i,0}}{\underset{u_{i,0}}{\rightsquigarrow}} \dots \overset{\alpha_{i,n}}{\underset{u_{i,n}}{\rightsquigarrow}} (\emptyset, \emptyset, \emptyset) \vdash P'''_i$ *for* $i \in \{0, 1\}$

3. $E((\overline{H} \cup \{(\overline{a}(b), [Q][P'], u)\}, \underline{H}, H) \vdash P, P') = E(\mathbf{H} \vdash \overline{a}(b)\,[u]\,.P + \sum_{i \in I \setminus \{j\}} \alpha_i.P_i, Q)$
   *if* $Q = \sum_{i \in I} \alpha_i.P_i$, $\overline{a}(b) = \alpha_j$, *and* $P' = P_j$
4. $E((\overline{H}, \underline{H} \cup \{(a(b), [Q][P'], u)\}, H) \vdash P, P') =$
   $E(\mathbf{H} \vdash a(b)\,[u]\,.S(P, P_j, [u], x) + \sum_{i \in I \setminus \{j\}} \alpha_i.P_i, Q)$
   *if* $Q = \sum_{i \in I} \alpha_i.P_i$, $a(x) = \alpha_j$, *and* $P' = P_j[x := b]$
5. $E(\mathbf{H} \vdash (\nu x)P, (\nu x)P') = E(\mathbf{H} - (\alpha, u, u') \vdash P'', (\nu x)Q')$
   *where* $P'' = (\nu x)E((\emptyset, \emptyset, \emptyset) + (\alpha, u, u') \vdash P, P')$
   *if* $(\alpha, u, u') \in \overline{H} \cup \underline{H}$ *and* $(\emptyset, \emptyset, \emptyset) + (\alpha, u, u) \vdash P \overset{\alpha}{\underset{u}{\rightsquigarrow}} (\emptyset, \emptyset, \emptyset) \vdash Q'$
6. $E(\mathbf{H} \vdash !P, !P') = E(\mathbf{H} \vdash !P|P, !P'|P')$ *if there exists* $(\alpha, u, u') \in \overline{H} \cup \underline{H} \cup H$ *such that* $u \neq [Q][Q']$.

*Example 4.5.* We will now apply $E$ to the process

$$(\{(\overline{b}(c), u_2)\}, \emptyset, \{(b(a), \overline{b}(a), \langle 0u_0, 1u_1\rangle)\}) \vdash a(x) \mid 0$$

with locations $u_0 = [b(y).y(x)][a(x)]$, $u_1 = [\overline{b}(a)][0]$, and $u_2 = [\overline{b}(c).(b(y).y(x) \mid \overline{b}(a)][b(y).y(x) \mid \overline{b}(a)]$. We perform

$$E(\mathsf{lcopy}((\{(\overline{b}(c), u_2)\}, \emptyset, \{(b(a), \overline{b}(a), \langle 0u_0, 1u_1\rangle)\})) \vdash a(x) \mid 0, a(x) \mid 0)$$

Since we are at a parallel, we use Case 2 of Definition 4.4 to split the extrusion histories into three to get $E((\{(\overline{b}(c), u_2, u_2)\}, \emptyset, \emptyset) \vdash P_0 \mid P_1, b(y).y(x) \mid \overline{b}(a))$ where $P_0 = E((\emptyset, \{(b(a), u_0, \langle 0u_0, 1u_1\rangle)\}, \emptyset) \vdash a(x), a(x))$ and $P_1 = E((\{(\overline{b}(a), u_1, \langle 0u_0, 1u_1\rangle)\}, \emptyset, \emptyset) \vdash 0, 0)$.

To find $P_0$, we look at $u_0$, and find that it has $a(x)$ as its result, meaning we can apply Case 4 to obtain $E((\emptyset, \emptyset, \emptyset) \vdash b(a)[\langle 0u_0, 1u_1\rangle].S(a(x), y(x), [\langle 0u_0, 1u_1\rangle], y), b(y).y(x))$. And by applying Case 5 of Definition 4.2, $S(a(x), y(x), [\langle 0u_0, 1u_1\rangle], y) = a_{[\langle 0u_0, 1u_1\rangle]}(x)$. Since we have no more extrusions to add, we apply Case 1 to get our process $P_0 = b(a)[\langle 0u_0, 1u_1\rangle].a_{[\langle 0u_0, 1u_1\rangle]}(x)$.

To find $P_1$, we similarly look at $u_1$ and find that we can apply Case 3. This gives us $P_1 = \overline{b}(a)[\langle 0u_0, 1u_1\rangle].0$.

We can then apply Case 3 to $E((\{(\overline{b}(c), u_2, u_2)\}, \emptyset, \emptyset) \vdash P_0 \mid P_1, b(y).y(x) \mid \overline{b}(a))$. This gives us our final process,

$$\overline{b}(c)[k'].b(a)[k].a_{[k]}(x) \mid \overline{b}(a)[k].0$$

where $k = \langle 0u_0, 1u_1 \rangle$ and $k' = u_2$

We can then show, in Theorem 4.6, that we have an operational correspondence between our two calculi and $E$ preserves transitions. Item 1 states that every transition in $\pi$IH corresponds to one in $\pi$IK process generated by $E$, and Item 2 vice versa.

**Theorem 4.6.** *Given a reachable $\pi IH$ process, $\mathbf{H} \vdash P$, and an action, $\mu$,*

1. *if there exists a location u such that $\mathbf{H} \vdash P \underset{u}{\overset{\mu}{\rightarrowtail}} \mathbf{H}' \vdash P'$ then there exists a key, m, such that $E(\mathsf{lcopy}(\mathbf{H}) \vdash P, P) \xrightarrow{\mu[m]} E(\mathsf{lcopy}(\mathbf{H}') \vdash P', P')$;*
2. *if there exists a key, m, such that $E(\mathsf{lcopy}(\mathbf{H}) \vdash P, P) \xrightarrow{\mu[m]} P''$, then there exists a location, u, and a $\pi IH$ process, $\mathbf{H}' \vdash P'$, such that $\mathbf{H} \vdash P \underset{u}{\overset{\mu}{\rightarrowtail}} \mathbf{H}' \vdash P'$ and $P'' \equiv E(\mathsf{lcopy}(\mathbf{H}') \vdash P', P')$.*

## 5 Bundle event structures

In this section we will recall the definition of *labelled reversible bundle event structures* (LRBESs), which we intend to use later to define the event structure semantics of $\pi$IK and through that $\pi$IH. We also describe some operations on LRBESs, which our semantics will make use of. This section is primarily a review of definitions from [10]. We use bundle event structures, rather than the more common prime event structures, because LRBESs yield more compact event structures with fewer events and simplifies parallel composition.

An LRBES consists of a set of events, $E$, a subset of which, $F$, are reversible,and three relations on them. The bundle relation, $\mapsto$, says that if $X \mapsto e$ then one of the events of $X$ must have happened before $e$ can and all events in $X$ are in conflict with each other. The conflict relation, $\sharp$, says that if $e \ \sharp \ e'$ then $e$ and $e'$ cannot occur in the same configuration. The prevention relation, $\triangleright$, says that if $e \triangleright \underline{e'}$ then $e'$ cannot reverse after $e$ has happened. Since the event structure is labelled, we also have a set of labels $\mathsf{Act}$, and a labelling function $\lambda$ from events to labels. We use $\underline{e}$ to denote $e$ being reversed, and $e^*$ to denote either $e$ or $\underline{e}$.

**Definition 5.1 (Labelled Reversible Bundle Event Structure [10]).** *A labelled reversible bundle event structure is a 7-tuple $\mathcal{E} = (E, F, \mapsto, \sharp, \triangleright, \lambda, \mathsf{Act})$ where:*

1. *$E$ is the set of events;*
2. *$F \subseteq E$ is the set of reversible events;*
3. *the bundle set, $\mapsto \subseteq 2^E \times (E \cup \underline{F})$, satisfies $X \mapsto e^* \Rightarrow \forall e_1, e_2 \in X . e_1 \neq e_2 \Rightarrow e_1 \ \sharp \ e_2$ and for all $e \in F$, $\{e\} \mapsto \underline{e}$;*
4. *the conflict relation, $\sharp \subseteq E \times E$, is symmetric and irreflexive;*
5. *$\triangleright \subseteq E \times \underline{F}$ is the prevention relation.*
6. *$\lambda : E \to \mathsf{Act}$ is a labelling function.*

An event in a LRBES can have multiple possible causes as defined in Definition 5.2. A possible cause $X$ of an event $e$ is a conflict-free set of events which contains a member of each bundle associated with $e$ and contains possible causes of all events in $X$.

**Definition 5.2 (Possible Cause).** *Given an LRBES, $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$ and an event $e \in E$, $X \subseteq E$ is a possible cause of $e$ if*

- *$e \notin X$, $X$ is finite, whenever $X' \mapsto e$ we have $X' \cap X \neq \emptyset$;*
- *for any $e', e'' \in \{e\} \cup X$, we have $e' \not\sharp\, e''$ ($X \cup \{e\}$ is conflict-free);*
- *for all $e' \in X$, there exists $X'' \subseteq X$, such that $X''$ is a possible cause of $e'$;*
- *there does not exist any $X''' \subset X$, such that $X'''$ is a possible cause of $e$.*

Since we want to compare the event structures generated by a process to the operational semantics, we need a notion of transitions on event structures. For this purpose we use configuration systems (CSs), which event structures can be translated into.

**Definition 5.3 (Configuration system [23]).** *A* configuration system *(CS) is a quadruple $C = (E, F, \mathsf{C}, \rightarrow)$ where $E$ is a set of events, $F \subseteq E$ is a set of reversible events, $\mathsf{C} \subseteq 2^E$ is the set of configurations, and $\rightarrow \subseteq \mathsf{C} \times 2^{E \cup \underline{F}} \times \mathsf{C}$ is a labelled transition relation such that if $X \xrightarrow{A \cup \underline{B}} Y$ then:*

- *$X, Y \in \mathsf{C}$, $A \cap X = \emptyset$; $B \subseteq X \cap F$; and $Y = (X \setminus B) \cup A$;*
- *for all $A' \subseteq A$ and $B' \subseteq B$, we have $X \xrightarrow{A' \cup \underline{B'}} Z \xrightarrow{(A \setminus A') \cup \underline{(B \setminus B')}} Y$, meaning $Z = (X \setminus B') \cup A' \in \mathsf{C}$.*

**Definition 5.4 (From LRBES to CS [10]).** *We define a mapping $C_{br}$ from LRBESs to CSs as: $C_{br}((E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})) = (E, F, \mathsf{C}, \rightarrow)$ where:*

1. *$X \in C$ if $X$ is conflict-free;*
2. *For $X, Y \in C$, $A \subseteq E$, and $B \subseteq F$, there exists a transition $X \xrightarrow{A \cup \underline{B}} Y$ if:*
   (a) *$Y = (X \setminus B) \cup A$; $X \cap A = \emptyset$; $B \subseteq X$; and $X \cup A$ conflict-free;*
   (b) *for all $e \in B$, if $e' \rhd \underline{e}$ then $e' \notin X \cup A$;*
   (c) *for all $e \in A$ and $X' \subseteq E$, if $X' \mapsto e$ then $X' \cap (X \setminus B) \neq \emptyset$;*
   (d) *for all $e \in B$ and $X' \subseteq E$, if $X' \mapsto \underline{e}$ then $X' \cap (X \setminus (B \setminus \{e\})) \neq \emptyset$.*

For our semantics we need to define a prefix, restriction, parallel composition, and choice. Causal prefixing takes a label, $\mu$, an event, $e$, and an LRBES, $\mathcal{E}$, and adds $e$ to $\mathcal{E}$ with the label $\mu$ and associating every other event in $\mathcal{E}$ with a bundle containing only $e$. Restriction removes a set of events from an LRBES.

**Definition 5.5 (Causal Prefixes [10]).** *Given an LRBES $\mathcal{E}$, a label $\mu$, and an event $e$, $(\mu)(e).\mathcal{E} = (E', F', \mapsto', \sharp', \rhd', \lambda', \mathsf{Act}')$ where:*

1. *$E' = E \cup e$*
2. *$F' = F \cup e$*
3. *$\mapsto' = \mapsto \cup (\{\{e\}\} \times (E \cup \{\underline{e}\}))$*
4. *$\sharp' = \sharp$*
5. *$\rhd' = \rhd \cup (E \times \{\underline{e}\})$*
6. *$\lambda' = \lambda[e \mapsto \mu]$*
7. *$\mathsf{Act}' = \mathsf{Act} \cup \{\mu\}$*

Removing a set of labels $L$ from an LRBES removes not just events with labels in $A$ but also events dependent on events with labels in $L$.

**Definition 5.6 (Removing labels and their dependants).** *Given an event structure $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$ and a set of labels $L \subseteq \mathsf{Act}$, we define $\rho_{\mathcal{E}}(L) = X$ as the maximum subset of $E$ such that*

1. *if $e \in X$ then $\lambda(e) \notin L$;*
2. *if $e \in X$ then there exists a possible cause of $e$, $x$, such that $x \subseteq X$.*

A choice between LRBESs puts all the events of one event structure in conflict with the events of the others.

**Definition 5.7 (Choice [10]).** *Given LRBESs $\mathcal{E}_0, \mathcal{E}_1, \dots, \mathcal{E}_n$, the choice between them is $\sum_{0 \le i \le n} \mathcal{E}_i = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$ where:*

1. $E = \bigcup_{0 \le i \le n} \{i\} \times E_i$
2. $F = \bigcup_{0 \le i \le n} \{i\} \times F_i$
3. *$X \mapsto e^*$ if $e = (i, e_i)$, $X_i \mapsto_i e_i^*$, and $X = \{i\} \times X_i$*
4. *$(i, e) \sharp (j, e')$ if $i \neq j$ or $e \sharp_i e'$*
5. *$(i, e) \rhd (j, e')$ if $i \neq j$ or $e \sharp_i e'$*
6. $\lambda(j, e) = \lambda_j(e)$
7. $\mathsf{Act} = \bigcup_{0 \le i \le n} \mathsf{Act}_i$

**Definition 5.8 (Restriction [10]).** *Given an LRBES, $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$, restricting $\mathcal{E}$ to $E' \subseteq E$ creates $\mathcal{E} \restriction E' = (E', F', \mapsto', \sharp', \rhd', \lambda', \mathsf{Act}')$ where:*

1. *$F' = F \cap E'$;*
2. *$\mapsto' = \mapsto \cap(\mathcal{P}(E') \times (E' \cup \underline{F'}))$;*
3. *$\sharp' = \sharp \cap(E' \times E')$;*
4. *$\rhd' = \rhd \cap (E' \times \underline{F'})$;*
5. *$\lambda' = \lambda \restriction_{E'}$;*
6. $\mathsf{Act} = \mathsf{ran}(\lambda')$.

For parallel composition we construct a product of event structures, which consists of events corresponding to synchronisations between the two event structures. The possible causes of an event $(e_0, e_1)$ contain a possible cause of $e_0$ and a possible cause of $e_1$.

**Definition 5.9 (Parallel [10]).** *Given two LRBESs $\mathcal{E}_0 = (E_0, F_0, \mapsto_0, \sharp_0, \rhd_0, \lambda_0, \mathsf{Act}_0)$ and $\mathcal{E}_1 = (E_1, F_1, \mapsto_1, \sharp_1, \rhd_1, \lambda_1, \mathsf{Act}_1)$, their parallel composition $\mathcal{E}_0 \times \mathcal{E}_1 = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$ with projections $\pi_0$ and $\pi_1$ where:*

1. *$E = E_0 \times_* E_1 = \{(e, *) \mid e \in E_0\} \cup \{(*, e) \mid e \in E_1\} \cup \{(e, e') \mid e \in E_0$ and $e' \in E_1\}$;*
2. *$F = F_0 \times_* F_1 = \{(e, *) \mid e \in F_0\} \cup \{(*, e) \mid e \in F_1\} \cup \{(e, e') \mid e \in F_0$ and $e' \in F_1\}$;*
3. *for $i \in \{0, 1\}$ we have $(e_0, e_1) \in E$, $\pi_i((e_0, e_1)) = e_i$;*
4. *for any $e^* \in E \cup \underline{F}$, $X \subseteq E$, $X \mapsto e^*$ iff there exists $i \in \{0, 1\}$ and $X_i \subseteq E_i$ such that $X_i \mapsto \pi_i(e)^*$ and $X = \{e' \in E \mid \pi_i(e') \in X_i\}$;*
5. *for any $e, e' \in E$, $e \sharp e'$ iff there exists $i \in \{0, 1\}$ such that $\pi_i(e) \sharp_i \pi_i(e')$, or $\pi_i(e) = \pi_i(e') \neq \bot$ and $\pi_{1-i}(e) \neq \pi_{1-i}(e')$;*
6. *for any $e \in E$, $e' \in F$, $e \rhd \underline{e'}$ iff there exists $i \in \{0, 1\}$ such that $\pi_i(e) \rhd_i \underline{\pi_i(e')}$.*
7. $\lambda(e) = \begin{cases} \lambda_0(e_0) & \text{if } e = (e_0, *) \\ \lambda_1(e_1) & \text{if } e = (*, e_1) \\ \tau & \text{if } e = (e_0, e_1) \text{ and either } \lambda_0(e_0) = a(x) \text{ and } \lambda_1(e_1) = \overline{a}(x) \\ & \text{or } \lambda_0(e_0) = \overline{a}(x) \text{ and } \lambda_1(e_1) = a(x) \\ 0 & \text{otherwise} \end{cases}$
8. $\mathsf{Act} = \{\tau\} \cup \mathsf{Act}_0 \cup \mathsf{Act}_1$

## 6 Event structure semantics of $\pi$IK

In this section we define event structure semantics of $\pi$IK using the LRBESs and operations defined in Section 5. Theorems 6.3 and 6.4 give us an operational correspondence between a $\pi$IK process and the generated event structure. Together with Theorem 4.6, this gives us a correspondence between a $\pi$IH process and the event structure it generates by going via a $\pi$IK process.

As we want to ensure that all free and bound names in our process are distinct, we modify our syntax for replication, assigning each replication an infinite set, $\mathbf{x}$, of names to substitute into the place of bound names in each created copy of the process, so that

$$!_{\mathbf{x}} P \equiv !_{\mathbf{x}\setminus\{x_0,\dots,x_k\}} P | P\{^{x_0,\dots,x_k}/_{a_0,\dots,a_k}\} \text{ if } \{x_0,\dots,x_k\} \subseteq \mathbf{x} \text{ and } \mathsf{bn}(P) = \{a_0,\dots,a_k\}$$

Before proceeding to the semantics we also define the standard bound names of a process $P$, $\mathsf{sbn}(P)$, meaning the names that would be bound in $P$ if every action was reversed, in Definition 6.1.

**Definition 6.1.** *The standard bound names of a process $P$, $\mathsf{sbn}(P)$, are defined as:*

$$\begin{array}{ll}
\mathsf{sbn}(a(x).P') = \{x\} \cup \mathsf{sbn}(P') & \mathsf{sbn}(a(x)[m].P') = \{x\} \cup \mathsf{sbn}(P') \\
\mathsf{sbn}(\overline{a}(x).P') = \{x\} \cup \mathsf{sbn}(P') & \mathsf{sbn}(\overline{a}(x)[m].P') = \{x\} \cup \mathsf{sbn}(P') \\
\mathsf{sbn}(P_0|P_1) = \mathsf{sbn}(P_0) \cup \mathsf{sbn}(P_1) & \mathsf{sbn}(P_0 + P_1) = \mathsf{sbn}(P_0) \cup \mathsf{sbn}(P_1) \\
\mathsf{sbn}(\nu x)P' = \{x\} \cup \mathsf{sbn}(P') & \mathsf{sbn}(!_{\mathbf{x}} P) = \mathbf{x}
\end{array}$$

We can now define the event structure semantics in Table 6. We do this using rules of the form $\{\!|P|\!\}_{(\mathcal{N},l)} = \langle \mathcal{E}, \mathsf{Init}, k\rangle$ where $l$ is the level of unfolding of replication, $\mathcal{E}$ is an LRBES, $\mathsf{Init}$ is the initial configuration, $\mathcal{N} \supseteq n(P)$ is a set of names, which any input in the process could receive, and $k : \mathsf{Init} \to \mathcal{K}$ is a function assigning communication keys to the past actions, which we use in parallel composition to determine which synchronisations of past actions to put in $\mathsf{Init}$. We define $\{\!|P|\!\}_{\mathcal{N}} = \sup_{l\in\mathbb{N}} \{\!|P|\!\}_{(\mathcal{N},l)}$

The denotational semantics in Table 6 make use of of the LRBES operators defined in Section 5. The choice and output cases are straightforward uses of the choice and causal prefix operators. The input creates a case for prefixing an input of each name in $\mathcal{N}$ and a choice between the cases. We have two cases for restriction, one for restriction originating from a past communication and another for restriction originating from the original process. If the restriction does not originate from the original process, then we ignore it, otherwise we remove events which would use the restricted channel and their causes. The parallel composition uses the parallel operator, but additionally needs to consider link causation caused by the early semantics. Each event labelled with an input of a name in standard bound names gets a bundle consisting of the event labelled with the output on that name. And each output event is prevented from reversing by the input names receiving that name. This way, inputs on extruded names are caused by the output that made the name free. Replication substitutes the names and counts down the level of replication.

Note that the only difference between a future and a past action is that the event corresponding to a past action is put in the initial state and given a communication key.

$$\{\![0]\!\}_{(\mathcal{N},l)} = \langle(\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\emptyset),\emptyset,\emptyset\rangle$$

$$\{\![P_0 + P_1]\!\}_{(\mathcal{N},l)} = \langle \mathcal{E}_0 + \mathcal{E}_1, \{0\}\times\mathsf{Init}_0 \cup \{1\}\times \mathsf{Init}_1, k((i,e)) = k_i(e)\rangle \text{ where } \{\![P_i]\!\} = \langle \mathcal{E}_i, \mathsf{Init}_i, k_i\rangle \text{ for } i \in \{0,1\}$$

$$\left\{\!\!\left[\overline{a}(n).P\right]\!\!\right\}_{(\mathcal{N},l)} = \left\langle \overline{a}(n)(e).\mathcal{E}_P, \mathsf{Init}_P, k_P\right\rangle \text{ for some fresh } e \notin E \text{ where } \{\![P]\!\}_{(\mathcal{N},l)} = \langle \mathcal{E}_P, \mathsf{Init}_P, k_P\rangle$$

$$\{\![a(x).P]\!\}_{(\mathcal{N},l)} = \left\langle \sum_{n\in(\mathcal{N}\setminus\mathsf{sbn}(P))} a(n)(e).\mathcal{E}_{P_n}, \bigcup_{n\in(\mathcal{N}\setminus\mathsf{sbn}(P))}\{n\}\times\mathsf{Init}_{P_n}, (n,e)\mapsto k_{P_n}(e)\right\rangle$$
for some fresh $e_n \notin E_n$ where $\{\![P[x := n]]\!\}_{(\mathcal{N},l)} = \left\langle \mathcal{E}_{P_n}, \mathsf{Init}_{P_n}, k_{P_n}\right\rangle$

$$\left\{\!\!\left[\overline{a}(n)[m].P\right]\!\!\right\}_{(\mathcal{N},l)} = \left\langle \overline{a}(n)(e).\mathcal{E}_P, \mathsf{Init}_P \cup \{e\}, k_P[e\mapsto m]\right\rangle \text{ for some fresh } e \notin E \text{ where } \{\![P]\!\}_{(\mathcal{N},l)} = \langle \mathcal{E}_P, \mathsf{Init}_P, k_P\rangle$$

$$\{\![a(b)[m].P]\!\}_{(\mathcal{N},l)} = \left\langle \sum_{n\in(\mathcal{N}\setminus\mathsf{sbn}(P))} a(n)(e_n).\mathcal{E}_{P_n}, \Big(\bigcup_{n\in(\mathcal{N}\setminus\mathsf{sbn}(P))}\{n\}\times\mathsf{Init}_{P_n}\Big)\cup\{(b,e_b)\}, k\right\rangle$$
for some fresh $e_n \notin E_n$ where
$$\{\![P[b_{[m]} := n]]\!\}_{(\mathcal{N},l)} = \left\langle \mathcal{E}_{P_n}, \mathsf{Init}_{P_n}, k_{P_n}\right\rangle$$
$$k((n,e)) = \begin{cases} m & \text{if } e = e_b \text{ and } n = b\\ k_{P_n}(e) & \text{otherwise}\end{cases}$$

$$\{\![(\nu a)P]\!\}_{(\mathcal{N},l)} = \langle \mathcal{E}\restriction E_\alpha, \mathsf{Init}\cap E_\alpha, k\restriction E_\alpha)\rangle \text{ where: } \{\![P]\!\}_{(\mathcal{N},l)} = \langle\mathcal{E},\mathsf{Init},k\rangle,\ E_\alpha = \rho(\{\alpha \mid a \in n(\alpha)\}$$
if whenever there exist past actions $b(a)[m]$ and $\overline{b}(a)[m]$ in $P$ then they are guarded by a restriction $(\nu a)$ in $P$

$$\{\![(\nu a)P]\!\}_{(\mathcal{N},l)} = \langle \mathcal{E},\mathsf{Init},k\rangle \text{ where: } \{\![P]\!\}_{(\mathcal{N},l)} = \langle\mathcal{E},\mathsf{Init},k\rangle$$
if there exist past actions $b(a)[m]$ and $\overline{b}(a)[m]$ in $P$ which are not guarded by a restriction $(\nu a)$ in $P$

$$\{\![P_0|P_1]\!\}_{(\mathcal{N},l)} = \left\langle (E,F,\mapsto,\sharp,\rhd,\lambda,\mathsf{Act})\restriction\{e \mid \lambda(e)\neq 0\}, \mathsf{Init}, k\right\rangle \text{ where}$$
for $i \in \{0,1\}$, $\{\![P_i]\!\}_l = \langle \mathcal{E}_i, \mathsf{Init}_i, k_i\rangle$

$(E_0,F_0,\mapsto_0,\sharp_0,\rhd_0)\times(E_0,F_0,\mapsto_0,\sharp_0,\rhd_0) = (E,F,\mapsto',\sharp,\rhd')$

$\mathsf{Init} = \{(e_0,*) | e_0 \in \mathsf{Init}_0 \text{ and } \nexists e_1 \in \mathsf{Init}_1.k_1(e_1) = k_0(e_0)\}\cup$
$\{(*,e_1) | e_1 \in \mathsf{Init}_1 \text{ and } \nexists e_0 \in \mathsf{Init}_0.k_1(e_1) = k_0(e_0)\}\cup$
$\{(e_0,e_1) | e_0 \in \mathsf{Init}_0 \text{ and } e_1 \in \mathsf{Init}_1 \text{ and } k_1(e_1) = k_0(e_0)\}$

$X \mapsto e$ if $X \mapsto' e$ or there exists $x \in \mathsf{no}(\lambda(e))$ such that $X = \{e' \mid \exists a.\lambda(e') = \overline{a}(x)\}$ and $x \in \mathsf{sbn}(P)$

$e \rhd \underline{e'}$ if either $e \rhd' \underline{e'}$ or there exists $x \in \mathsf{no}(\lambda(e))$ and $a$ such that $\lambda(e') = \overline{a}(x)$

$$k(e) = \begin{cases} k_0(e_0) & \text{if } e = (e_0,*)\\ k_1(e_1) & \text{if } e = (*,e_1)\\ k_0(e_0) & \text{if } e = (e_0,e_1)\end{cases}$$

$$\{\![!_{\mathbf{x}}P]\!\}_{(\mathcal{N},0)} = \langle(\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\emptyset),\emptyset,\emptyset\rangle$$

$$\{\![!_{\mathbf{x}}P]\!\}_{(\mathcal{N},l)} = \left\{\!\!\left[!_{\mathbf{x}\setminus\{x_0,\dots,x_k\}}P|P\{^{x_0,\dots,x_k}/_{a_0,\dots,a_k}\}\right]\!\!\right\}_{(\mathcal{N},l-1)} \text{ if } \{x_0,\dots,x_k\}\subseteq\mathbf{x} \text{ and } \mathsf{bn}(P) = \{a_0,\dots,a_k\}$$

**Table 6.** Denotational event structure semantics of $\pi$IK

*Example 6.2.* Consider the process $a(b)[n] \mid \overline{a}(b)[n]$. Our event structure semantics generate a LRBES $\{\!\!\{ a(x)[n] \mid \overline{a}(b[n]) \}\!\!\}_{\{a,b,x\}} = \left\langle (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act}), \mathsf{Init}, k \right\rangle$ where:

$$\begin{array}{ll}
E = F = \{a(b), a(a), a(x), \overline{a}(b), \tau\} & \lambda(e) = e \\
\{\overline{a}(b)\} \mapsto a(b) & \mathsf{Act} = \{a(b), a(a), a(x), \overline{a}(b), \tau\} \\
a(b) \sharp a(a),\ a(b) \sharp a(x),\ a(a) \sharp a(x), & \mathsf{Init} = \{\tau\} \\
a(b) \sharp \tau,\ a(a) \sharp \tau,\ a(x) \sharp \tau,\ \overline{a}(b) \sharp \tau & k(\tau) = n \\
a(b) \rhd \overline{a}(b) &
\end{array}$$

From this we see that (1) receiving $b$ is causally dependent on sending $b$, (2) all the possible inputs on $a$ are in conflict with one another, (3) the synchronisation between the input and the output is in conflict with either happening on their own, and (4) since the two past actions have the same key, the initial state contains their synchronisation.

We show in Theorems 6.3 and 6.4 that given a process $P$ with a conflict-free initial state, including any reachable process, performing a transition $P \xrightarrow{\mu[m]} P'$ does not affect the event structure, as $\{\!\!\{ P \}\!\!\}_{\mathcal{N}}$ and $\{\!\!\{ P' \}\!\!\}_{\mathcal{N}}$ are isomorphic. It also means we have an event $e$ labelled $\mu$ such that $e$ is available in $P$'s initial state, and $P'$'s initial state is $P$'s initial state with $e$ added. A similar event can be removed to correspond to a reverse action.

**Theorem 6.3.** *Let $P$ be a forwards reachable process wherein all bound and free names are different and let $\mathcal{N} \supseteq n(P)$ be a set of names. If (1) $\{\!\!\{ P \}\!\!\}_{\mathcal{N}} = \langle \mathcal{E}, \mathsf{Init}, k \rangle$ where $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$, and $\mathsf{Init}$ is conflict-free, and (2) there exists a transition $P \xrightarrow{\mu[m]} P'$ such that $\{\!\!\{ P' \}\!\!\}_{\mathcal{N}} = \left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$, then there exists an isomorphism $f : \mathcal{E} \to \mathcal{E}'$ and a transition in $C_{br}(\mathcal{E})$, $\mathsf{Init} \xrightarrow{\{e\}} X$, such that $\lambda(e) = \mu$, $f \circ k' = k[e \mapsto m]$, and $f(X) = \mathsf{Init}'$.*

**Theorem 6.4.** *Let $P$ be a forwards reachable process wherein all bound and free names are different and let $\mathcal{N} \supseteq n(P)$ be a set of names. If (1) $\{\!\!\{ P \}\!\!\}_{\mathcal{N}} = \langle \mathcal{E}, \mathsf{Init}, k \rangle$ where $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$, and (2) there exists a transition $\mathsf{Init} \xrightarrow{\{e\}} X$ in $C_{br}(\mathcal{E})$, then there exists a transition $P \xrightarrow{\mu[m]} P'$ such that $\{\!\!\{ P' \}\!\!\}_{\mathcal{N}} = \left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$ and an isomorphism $f : \mathcal{E} \to \mathcal{E}'$ such that $\lambda(e) = \mu$, $f \circ k' = k[e \mapsto m]$, and $f(X) = \mathsf{Init}'$.*

By Theorems 4.6, 6.3, and 6.4 we can combine the event structure semantics of $\pi$IK and mapping $E$ (Definition 4.4) and get an operational correspondence between $\mathbf{H} \vdash P$ and the event structure $\{\!\!\{ E(\mathsf{lcopy}(\mathbf{H}) \vdash P, P) \}\!\!\}_{n(E(\mathsf{lcopy}(\mathbf{H}) \vdash P, P))}$.

## 7 Conclusion and future work

All existing reversible versions of the $\pi$-calculus use reduction semantics [14,26] or late semantics [7,17], despite the early semantics being used more widely than the late in the forward-only setting. We have introduced $\pi$IH, the first reversible early $\pi$-calculus. It is a reversible form of the *internal* $\pi$-calculus, where names being sent in output actions are always bound. As well as structural causation, as in CCS, the early form of the internal $\pi$-calculus also has a form of link causation created by the semantics being early, which is not present in other reversible $\pi$-calculi. In $\pi$IH past actions are tracked by using extrusion histories adapted from [12], which move past actions and their locations into separate histories for dynamic reversibility. We mediate the event structure semantics of $\pi$IH via a statically reversible version of the internal $\pi$-calculus, $\pi$IK, which keeps the structure of the process intact but annotates past actions with keys, similarly to $\pi$K [17] and CCSK [21]. We showed that a process $\pi$IH with extrusion histories can be mapped to a $\pi$IK process with keys, creating an operational correspondence (Theorem 4.6).

The event structure semantics of $\pi$IK, and by extension $\pi$IH, are defined inductively on the syntax of the process. We use labelled reversible bundle event structures [10], rather than prime event structures, to get a more compact representation where each action in the calculus has only one corresponding event. While causation in the internal $\pi$-calculus is simpler that in the full $\pi$-calculus, our early semantics means that we still have to handle link causation, in the form of an input receiving a free name being caused by a previous output of that free name. We show an operational correspondence between $\pi$IK processes and their event structure representations in Theorems 6.3 and 6.4. Cristescu *et al.* [8] have used rigid families [4], related to event structures, to describe the semantics of R$\pi$ [7]. However, unlike our denotational event structure semantics, their semantics require one to reverse every action in the process before applying the mapping to a rigid family, and then redo every reversed action in the rigid family. Our approach of using a static calculus as an intermediate step means we get the current state of the event structure immediately, and do not need to redo the past steps.

*Future work:* We could expand the event structure semantics of $\pi$IK to $\pi$K. This would entail significantly more link causation, but would give us event structure semantics of a full $\pi$-calculus. Another possibility is to expand $\pi$IH to get a full reversible early $\pi$-calculus.

*Acknowledgements:* We thank Thomas Hildebrandt and Håkon Normann for discussions on how to translate their work on $\pi$-calculus with extrusion histories to a reversible setting. We thank the anonymous reviewers of RC 2020 for their helpful comments.

This work was partially supported by an EPSRC DTP award; also by the following EPSRC projects: EP/K034413/1, EP/K011715/1, EP/L00058X/1, EP/N027833/1, EP/T006544/1, EP/N028201/1 and EP/T014709/1; and by EU COST Action IC1405 on Reversible Computation.

## References

1. Aubert, C., Cristescu, I.: Contextual equivalences in configuration structures and reversibility. JLAMP **86**(1), 77 – 106 (2017). https://doi.org/10.1016/j.jlamp.2016.08.004
2. Boreale, M.: On the expressiveness of internal mobility in name-passing calculi. Theoretical Computer Science **195**(2), 205 – 226 (1998). https://doi.org/10.1016/S0304-3975(97)00220-X

3. Boudol, G., Castellani, I.: Permutation of transitions: An event structure semantics for CCS and SCCS. In: Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency. pp. 411–427. No. 354 in LNCS, Springer, Berlin, Heidelberg (1989). https://doi.org/10.1007/BFb0013028
4. Castellan, S., Hayman, J., Lasson, M., Winskel, G.: Strategies as concurrent processes. Electr. Notes Theor. Comput. Sci. **308**, 87–107 (2014). https://doi.org/10.1016/j.entcs.2014.10.006
5. Crafa, S., Varacca, D., Yoshida, N.: Compositional event structure semantics for the internal $\pi$-calculus. In: CONCUR. pp. 317–332. Springer Berlin Heidelberg, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74407-8_22
6. Crafa, S., Varacca, D., Yoshida, N.: Event Structure Semantics of Parallel Extrusion in the Pi-Calculus. In: FOSSACS. pp. 225–239. No. 7213 in LNCS, Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28729-9_15
7. Cristescu, I., Krivine, J., Varacca, D.: A compositional semantics for the reversible pi-calculus. In: LICS. pp. 388–397. IEEE Computer Society, Washington, DC, USA (2013). https://doi.org/10.1109/LICS.2013.45
8. Cristescu, I., Krivine, J., Varacca, D.: Rigid families for the reversible $\pi$-calculus. In: RC. LNCS, vol. 9720, pp. 3–19. Springer (2016). https://doi.org/10.1007/978-3-319-40578-0_1
9. Danos, V., Krivine, J.: Reversible Communicating Systems. In: CONCUR. pp. 292–307. No. 3170 in LNCS, Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28644-8_19
10. Graversen, E., Phillips, I., Yoshida, N.: Event structure semantics of (controlled) reversible CCS. In: RC. pp. 102–122 (2018). https://doi.org/10.1007/978-3-319-99498-7_7
11. Graversen, E., Phillips, I., Yoshida, N.: Event structures for the reversible early internal pi-calculus. arXiv:2004.01211 [cs.FL] (2020), `https://arxiv.org/abs/2004.01211`
12. Hildebrandt, T.T., Johansen, C., Normann, H.: A stable non-interleaving early operational semantics for the pi-calculus. In: LATA. pp. 51–63. No. 10168 in LNCS, Springer, Berlin, Heidelberg (2017). https://doi.org/10.1007/978-3-319-53733-7_3
13. Honda, K., Yoshida, N.: On reduction-based process semantics. TCS **151**(2), 437 – 486 (1995). https://doi.org/10.1016/0304-3975(95)00074-7
14. Lanese, I., Mezzina, C.A., Stefani, J.B.: Reversing Higher-Order Pi. In: CONCUR. pp. 478–493. No. 6269 in LNCS, Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15375-4_33
15. Lanese, I., Mezzina, C.A., Stefani, J.B.: Reversibility in the higher-order $\pi$-calculus. Theoretical Computer Science **625**, 25 – 84 (2016). https://doi.org/10.1016/j.tcs.2016.02.019
16. Medić, D., Mezzina, C.A.: Static VS Dynamic Reversibility in CCS. In: RC. LNCS, vol. 9720, pp. 36–51. Springer International Publishing (2016). https://doi.org/10.1007/978-3-319-40578-0_3
17. Medic, D., Mezzina, C.A., Phillips, I., Yoshida, N.: A parametric framework for reversible pi-calculi. In: EXPRESS/SOS. pp. 87–103 (2018). https://doi.org/10.4204/EPTCS.276.8
18. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, I and II. Inf. Comput. **100**(1), 1–77 (1992). https://doi.org/10.1016/0890-5401(92)90008-4
19. Milner, R., Parrow, J., Walker, D.: Modal logics for mobile processes. Theor. Comput. Sci. **114**(1), 149–171 (1993). https://doi.org/10.1016/0304-3975(93)90156-N
20. Milner, R., Sangiorgi, D.: Barbed bisimulation. In: ICALP. pp. 685–695. Springer-Verlag, Berlin, Heidelberg (1992). https://doi.org/10.5555/646246.684864
21. Phillips, I., Ulidowski, I.: Reversing algebraic process calculi. JLAMP **73**(1-2), 70–96 (2007). https://doi.org/10.1016/j.jlap.2006.11.002
22. Phillips, I., Ulidowski, I.: Reversibility and models for concurrency. Electr. Notes Theor. Comput. Sci. **192**(1), 93–108 (2007). https://doi.org/10.1016/j.entcs.2007.08.018
23. Phillips, I., Ulidowski, I.: Reversibility and asymmetric conflict in event structures. JLAMP **84**(6), 781 – 805 (2015). https://doi.org/10.1016/j.jlamp.2015.07.004

24. Sangiorgi, D.: $\pi$-calculus, internal mobility, and agent-passing calculi. Theoretical Computer Science **167**(1), 235 – 274 (1996). https://doi.org/10.1016/0304-3975(96)00075-8
25. Sewell, P., Wojciechowski, P.T., Unyapoth, A.: Nomadic pict: Programming languages, communication infrastructure overlays, and semantics for mobile computation. ACM Trans. Program. Lang. Syst. **32**(4), 12:1–12:63 (2010). https://doi.org/10.1145/1734206.1734209
26. Tiezzi, F., Yoshida, N.: Reversible session-based pi-calculus. JLAMP **84**(5), 684 – 707 (2015). https://doi.org/10.1016/j.jlamp.2015.03.004
27. Winskel, G.: Event structure semantics for CCS and related languages. In: ICALP. pp. 561–576. No. 140 in LNCS, Springer, Berlin, Heidelberg (1982). https://doi.org/10.1007/BFb0012800

# A Section 2

**Lemma A.1.** *Let $P$ be a process. If there exists an extrusion history $\mathbf{H}$ such that $\mathbf{H} \vdash P \overset{\mu}{\underset{u}{\rightsquigarrow}} \mathbf{H}' \vdash P'$ then there exists $L$ such that $\mathbf{H} = \mathbf{H}' + L$, and for any extrusion history $\mathbf{H}''$ not containing $L$, $\mathbf{H}'' + L \vdash P \overset{\mu}{\underset{u}{\rightsquigarrow}} \mathbf{H}'' \vdash P'$.*

*Proof.* [SCOPE$^{-1}$] and [PAR$_i^{-1}$] simply propagate the changes to extrusion histories, and [COM$_i^{-1}$], [IN$^{-1}$], and [OUT$^{-1}$] remove exactly one extrusion from the histories, which is the only one they depend on.

*Proof (Proof of Proposition 2.4).*

1. We prove this by induction in $\mathbf{H} \vdash P \xrightarrow[u]{\alpha} \mathbf{H}' \vdash Q$:

   **[SCOPE]** In this case $P = (\nu x)P'$ and $Q = (\nu x)Q'$, $x \notin n(\alpha)$, and by induction $\mathbf{H}' \vdash Q' \overset{\alpha}{\underset{u}{\rightsquigarrow}} \mathbf{H} \vdash P'$. From rule [SCOPE$^{-1}$] we therefore get $\mathbf{H}' \vdash Q \overset{\alpha}{\underset{u}{\rightsquigarrow}} \mathbf{H} \vdash P$.

   **[PAR$_i$]** In this case $P = P_0|P_1$ and $Q = Q_0|Q_1$, $P_{1-i} = Q_{1-i}$, in $\alpha = \overline{a}(n)$ then $n \notin \mathtt{fn}(P_{1-i})$, and by induction $([\check{i}]\overline{H}', [\check{i}]\underline{H}', [\check{i}]H') \vdash Q_i \overset{\alpha}{\underset{u}{\rightsquigarrow}} ([\check{i}]\overline{H}, [\check{i}]\underline{H}, [\check{i}]H) \vdash P_i$, meaning according to rule [PAR$_i^{-1}$], $\mathbf{H}' \vdash Q \overset{\alpha}{\underset{u}{\rightsquigarrow}} \mathbf{H} \vdash P$.

   **[COM$_i$]** In this case $P = P_0|P_1$ and $Q = Q_0|Q_1$, $n \notin \mathtt{fn}(P_j)$, $\overline{H} = \overline{H}'$, $\underline{H} = \underline{H}'$, $H' = H \cup 0\{(n, (0v_0, 1v_1))\}$, and by induction and Lemma A.1, we have $([\check{i}]\overline{H}', [\check{i}]\underline{H}', [\check{i}]H') \vdash Q_i \overset{\overline{a}(n)}{\underset{v_i}{\rightsquigarrow}} ([\check{i}]\overline{H}, [\check{i}]\underline{H}, [\check{i}]H) \vdash P_i$ and $([\check{j}]\overline{H}', [\check{j}]\underline{H}', [\check{j}]H') \vdash Q_j \overset{a(n)}{\underset{v_j}{\rightsquigarrow}} ([\check{j}]\overline{H}, [\check{j}]\underline{H}, [\check{j}]H) \vdash P_j$. This means according to [COM$^{-1}$], $\mathbf{H}' \vdash Q \overset{\alpha}{\underset{u}{\rightsquigarrow}} \mathbf{H} \vdash P$.

   **[STR]** In this case $Q \equiv Q'$,$\mathbf{H}' \vdash Q' \overset{\alpha}{\underset{u}{\rightsquigarrow}} \mathbf{H} \vdash P'$, and $P' \equiv P$, and by rule [STR$^{-1}$], $\mathbf{H}' \vdash Q \overset{\alpha}{\underset{u}{\rightsquigarrow}} \mathbf{H} \vdash P$.

**[OUT]** In this Case $P = \sum_{i \in I} \alpha_i.P_i$, $Q = P_j$ and $\alpha = \overline{a}(n) = \alpha_j$ for some $j \in I$, and by [OUT$^{-1}$], $\mathbf{H}' \vdash Q \overset{\alpha}{\underset{u}{\rightsquigarrow}} \mathbf{H} \vdash P$.

**[IN]** Similar to [OUT].

2. Similar to previous.

**Definition A.2 (Location).** *Given a location u, its set of paths is defined as*

$$\mathsf{loc}(u) = \begin{cases} \{l\} & \text{if } u = l[P][P'] \\ \{ll_0, ll_1\} & \text{if } u = l \left\langle l_0[P_0][P_0'], l_1[P_1][P_0] \right\rangle \end{cases}$$

To get causal semantics of $\pi$IH, we add a set of causes to each transition, consisting of the previous extrusions, from the output history, which extruded the names of the action.

**Definition A.3 (Causal semantics).** *The early causal semantics consist of transitions of the form* $\mathbf{H} \vdash P \xrightarrow[u,D]{\alpha} \mathbf{H}' \vdash P$ *where* $\mathbf{H} \vdash P \xrightarrow[u]{\alpha} \mathbf{H}' \vdash P$ *and*

1. $(n, u) \in D \Rightarrow \exists a.\ (\overline{a}(n), u) \in \overline{H}$;
2. *if* $(n, l), (n, l') \in D$ *then* $l = l'$;
3. $\mathsf{dom}(D) = \mathsf{dom}(\overline{H}) \cap \mathsf{no}(\alpha)$ *where* $\mathsf{no}(\alpha)$ *is the set of non-output names in* $\alpha$*, defined by* $\mathsf{no}(\overline{a}(b)) = \{a\} \setminus \{b\}$, $\mathsf{no}(a(b)) = \{a, b\}$ *and* $\mathsf{no}(\tau) = \emptyset$.

**Definition A.4 (Independence).** *Two locations,* $u_0$ *and* $u_1$*, are independent if for all* $l_0 \in \mathsf{loc}(u_0)$ *and* $l_1 \in \mathsf{loc}(u_1)$*, there exist* $l$, $l_0'$, $l_1'$ *such that either* $l_0 = l0l_0'$ *and* $l_1 = l1l_1'$ *or* $l_0 = l1l_0'$ *and* $l_1 = l0l_1'$.

*Two transitions* $\overset{\alpha_0}{\underset{u_0,D_0}{\rightarrowtail}}$ *and* $\overset{\alpha_1}{\underset{u_1,D_1}{\rightarrowtail}}$ *are independent if* $u_0$ *and* $u_1$ *are independent, there does not exist n such that* $D_i(n) = u_{1-i}$.

**Proposition A.5 (Forward diamond [9]).** *If* $\mathbf{H} \vdash P \overset{\alpha_0}{\underset{u_0,D_0}{\rightarrowtail}} \mathbf{H}_0 \vdash P_0$ *and* $\mathbf{H} \vdash P \overset{\alpha_1}{\underset{u_1,D_1}{\rightarrowtail}} \mathbf{H}_1 \vdash P_1$ *are independent transitions then there exists* $\mathbf{H}' \vdash P'$ *such that* $\mathbf{H}_0 \vdash P_0 \overset{\alpha_1}{\underset{u_1,D_1}{\rightarrowtail}} \mathbf{H}' \vdash P'$ *and* $\mathbf{H}_1 \vdash P_1 \overset{\alpha_0}{\underset{u_0,D_0}{\rightarrowtail}} \mathbf{H}' \vdash P'$.

*Proof (Proof of Proposition A.5).* This proof is similar to Theorem 14 of [12]. We have a path $l$ such that $u_i = l0u_i'$ and $u_{1-i} = l1u_{1-i}'$. If $Q|R$ is the parallel composition at location $l$, then $([\check{l}0]\overline{H}, [\check{l}0]\underline{H}, [\check{l}0]H) \vdash Q \overset{\alpha_i}{\underset{u_i'}{\rightarrowtail}}$ and $([\check{l}1]\overline{H}, [\check{l}1]\underline{H}, [\check{l}1]H) \vdash Q \overset{\alpha_{1-i}}{\underset{u_{1-i}'}{\rightarrowtail}}$ and there does not exist $n$ such that $D_i(n) = u_{1-i}$ or there does not exist $n$ such that $D_{1-i}(n) = u_i$, and by [PAR$_i$] and [PAR$_i^-1$], this means $\mathbf{H}_0 \vdash P_0 \xrightarrow[u_1]{\alpha_1} \mathbf{H}_0' \vdash P'$ and $\mathbf{H}_1 \vdash P_1 \xrightarrow[u_0]{\alpha_0} \mathbf{H}_1' \vdash P'$ and by Lemma 15 of [12], $\mathbf{H}_0' = \mathbf{H}_1'$.

**Definition A.6 (Trace equivalence).** *We define trace equivalence $\sim$ as the least equivalence relation closed under composition such that:*

$$\mathsf{t};\underline{\mathsf{t}} \sim \varepsilon_{\mathsf{t}}$$

$$\mathbf{H} \vdash P \overset{\alpha_0}{\underset{u_0,D_0}{\rightarrowtail}} \overset{\alpha_1}{\underset{u_1,D_1}{\rightarrowtail}} \mathbf{H}' \vdash P' \sim \mathbf{H} \vdash P \overset{\alpha_1}{\underset{u_1,D_1}{\rightarrowtail}} \overset{\alpha_0}{\underset{u_0,D_0}{\rightarrowtail}} \mathbf{H}' \vdash P'$$

$$\textit{if } \overset{\alpha_1}{\underset{u_1,D_1}{\rightarrowtail}} \textit{ and } \overset{\alpha_0}{\underset{u_0,D_0}{\rightarrowtail}} \textit{ are independent}$$

**Proposition A.7 (Parabola).** *Let* $\mathsf{t}$ *be a trace, then there exists a forward trace* $\mathsf{t}_f$ *and a backward trace* $\mathsf{t}_b$ *such that* $\mathsf{t} \sim \mathsf{t}_b;\mathsf{t}_f$.

*Proof.* We say that $\mathsf{t} = \mathbf{H} \vdash P \overset{\alpha_0}{\underset{u_0,D_0}{\rightarrowtail}} \mathbf{H}_0 \vdash P_0 \overset{\alpha_1}{\underset{u_1,D_1}{\rightarrowtail}} \ldots \overset{\alpha_n}{\underset{u_n,D_n}{\rightarrowtail}} \mathbf{H}' \vdash Q$ and $\mathsf{t}_b;\mathsf{t}_f =$ $\mathbf{H} \vdash P \overset{\alpha'_0}{\underset{u'_0,D'_0}{\rightsquigarrow}} \ldots \overset{\alpha'_k}{\underset{u'_k,D'_k}{\rightsquigarrow}} \mathbf{H}'_k \vdash P'_k \xrightarrow[u'_{k+1},D'_{k+1}]{\alpha'_{k+1}} \ldots \xrightarrow[u'_m,D'_m]{\alpha'_m} \mathbf{H} \vdash Q$.

We prove that they are equivalent by induction on the number of pairs $\xrightarrow[u_i,D_i]{\alpha_i}$, $\overset{\alpha_{i+1}}{\underset{u_{i+1},D_{i+1}}{\rightsquigarrow}}$ and the length of the trace.

If no such pair exists, then $\mathsf{t} = \mathsf{t}_b;\mathsf{t}_f$, otherwise we find the first such pair $\xrightarrow[u_i,D_i]{\alpha_i}$, $\overset{\alpha_{i+1}}{\underset{u_{i+1}}{\rightsquigarrow}}$.

If $u_i = u_{i+1}$ and $\alpha_i = \alpha_{i+1}$ then by Proposition 2.4, $\mathbf{H}_{i-1} \vdash P_{i-1} = \mathbf{H}_{i+1} \vdash P_{i+1}$, and we have a shorter trace $\mathbf{H} \vdash P \overset{\alpha_0}{\underset{u_0,D_0}{\rightarrowtail}} \mathbf{H}_0 \ldots \mathbf{H}_{i-1} \vdash P_{i-1} \overset{\alpha_{i+2}}{\underset{u_{i+2},D_{i+2}}{\rightarrowtail}} \ldots \overset{\alpha_n}{\underset{u_n,D_n}{\rightarrowtail}} \mathbf{H}' \vdash Q \sim$ $\mathsf{t}$.

If $u_i \neq u_{i+1}$ or $\alpha_i \neq \alpha_{i+1}$ then if $u_i \npreceq u_{i+1}$ and $u_{i+1} \npreceq u_i$ then by Proposition A.5, we have a trace $\mathbf{H} \vdash P \overset{\alpha_0}{\underset{u_0,D_0}{\rightarrowtail}} \mathbf{H}_0 \ldots \mathbf{H}_{i-1} \vdash P_{i-1} \overset{\alpha_{i+1}}{\underset{u_{i+1},D_{i+1}}{\rightarrowtail}} \overset{\alpha_i}{\underset{u_i,D_i}{\rightarrowtail}} \mathbf{H}_{i+1} \vdash P_{i+1} \overset{\alpha_{i+2}}{\underset{u_{i+2},D_{i+2}}{\rightarrowtail}}$ $\ldots \overset{\alpha_n}{\underset{u_n,D_n}{\rightarrowtail}} \mathbf{H}' \vdash Q \sim \mathsf{t}$. If $u_i \preceq u_{i+1}$ then, since $\xrightarrow[u_i,D_i]{\alpha_i}$ is the most recent action in $\mathbf{H}_i$, $u_i = u_{i+1}$ and $\alpha_i \neq \alpha_{i+1}$. If $u_{i+1} \preceq u_i$ then, if $u_{i+1} = l[P_a][P_b]$, $P_b$ is not the subprocess located at location $l$ of $P_i$, meaning there cannot exist a transition $\mathbf{H}_i \vdash P_i \overset{\alpha_{i+1}}{\underset{u_{i+1},D_{i+1}}{\rightsquigarrow}}$.

## B Section 3

**Lemma B.1.** *Given a forwards reachable process* $P$*, if* $P \xrightarrow{\overline{a}(x)[n]}$ *then there cannot exist a past output action* $\overline{b}(x)[m]$ *anywhere in* $P$.

*Proof.* This would require $\overline{b}(x)[m]$ to either prefix, be in parallel with, or be an alternative choice to $\overline{a}(x)$ in $P$. The first two cases are impossible due to the if $\mu = \overline{a}(x)$ then $x \notin n(\alpha)$ and if $\mu = \overline{a}(x)$ then $x \notin \mathsf{fn}(P_1)$ requirement in the rules for propagating $\overline{a}(x)[n]$ past past actions and parallel composition, and the last case is prevented by requiring alternative paths to be standard if we want to propagate an action past the choice.

*Proof (Proof of Proposition 3.1).*

1. We perform induction on $P \xrightarrow{\mu[n]} Q$:
   (a) Suppose $P = a(x).P'$, $\mu = a(b)$, $\mathsf{std}(P')$, $Q = a(b)[n].Q'$, and $Q' = Q[x := b_{[n]}]$. Then, since $x \notin n(Q')$, $Q \overset{a(b)}{\rightsquigarrow} P$.
   (b) Suppose $P = \overline{a}(x).P'$, $\mu = \overline{a}(x)$, $\mathsf{std}(P')$, $Q = \overline{a}(x)[n].P'$. Then clearly $Q \overset{\overline{a}(x)}{\rightsquigarrow} P$.
   (c) Suppose $P = \alpha[m].P'$, $P' \xrightarrow{\mu[n]} Q'$, $Q = \alpha[m].Q'$, $n \neq m$, and if $\mu = \overline{a}(x)$ then $x \notin n(\alpha)$. Then by induction $Q' \underset{P}{\overset{\mu[n]}{\rightsquigarrow}}\,'$, and clearly $Q \overset{\mu[n]}{\rightsquigarrow} P$.
   (d) Suppose $P = P_0|P_1$, $P_0 \xrightarrow{\mu[n]} Q_0$, $\mathsf{fsh}[n](P_1)$, $Q = Q_0|P_1$, and if $\mu = \overline{a}(x)$ then $x \notin \mathtt{fn}(P_1)$. Then by induction, $Q_0 \overset{\mu[n]}{\rightsquigarrow}$, and obviously $Q \overset{\mu[n]}{\rightsquigarrow} P$.
   (e) Suppose $P = P_0|P_1$, $P_0 \xrightarrow{a(x)[n]} Q_0$, $P_1 \xrightarrow{\overline{a}(x)[n]} Q_1$, $\mu = \tau$, and $Q = (\nu x)(Q_0|Q_1)$. Then by induction $Q_0 \overset{a(x)}{\rightsquigarrow} P_0$ and $Q_1 \overset{\overline{a}(x)}{\rightsquigarrow} P_1$, meaning clearly $Q \overset{\mu[n]}{\rightsquigarrow} P$.
   (f) Suppose $P = P_0 + P_1$, $P_0 \xrightarrow{\mu[n]} Q_0$, $\mathsf{std}(P_1)$, and $Q = Q_0 + P_1$. Then by induction $Q_0 \overset{\mu[n]}{\rightsquigarrow} P_0$, meaning $Q \overset{\mu[n]}{\rightsquigarrow} P$.
   (g) Suppose $P = (\nu x)P'$, $P' \xrightarrow{\mu[n]} Q'$, $x \notin n(\mu)$, and $Q = (\nu x)Q'$. Then by induction $Q' \overset{\mu[n]}{\rightsquigarrow} P'$, and we get $Q \overset{\mu[n]}{\rightsquigarrow} P$.
   (h) Suppose $P \equiv P'$, $P' \xrightarrow{\mu[n]} Q'$, and $Q \equiv Q'$. Then by induction $Q' \overset{\mu[n]}{\rightsquigarrow} P'$, and therefore $Q \overset{\mu[n]}{\rightsquigarrow} P$.
2. We prove this by induction on $P \overset{\mu[n]}{\rightsquigarrow} Q$:
   (a) Suppose $P = a(b)[n].P'$, $\mu = a(b)$, $\mathsf{std}(P')$, $x \notin n(P')$, $Q' = P'[b_{[n]} := x]$, and $Q = a(x).Q'$. Then clearly $Q \xrightarrow{\mu[n]} P$.
   (b) Suppose $P = \overline{a}(x)[n].P'$, $\mu = \overline{a}(x)$, $\mathsf{std}(P')$, $Q = \overline{a}(x).P'$. Then clearly $Q \xrightarrow{\overline{a}(x)} P$.
   (c) Suppose $P = \alpha[m].P'$, $P' \overset{\mu[n]}{\rightsquigarrow} Q'$, $m \neq n$, and $Q = \alpha[n].Q'$. Then by induction, $Q' \xrightarrow{\mu[n]} P'$, and since $P$ is forwards reachable, if $\mu = \overline{a}(x)$ then $x \notin n(\alpha)$. This means $Q \xrightarrow{\mu[n]} P$.
   (d) Suppose $P = P_0|P_1$, $P_0 \overset{\mu[n]}{\rightsquigarrow} Q_0$, $\mathsf{fsh}[n](P_1)$, $Q = Q_0|P_1$, and if $\mu = \overline{a}(()x)$ then $x \notin \mathtt{fn}(P_1)$. Then by induction $Q_0 \xrightarrow{\mu[n]} P_0$, and clearly $Q \xrightarrow{\mu[n]} P$.
   (e) Suppose $P = (\nu x)(P_0|P_1)$, $\mu = \tau$, $P_0 \overset{a(x)[n]}{\rightsquigarrow} Q_0$, $P_1 \overset{\overline{a}(x)[n]}{\rightsquigarrow} Q_1$, and $Q = Q_0|Q_1$. Then by induction $Q_0 \xrightarrow{a(x)} P_0$ and $Q_1 \xrightarrow{\overline{a}(x)} P_1$, meaning clearly $Q \xrightarrow{\mu[n]} P$.

(f) Suppose $P = P_0 + P_1$, $P_0 \overset{\mu[n]}{\rightsquigarrow} Q_0$, $\mathsf{std}(P_1)$, and $Q = Q_0 + P_1$. Then by induction $Q_0 \xrightarrow{\mu[n]} P_0$, meaning $Q \xrightarrow{\mu[n]} P$.

(g) Suppose $P = (\nu x)P'$, $P' \overset{\mu[n]}{\rightsquigarrow} Q'$, $x \notin n(\mu)$, and $Q = (\nu x)Q'$. Then by induction $Q' \xrightarrow{\mu[n]} P'$, and we get $Q \xrightarrow{\mu[n]} P$.

(h) Suppose $P \equiv P'$, $P' \overset{\mu[n]}{\rightsquigarrow} Q'$, and $Q \equiv Q'$. Then by induction $Q' \xrightarrow{\mu[n]} P'$, and therefore $Q \xrightarrow{\mu[n]} P$.

**Proposition B.2 (Reverse diamond).** *Given forwards reachable processes $P$, $Q$, and $R$, if $P \overset{\mu[m]}{\rightsquigarrow} Q$ and $P \overset{\mu'[n]}{\rightsquigarrow} R$ and $m \neq n$, then there exists a process $S$ such that $Q \overset{\mu'[n]}{\rightsquigarrow} S$ and $R \overset{\mu[m]}{\rightsquigarrow} S$.*

*Proof (Proof of Proposition B.2).* We use structural induction on $P$ to prove both these at once:

1. Suppose $P = 0$ or $P = \alpha.P'$. Then $P$ cannot do any backwards transitions.
2. Suppose $P = \alpha[o].P'$. Then either $\mathsf{std}(P')$ and $n = m = o$, or $Q = a(b)[o].Q'$, $R = a(b)[o].R'$, $P' \overset{\mu[m]}{\rightsquigarrow} Q'$, and $P' \overset{\mu'[n]}{\rightsquigarrow} R'$, meaning by induction there exists $S'$ such that $Q' \overset{\mu'[n]}{\rightsquigarrow} S'$ and $R' \overset{\mu[m]}{\rightsquigarrow} S'$. We say that $S = \alpha[n].S'$, and the theorem holds.
3. Suppose $P = P_0 + P_1$, then either $\mathsf{std}(P_0)$, $P_1 \overset{\mu[m]}{\rightsquigarrow} Q_1$, $P \overset{\mu'[n]}{\rightsquigarrow} R_1$, $Q = P_0 + Q_1$, and $R = P_0 + R_1$, or $\mathsf{std}(P_1)$, $P_0 \overset{\mu[m]}{\rightsquigarrow} Q_0$, $P \overset{\mu'[n]}{\rightsquigarrow} R_0$, $Q = Q_0 + P_1$, and $R = R_0 + P_1$. In the first case, by induction there exists an $S_1$ such that $Q_1 \overset{\mu'[n]}{\rightsquigarrow} S_1$ and $R_1 \overset{\mu[m]}{\rightsquigarrow} S_1$, and we define $S = P_0 + S_1$, and theorem holds. The second case is similar.
4. Suppose $P = (\nu x)P'$. Then either (1) $P' \overset{\mu[m]}{\rightsquigarrow} Q'$ and $x \notin n(\mu)$ and $Q = (\nu x)Q'$ or (2) $P' = P_0|P_1$, $P_i \overset{a(x)[m]}{\rightsquigarrow} Q_i$, $P_{1-i} \overset{\overline{a}(x)[m]}{\rightsquigarrow} Q_{i-1}$, $\mu = \tau$, and $Q = Q_0|Q_1$, and either (a) $P' \overset{\mu'[n]}{\rightsquigarrow} R'$ and $x \notin n(\mu')$ and $R = (\nu x)R'$ or (b) $P' = P_0|P_1$, $P_i \overset{a(x)[n]}{\rightsquigarrow} R_i$, $P_{1-i} \overset{\overline{a}(x)[n]}{\rightsquigarrow} R_{i-1}$, $\mu' = \tau$, and $R = R_0|R_1$.

   In case 1a, by induction there exists $S'$ such that $Q' \overset{\mu'[n]}{\rightsquigarrow} S'$ and $R' \overset{\mu[m]}{\rightsquigarrow} S'$, and we define $S = (\nu x)S'$, and the theorem holds.

   In case 1b, there exists $P_j$ such that $P_j \overset{\mu[m]}{\rightsquigarrow} Q_j$, and $\mathsf{fsh}[m](P_{1-j})$, and if $\mu = \overline{a}(x)$ then $x \notin \mathsf{fn}(P_1)$. If $j = i$ then by induction there exists an $S_i$ such that $Q_j \overset{\mu'[n]}{\rightsquigarrow} S_i$ and $R_i \overset{a(x)[m]}{\rightsquigarrow} S_i$, and we define $S = S_i|R_{1-i}$, and the theorem holds. If $I = 1 - j$, the argument is similar.

Case 2a is similar to case 1b.
Case 2b cannot occur because we cannot have more than one past action outputting the same name according to Lemma B.1.

5. Suppose $P = P_0|P_1$. Then there exists an $i$ such that either $P_i \xrightarrow{\mu[m]} Q_i$ and $P_i \xrightarrow{\mu'[n]} R_i$ and $Q = Q_i|P_{1-i}$ and $R = R_i|P_{1-i}$, or $P_i \xrightarrow{\mu[m]} Q_i$ and $P_{1-i} \xrightarrow{\mu'[n]} R_{1-i}$ and $Q = Q_i|P_{1-i}$ and $R = P_i|R_{1-i}$.

   In the first case, there exists $S_i$ such that $Q_i \xrightarrow{\mu'[n]} S_i$ and $R_i \xrightarrow{\mu[m]} S_i$, and we define $S = S_i|P_{1-i}$ and the theorem holds.
   If the second case we define $S = Q_i|R_{1-i}$, and the theorem holds.

**Proposition B.3.** *Given forwards reachable processes $P$, $Q$, and $R$, if $P \xrightarrow{\mu[m]} Q$ and $P \xrightarrow{\mu'[m]} R$ then $\mu = \mu'$ and $R \equiv Q$.*

*Proof.* We prove this by structural induction:

1. Suppose $P = 0$ or $P = \alpha.P'$. Then $P$ cannot do any reverse transitions.
2. Suppose $P = \alpha[n].P'$. Then either $\mathsf{std}(P')$, meaning $\mu = \mu' = \alpha$, $n = m$, and $Q \equiv R$, or $P' \xrightarrow{\mu[m]} Q'$, $P' \xrightarrow{\mu'[m]} R'$, $Q = \alpha[n].Q'$, and $R = \alpha[n].R'$, and the result follows from induction.
3. Suppose $P = P_0 + P_1$. Then the result follows from induction.
4. Suppose $P = (\nu x)P'$. Then either (1) $P' \xrightarrow{\mu[m]} Q'$ and $x \notin n(\mu)$ and $Q = (\nu x)Q'$ or (2) $P' = P_0|P_1$, $P_i \xrightarrow{a(x)[m]} Q_i$, $P_{1-i} \xrightarrow{\overline{a}(x)[m]} Q_{i-1}$, $\mu = \tau$, and $Q = Q_0|Q_1$, and either (a) $P' \xrightarrow{\mu'[m]} R'$ and $x \notin n(\mu')$ and $R = (\nu x)R'$ or (b) $P' = P_0|P_1$, $P_i \xrightarrow{a(x)[m]} R_i$, $P_{1-i} \xrightarrow{\overline{a}(x)[m]} R_{i-1}$, $\mu' = \tau$, and $R = R_0|R_1$.
   In case 1a the result follows from induction.
   In case 1b $P_j$ such that $P_j \xrightarrow{\mu[m]} Q_j$, and $\mathsf{fsh}[m](P_{1-j})$, contradicting $P_{1-j} \xrightarrow{\alpha[m]} R_{1-j}$. Meaning this case cannot occur.
   Similar for case 2a.
   Case 2b follows from induction.
5. Suppose $P = P_0|P_1$. Then there exists an $i$ such that either $P_i \xrightarrow{\mu[m]} Q_i$ and $P_i \xrightarrow{\mu'[m]} R_i$ and $Q = Q_i|P_{1-i}$ and $R = R_i|P_{1-i}$, or $P_i \xrightarrow{\mu[m]} Q_i$ and $P_{1-i} \xrightarrow{\mu'[m]} R_{1-i}$ and $Q = Q_i|P_{1-i}$ and $R = P_i|R_{1-i}$.

   In the first case the result follows from induction. In the second case $P_i \xrightarrow{\mu[m]} Q_i$ requires $\mathsf{fsh}[m](P_{1-i})$, which contradicts $P_{1-i} \xrightarrow{\mu'[m]} R_{1-i}$, meaning this case cannot occur.

**Theorem B.4 (Parabola).** *Given processes $P$ and $Q$, such that $P \rightarrowtail^* Q$, there exists a process $R$ such that $P \rightsquigarrow^* R \rightarrow^* Q$.*

*Proof (Proof of Theorem B.4).* We say that $P \xrightarrow{\mu_0[m_0]} P_0 \ldots \xrightarrow{\mu_n[m_n]} P_n = Q$ and perform induction on the length of the trace, the number of pairs $\xrightarrow{\mu_i[m_i]} \overset{\mu_{i+1}[m_{i+1}]}{\rightsquigarrow}$ in the trace, and the location of the first such pair.

If no such pair exists then $R$ must exist.

Otherwise, we say that $\xrightarrow{\mu_i[m_i]} \overset{\mu_{i+1}[m_{i+1}]}{\rightsquigarrow}$ is the fist such pair in the trace. We have 2 cases, either $m_i = m_{i+1}$ or not.

If $m_i = m_{i+1}$ then by Propositions 3.1 and B.3, $P_{i-1} = P_{i+1}$, and we therefore have a trace $P \xrightarrow{\mu_0[m_0]} P_0 \ldots \xrightarrow{\mu_{i-1}[m_{i-1}]} P_{i-1} \xrightarrow{\mu_{i+2}[m_{i+2}]} \ldots \xrightarrow{\mu_n[m_n]} P_n = Q$.

If $m_i \neq m_{i+1}$ then by Proposition B.2 we have a trace $P \xrightarrow{\mu_0[m_0]} P_0 \ldots P_{i-1} \overset{\mu_{i+1}[m_{i+1}]}{\rightsquigarrow} \xrightarrow{\mu_i[m_i]} P_{i+1} \ldots \xrightarrow{\mu_n[m_n]} P_n = Q$

## C Section 4

In Lemma C.2 we demonstrate, that $S$ does indeed annotate any name, which was substituted for $x_1$, with $n$.

We also define the root of a $\pi$IK process as removing all keys from the process.

**Definition C.1 (Root).** *We say that a $\pi$IK process, $P$, has a root,* $\mathsf{rt}(P)$*, defined as:*

$$\mathsf{rt}(0) = 0 \quad \mathsf{rt}(!P) = !\mathsf{rt}(P) \quad \mathsf{rt}(P_0|P_1) = \mathsf{rt}(P_0)|\mathsf{rt}(P_1) \quad \mathsf{rt}(\alpha.P) = \alpha.\mathsf{rt}(P)$$
$$\mathsf{rt}(P_0 + P_1) = \mathsf{rt}(P_0) + \mathsf{rt}(P_1) \quad \mathsf{rt}(\alpha[m].P) = \alpha.\mathsf{rt}(P) \quad \mathsf{rt}((\nu x)P) = (\nu x)\mathsf{rt}(P)$$

**Lemma C.2.** *Given a standard $\pi$IK process $P$, a $\pi$IK process $P'$, a series of substitutions $[x_1 := a_1][x_2 := a_2] \ldots [x_k := a_k]$, such that $\mathsf{rt}(P') \equiv P[x_1 := a_1][x_2 := a_2] \ldots [x_k := a_k]$ using the definition of $\equiv$ from Section 2, and a key $[n]$, we get that $S(P', P, [n], x_1) = P''$ for some $P''$ such that*

$$\mathsf{rt}(P'') \equiv P[x_1 := a_{1_{[n]}}][x_2 := a_2] \ldots [x_k := a_k] .$$

*Proof (Proof of Lemma C.2).* We prove this by structural induction on $P$:

– Assume $P = 0$. Then $P' = P[x_1 := a_1][x_2 := a_2] \ldots [x_k := a_k] = 0$ and $S(P[x_1 := a_1][x_2 := a_2] \ldots [x_k := a_k], P, [n], x_1) = 0$.
– Assume $P = b(c).Q$ Then either $P' = d(e).Q'$, or $P' = d(e)[m].Q'$, for some $d, e, m$. We then get 4 cases: either $b = x_1$, $c = x_1$, $b = c = x_1$, or $b \neq x_1$ and $c \neq x_1$.
  Assume $b = x_1$ and $c \neq x_1$. Then $d = a_1$, and

$$S(P', P, [n], x_1) = d[n](c).S(Q', Q, [n], x_1) ,$$

  and the result follows from induction.
  Assume $c = x_1$ and $b \neq x_1$. Then, since $c$ is bound, $P[x_1 := a_1] = P[x_1 := a_{1_{[n]}}] = P$, and the result follows.
  Assume $b = c = x_1$. Then $d = a_1$ and $Q[x_1 := a_1][x_2 := a_2] \ldots [x_k := a_k] = Q[x_2 := a_2] \ldots [x_k := a_k] = Q'$, and the result follows.

- Assume $P = \overline{b}(c).Q$. This is similar to the previous case.
- Assume $P = \sum_{i \in I} P_i$. Then the result follows trivially from induction.
- Assume $P = P_0|P_1$. Then either $P' = P'_0|P'_1$, or $P_0 \equiv\ !P_1$ and $P' = P'_0$.
  If $P' = P'_0|P'_1$ then the result follows trivially from induction.
  If $P_0 = !P_1$ and $P' = P_0$, then $P'' = S\left(!P'_0, P_0, [n], x\right) | S\left(P'_0, P_1, [n], x\right)$, and the result follows from induction.
- Assume $P = (\nu b)Q$. Then $P' = (\nu c)Q'$ and either $b = x_1$ or $b \neq x_1$.
  If $b = x_1$, then $P[x_1 := a_1] = P[x_1 := a_{1_{[n]}}] = P$.
  If $b \neq x_1$, then the result follows from induction.
- Assume $P = !Q$. Then either $P' = !Q'$, or $P' = P'_0|P'_1$.
  If $P' = !Q'$, the result follows trivially from induction.
  Otherwise the case is similar to the second case on parallel composition.

*Proof (Proof of Theorem 4.6).* We first show that if there exists a location $u$ such that $\mathbf{H} \vdash P \xrightarrow[u]{\mu} \mathbf{H}' \vdash P'$, then there exists a key $m$, such that $E((\{(a,u,u) \mid (a,u) \in \overline{H}\}, \{(a,u,u) \mid (a,u) \in \underline{H}\}, \{(a,u,u) \mid (a,u) \in H\}) \vdash P, P) \xrightarrow[E]{\mu[m]} ((\{(a,u,u) \mid (a,u) \in \overline{H'}\}, \{(a,u,u) \mid (a,u) \in \underline{H'}\}, \{(a,u,u) \mid (a,u) \in H'\}) \vdash P', P')$ by induction in the size of $\overline{H} \cup \underline{H} \cup H$ and the structure of $P$:

Assume $\mathbf{H} = (\emptyset, \emptyset, \emptyset)$. Then $E(\mathbf{H} \vdash P, P) = P$.

- Assume $P = a(x).Q$. Then $\mu = a(b)$, $u = [P][Q[x := b]]$, and $\mathbf{H}' \vdash P' = (\emptyset, \{(a(b), u)\}, \emptyset) \vdash Q[x := b]$. We then by Lemma C.2 get $E(\mathbf{H}' \vdash P', P') = a(x)\,[[P][Q[x := b]]]\,.Q[x := b_{[[P][Q[x:=b]]]}]$, and the rest of the case follows naturally.
- Assume $P = \overline{a}(x).Q$. This case is similar to the previous.
- Assume $P = P_0|P_1$. Then either $u = iu'$, or $u = \langle 0u_0, 1u_1 \rangle$.
  If $u = 0u'$, then $(\emptyset, \emptyset, \emptyset) \vdash P_0 \xrightarrow[u']{\mu} \mathbf{H}'_0 \vdash P'_0$, $\mathbf{H}' \vdash P' = (0\overline{H'}_0, 0\underline{H'}_0, 0H'_0) \vdash P'_0|P_1$, and if $\mu = \overline{a}(b)$ then $b \notin \mathsf{fn}(P_1)$. By induction, $P_0 \xrightarrow{\mu[m]} E(\mathbf{H}'_0 \vdash P'_0, P_0)$, and therefore $P_0|P_1 \xrightarrow{\mu[m]} E(\mathbf{H}'_0 \vdash P'_0, P_0)|P_1 = E((0\overline{H'}_0, 0\underline{H'}_0, 0H'_0) \vdash P'_0|P_1, P'_0|P_1)$.
  If $u = 1u'$, the case is similar to $u = 0u'$.
  If $u = \langle 0u_0, 1u_1 \rangle$, then $(\emptyset, \emptyset, \emptyset) \vdash P_i \xrightarrow[u_i]{a(b)} \mathbf{H}'_i \vdash P'_i$ and $(\emptyset, \emptyset, \emptyset) \vdash P_{1-i} \xrightarrow[u']{\overline{a}(b)} \mathbf{H}_{1-i} \vdash P'_{1-i}$ for some $i \in \{0, 1\}$ and $b \notin \mathsf{fn}(P_i)$ and $\mathbf{H}' \vdash P' = (\emptyset, \emptyset, \{(a(b), \overline{a}(b), u)\}) \vdash P'_0|P'_1$. By induction, $E((\emptyset, \emptyset, \emptyset) \vdash P_i, P_i) \xrightarrow{a(b)[m]} E(\mathbf{H}'_i \vdash P'_i, P'_i)$ and $E((\emptyset, \emptyset, \emptyset) \vdash P_{1-i}, P_{1-i}) \xrightarrow{\overline{a}(b)[m]} E(\mathbf{H}'_{1-i} \vdash P'_{1-i}, P_{i-1})$. Therefore

$$\begin{aligned} P_0|P_1 \xrightarrow{\tau[m]} & E((\emptyset, \emptyset, \{(a(b), \overline{a}(b), \langle 0u_0, 1u_1 \rangle, m)\}) \vdash (\nu b)(P'_0|P'_1), (\nu b)(P'_0|P'_1)) \\ &= (\nu b)E((\emptyset, \emptyset, \{(a(b), \overline{a}(b), \langle 0u_0, 1u_1 \rangle, m)\}) \vdash (P'_0|P'_1), (P'_0|P'_1)) \end{aligned}$$

- Assume $P = (\nu x)Q$. Then $(\emptyset, \emptyset, \emptyset) \vdash Q \xrightarrow[u]{\mu} \mathbf{H}' \vdash Q'$, $x \notin n(\mu)$, and $P' = (\nu x)Q'$. We then get by induction $Q \xrightarrow{\mu[m]} E(\mathbf{H}' \vdash Q', Q')$, and therefore $(\nu x)Q \xrightarrow{\mu[m]} (\nu x)E(\mathbf{H}' \vdash Q', Q') = E(\mathbf{H}' \vdash (\nu x)Q', (\nu x)Q')$.

- Assume $P = !Q$. Then $(\emptyset,\emptyset,\emptyset) \vdash !Q|Q \xrightarrow[u]{\mu} \mathbf{H}' \vdash P'$, and the rest follows from the parallel case.

If for any $(\mu', u') \in \overline{H} \cup \underline{H} \cup H$, if there exists a location $u$ such that $\mathbf{H} - (\mu', u') \vdash P \xrightarrow[u]{\mu} \mathbf{H}'' \vdash P'$, then there exists a key $m$, such that $E(\mathbf{H} - (\mu', u') \vdash P, P) \xrightarrow{\mu[m]} E(\mathbf{H}'' \vdash P', P')$, then $E$ only adds past actions and unused choice branches to the process, both of which one can easily propagate the action past.

We then show that if there exists a key, $m$, such that $E((\{(a,u,u) \mid (a,u) \in \overline{H}\}, \{(a,u,u) \mid (a,u) \in \underline{H}\}, \{(a,u,u) \mid (a,u) \in H\}) \vdash P, P) \overset{\mu[m]}{\rightarrowtail} P''$, then there exists a location, $u$, and a $\pi$IH process, $\mathbf{H}' \vdash P'$, such that $\mathbf{H} \vdash P \overset{\mu}{\underset{u}{\rightarrowtail}} \mathbf{H}' \vdash P'$ and $P'' \equiv E((\{(a,u,u) \mid (a,u) \in \overline{H'}\}, \{(a,u,u) \mid (a,u) \in \underline{H'}\}, \{(a,u,u) \mid (a,u) \in H'\}) \vdash P', P')$ We again do this by induction on the number of extrusions in $\overline{H} \cup \underline{H} \cup H$, and the structure of $P$.

Assume $\overline{H} \cup \underline{H} \cup H = \emptyset$. Then $E((\{(a,u,u) \mid (a,u) \in \overline{H}\}, \{(a,u,u) \mid (a,u) \in \underline{H}\}, \{(a,u,u) \mid (a,u) \in H\}) \vdash P, P) = P$. Since we are only proving operational correspondence up to structural congruence, we can discount any rules employing that.

- Assume $P = a(x).Q$. Then $\mu = a(b)$, we select $m = [[P][Q[x := b]]]$, and $E((\{(a,u,u) \mid (a,u) \in \overline{H'}\}, \{(a,u,u) \mid (a,u) \in \underline{H'}\}, \{(a,u,u) \mid (a,u) \in H'\}) \vdash P', P') = a(b)[[P][Q[x := b]]].Q[x := b_{[[P][Q[x:=b]]]}]$. We say that $u = [P][Q[x := b]]$, $\mathbf{H}' \vdash P' = (\emptyset, \{(a(b), [P][Q[x := b]], [P][Q[x := b]])\}, \emptyset) \vdash Q[x := b]$, and by Lemma C.2 the result follows.
- Assume $P = \overline{a}(b).Q$. This case is similar to the previous.
- Assume $P = P_0|P_1$. Then either $P_0 \xrightarrow{\mu[m]} P'_0$ and $E((\{(a,u,u) \mid (a,u) \in \overline{H'}\}, \{(a,u,u) \mid (a,u) \in \underline{H'}\}, \{(a,u,u) \mid (a,u) \in H'\}) \vdash P', P') = P'_0|P_1$, $P_1 \xrightarrow{\mu[m]} P'_1$ and $E((\{(a,u,u) \mid (a,u) \in \overline{H'}\}, \{(a,u,u) \mid (a,u) \in \underline{H'}\}, \{(a,u,u) \mid (a,u) \in H'\}) \vdash P', P') = P_0|P'_1$, or $P_i \xrightarrow{a(b)[m]} P'_i$, $P_{1-i} \xrightarrow{\overline{a}(b)[m]} P'_{1-i}$, $\mu = \tau$, and $E((\{(a,u,u) \mid (a,u) \in \overline{H'}\}, \{(a,u,u) \mid (a,u) \in \underline{H'}\}, \{(a,u,u) \mid (a,u) \in H'\}) \vdash P', P') = (\nu b)(P'_0|P'_1)$.

  If $P_0 \xrightarrow{\mu[m]} P'_0$ and $E((\{(a,u,u) \mid (a,u) \in \overline{H'}\}, \{(a,u,u) \mid (a,u) \in \underline{H'}\}, \{(a,u,u) \mid (a,u) \in H'\}) \vdash P', P') = P'_0|P_1$, then by induction, there exists $u_0$ and $\mathbf{H}_0 \vdash P''_0$ such that $E((\{(a,u,u) \mid (a,u) \in \overline{H'}_0\}, \{(a,u,u) \mid (a,u) \in \underline{H'}_0\}, \{(a,u,u) \mid (a,u) \in H'_0\}) \vdash P''_0, P''_0) = P'_0$ and $(\emptyset,\emptyset,\emptyset) \vdash P_0 \xrightarrow[u_0]{\mu} \mathbf{H}_0 \vdash P''_0$. We therefore get $(\emptyset,\emptyset,\emptyset) \vdash P \xrightarrow[0u_0]{\mu} (0\overline{H_0}, 0\underline{H_0}, 0H_0) \vdash P'_0|P_1$.

  If $P_1 \xrightarrow{\mu[m]} P'_1$ and $E((\{(a,u,u) \mid (a,u) \in \overline{H'}\}, \{(a,u,u) \mid (a,u) \in \underline{H'}\}, \{(a,u,u) \mid (a,u) \in H'\}) \vdash P', P') = P_0|P'_1$, then the case is similar to the previous.

  If $P_i \xrightarrow{a(b)[m]} P'_i$, $P_{1-i} \xrightarrow{\overline{a}(b)[m]} P'_{1-i}$, $\mu = \tau$, and $E((\{(a,u,u) \mid (a,u) \in \overline{H'}\}, \{(a,u,u) \mid (a,u) \in \underline{H'}\}, \{(a,u,u) \mid (a,u) \in H'\}) \vdash P', P') = (\nu b)(P'_0|P'_1)$, then by induction we have $u_i$ and $\mathbf{H}_i \vdash P''_i$, $u_{1-i}$ and $\mathbf{H}_{1-i} \vdash P''_{1-i}$ such that $E((\{(a,u,u) \mid$

$(a,u) \in \overline{H'}_0\}, \{(a,u,u) \mid (a,u) \in \underline{H'}_0\}, \{(a,u,u) \mid (a,u) \in H'_0\}) \vdash P''_0, P''_0) = P'_0$ and $E((\{(a,u,u) \mid (a,u) \in \overline{H'}_1\}, \{(a,u,u) \mid (a,u) \in \underline{H'}_1\}, \{(a,u,u) \mid (a,u) \in H'_1\}) \vdash P''_1, P''_1) = P'_1$ and $(\emptyset,\emptyset,\emptyset) \vdash P_i \xrightarrow[u_i]{a(b)} \mathbf{H}_i \vdash P''_i$ and $(\emptyset,\emptyset,\emptyset) \vdash P_{1-i} \xrightarrow[u_{1-i}]{\overline{a}(b)} \mathbf{H}_{1-i} \vdash P''_{1-i}$. We therefore say $m = \langle 0u_0, 1u_1 \rangle$, and get $(\emptyset,\emptyset,\emptyset) \vdash P_0|P_1 \xrightarrow[\langle 0u_0, 1u_1\rangle]{(} \emptyset, \emptyset, \{(a(b), \overline{a}(b), \langle 0u_0, 1u_1 \rangle, m)\}) \vdash (\nu b)(P''_0|P''_1)$.

- Assume $P = (\nu x)Q$. Then $x \notin n(\mu)$, $E((\{(a,u,u) \mid (a,u) \in \overline{H'}\}, \{(a,u,u) \mid (a,u) \in \underline{H'}\}, \{(a,u,u) \mid (a,u) \in H'\}) \vdash P', P') = (\nu x)Q'$ and $Q \xrightarrow{\mu[m]} Q'$. We therefore get $P' = (\nu x)Q''$, and by induction $\mathbf{H} \vdash Q \xrightarrow[u]{\mu} \mathbf{H}' \vdash Q'$, and therefore $\mathbf{H} \vdash P \xrightarrow[u]{\mu} \mathbf{H}' \vdash P'$.
- Assume $P = !Q$. Then the transition must involve structural congruence, $!Q|Q \xrightarrow{\mu[m]} P'''$ for $P''' \equiv P''$, and the rest follows from the parallel case.

If for any $(\mu', u') \in \overline{H} \cup \underline{H} \cup H$, if there exists a key $m$, such that $E(\mathbf{H} - (\mu', u') \vdash P, P) \xrightarrow{\mu[m]} E(\mathbf{H}'' \vdash P', P')$, then there exists a location $u$ such that $\mathbf{H} - (\mu', u') \vdash P \xrightarrow[u]{\mu} \mathbf{H}'' \vdash P'$, then having more past extrusions does not stop $\mathbf{H} - (\mu', u') \vdash P$ from performing any forwards actions and having more past actions does not allow $E(\mathbf{H} - (\mu', u') \vdash P, P)$ to perform additional forward actions.

We then need to prove that if there exists a location $u$ such that $\mathbf{H} \vdash P \overset{\mu}{\underset{u}{\rightsquigarrow}} \mathbf{H}' \vdash P'$, then there exists a key $m$, such that $E((\{(a,u,u) \mid (a,u) \in \overline{H}\}, \{(a,u,u) \mid (a,u) \in \underline{H}\}, \{(a,u,u) \mid (a,u) \in H\}) \vdash P, P) \overset{\mu[m]}{\rightsquigarrow} E((\{(a,u,u) \mid (a,u) \in \overline{H'}\}, \{(a,u,u) \mid (a,u) \in \underline{H'}\}, \{(a,u,u) \mid (a,u) \in H'\}) \vdash P', P')$.

This follows naturally from the above properties, and Propositions 3.1 and 2.4.

We finally need to prove that if there exists a key, $m$, such that $E((\{(a,u,u) \mid (a,u) \in \overline{H}\}, \{(a,u,u) \mid (a,u) \in \underline{H}\}, \{(a,u,u) \mid (a,u) \in H\}) \vdash P, P) \overset{\mu[m]}{\rightsquigarrow} P''$, then there exists a location, $u$, and a $\pi$IH process, $\mathbf{H}' \vdash P'$, such that $\mathbf{H} \vdash P \overset{\mu}{\underset{u}{\rightsquigarrow}} \mathbf{H}' \vdash P'$ and $P'' \equiv E((\{(a,u,u) \mid (a,u) \in \overline{H'}\}, \{(a,u,u) \mid (a,u) \in \underline{H'}\}, \{(a,u,u) \mid (a,u) \in H'\}) \vdash P', P')$.

As we have proven the above properties, and Propositions 3.1, and 2.4, we only need to prove that there exists a $\pi$IH process $\mathbf{H}' \vdash P'$, such that $P'' \equiv E((\{(a,u,u) \mid (a,u) \in \overline{H'}\}, \{(a,u,u) \mid (a,u) \in \underline{H'}\}, \{(a,u,u) \mid (a,u) \in H'\}) \vdash P', P')$. Since none of the transition rules - forward or reverse - in $\pi$IK can create unguarded choice from guarded choice, and $E$ only generates $\pi$I-calculus processes with guarded choice, we know $P''$ has guarded choice.

If $P''$ is a standard process, then $\mathbf{H}' \vdash P' = (\emptyset,\emptyset,\emptyset) \vdash P''$. Otherwise, by Theorems B.4 and A.7, $P''$ must be forwards reachable from a standard process $P'''$ such that $P''' \equiv E((\emptyset,\emptyset,\emptyset) \vdash P''', P''')$, and by the above properties, $\mathbf{H}' \vdash P'$ exists.

## D Section 6

**Proposition D.1 (Structural Congruence).** *Given processes $P$ and $P'$ and a set of names $\mathcal{N} \supseteq n(P) \cup n(P')$, if $P \equiv P'$, $\{\!\!\{ P \}\!\!\}_{\mathcal{N}} = \langle \mathcal{E}, \mathsf{Init}, k \rangle$, and $\{\!\!\{ P' \}\!\!\}_{\mathcal{N}} = \left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$, then there exists an isomorphism $f : \mathcal{E} \to \mathcal{E}'$ such that $f(\mathsf{Init}) = \mathsf{Init}'$ and for all $e \in \mathsf{Init}$, $k(e) = k'(f(e))$.*

*Proof.* We say that $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$ and $\mathcal{E}' = (E', F', \mapsto', \sharp', \rhd', \lambda', \mathsf{Act}')$ and do a case analysis on the Structural congruence rules:

**$P = P_0|P_1$ and $P' = P_1|P_0$:** Products are unique up to isomorphism, and

$$f(e) = \begin{cases} (e_1, e_0) & \text{if } e = (e_0, e_1) \\ (e_1, *) & \text{if } e = (*, e_1) \\ (*, e_0) & \text{if } e = (e_0, *) \end{cases}$$

clearly fulfils the conditions other conditions and remains a morphism after the enablings and preventions describing the link dependencies are added to the product.

**$P = P_0|(P_1|P_2)$ and $P' = (P_0|P_1)|P_2$:** Products are associative up to isomorphism, and $f((e_0, (e_1, e_2)) = ((e_0, e_1), e_2)$ clearly fulfils the other conditions and remains a morphism after the enablings and preventions describing the link dependencies are added to the product.

**$P = P'|0$:** If $f((e, *)) = e$, then this clearly holds.

**$P = P_0 + P_1$ and $P' = P_1 + P_0$:** Coproducts are unique up to isomorphism, and $f(i, e) = (1 - i, e)$ clearly fulfils the other conditions.

**$P = P_0 + (P_1 + P_2)$ and $P' = (P_0 + P_1) + P_2$:** Coproducts are associative up to isomorphism, and $f((e_0, (e_1, e_2))) = ((e_0, e_1), e_2)$ clearly fulfils the other conditions.

**$P = P' + 0$:** Clearly $f(0, e) = e$ is an isomorphism, $\mathsf{Init} = \{0\} \times \mathsf{Init}'$, and $k(0, e) = k'(e)$.

**$P = !Q$ and $P' = !Q|Q$:** Obvious.

*Proof (Proof of Theorem 6.3).* Let $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$ and $\mathcal{E}' = (E', F', \mapsto', \sharp'$ $, \rhd', \lambda', \mathsf{Act}')$. We prove the theorem by induction on $P \xrightarrow{\mu[m]} P'$:

1. Suppose $P = a(x).Q$, $P' = a(x)[m].Q[x := b_{[n]}]$, $\mathsf{std}(Q)$, and $\mu = a(b)$. Then for all $n \in (\mathcal{N} \setminus \mathsf{sbn}(Q)) = (\mathcal{N} \setminus \mathsf{sbn}(Q[x := b_{[n]}]))$, we have $\{\!\!\{ Q[x := n] \}\!\!\} = \langle \mathcal{E}_n, \mathsf{Init}_n, k_n \rangle$, $\left\{\!\!\{ Q[x := b_{[n]}][b_{[n]} := n] \right\}\!\!\} = \left\langle \mathcal{E}'_n, \mathsf{Init}'_n, k'_n \right\rangle$, and an isomorphism $f_n : \mathcal{E}_n \to \mathcal{E}'_n$. We define our isomorphism

   $$f((n, e_n)) = \begin{cases} (n, f_n(e_n)) & \text{if } e_n \in E_n \\ (n, e'_n) & \text{for } \{e'_n\} = \{e' \mid (n, e') \in E' \text{ and } e' \notin E'_n\} \text{ otherwise} \end{cases}$$

   Since all bound names are different from all other bound and free names, $b \notin \mathsf{bn}(Q)$, and therefore there exists an $e \in E$ such that $\lambda(e) = a(b)$, and for all $e' \in E$ either $e' = e$, $e' \sharp e$, or $\{e\} \mapsto e'$. We therefore get $\mathsf{Init} = \emptyset \xrightarrow{\{e\}} X$ and $f(X) = \mathsf{Init}'$, and the rest of the conditions fulfilled.

2. Suppose $P = \overline{a}(x).Q$, $P' = \overline{a}(x)[m].Q$, $\mu = \overline{a}(x)$, and $\mathsf{std}(Q)$. This case is similar to the previous, without the choice of substitutions.
3. Suppose $P = \alpha[n].Q$, $P' = \alpha[n].Q'$, $Q \xrightarrow{\mu[m]} Q'$, $m \neq n$, and if $\mu = \overline{a}(x)$ then $x \notin n(\alpha)$. Then let $\{\![Q]\!\} = \left\langle \mathcal{E}_Q, \mathsf{Init}_Q, k_Q \right\rangle$ and $\{\![Q']\!\} = \left\langle \mathcal{E}'_Q, \mathsf{Init}'_Q, k'_Q \right\rangle$. We have an isomorphism $f_Q : \mathcal{E}_Q \to \mathcal{E}'_Q$ and a transition $\mathsf{Init}_Q \xrightarrow{e_Q} X_Q$ such that $\lambda_Q(e_Q) = \mu$, $f \circ k'_Q = k_Q[e_Q \mapsto m]$, and $f(X_Q) = \mathsf{Init}'_Q$. We define our isomorphism

   $$f((n, e_n)) = \begin{cases} (n, f_n(e_n)) & \text{if } e_n \in E_n \\ (n, e'_n) & \text{for } \{e'_n\} = \{e' \mid (n, e') \in E' \text{ and } e' \notin E'_n\} \text{ otherwise} \end{cases}$$

   and $e = (x, e_Q)$ if $\alpha = a(x)$, and

   $$f(e') = \begin{cases} f_Q(e') & \text{if } e' \in E_Q \\ e'' & \text{for } \{e''\} = \{e''' \mid e''' \in E' \text{ and } e''' \notin E'_Q\} \text{ otherwise} \end{cases}$$

   and $e = e_Q$ if $\alpha = \overline{a}(x)$. These clearly fulfil the conditions.
4. Suppose $P = P_0|P_1$, $P' = P'_0|P_1$, $P_0 \xrightarrow{\mu[m]} P'_0$, $\mathsf{fsh}[n](P_1)$, and if $\mu = \overline{a}(x)$ then $x \notin \mathsf{fn}(P_1)$. Then let $\{\![P_0]\!\} = \langle \mathcal{E}_0, \mathsf{Init}_0, k_0 \rangle$, $\left\{\!\!\left[P'_0\right]\!\!\right\} = \left\langle \mathcal{E}'_0, \mathsf{Init}'_0, k'_0 \right\rangle$, and $\{\![P_1]\!\} = \langle \mathcal{E}_1, \mathsf{Init}_1, k_1 \rangle$. We then have an isomorphism $f_0 : \mathcal{E}_0 \to \mathcal{E}'_0$ and transition $\mathsf{Init}_0 \xrightarrow{e_0} X_0$ such that $\lambda_0(e_0) = \mu$, $f_0 \circ k'_0 = k_0[e_0 \mapsto m]$, and $f_0(X_0) = \mathsf{Init}'_0$. We define our isomorphism

   $$f(e') = \begin{cases} (f_0(e'_0), *) & \text{if } e' = (e'_0, *) \\ (*, e'_1) & \text{if } e' = (*, e'_1) \\ (f_0(e'_0), e'_1) & \text{if } e' = (e'_0, e'_1) \end{cases}$$

   and $e = (e_0, *)$. Since $\mathsf{sbn}(P_0) = \mathsf{sbn}(P'_0)$ this is an isomorphism, and since all free and bound names are different, $\mathsf{no}(\mu) \cap \mathsf{sbn}(P_1) = \emptyset$, implying $\mathsf{Init} \xrightarrow{e}$. The other conditions are clearly fulfilled.
5. Suppose $P = P_0|P_1$, $P' = (\nu x)(P'_0|P_1)$, $\mu = \tau$, $P_0 \xrightarrow{a(x)[m]} P'_0$, and $P_1 \xrightarrow{\overline{a}(x)[m]} P'_1$. Then let $\{\![P_0]\!\} = \langle \mathcal{E}_0, \mathsf{Init}_0, k_0 \rangle$, $\left\{\!\!\left[P'_0\right]\!\!\right\} = \left\langle \mathcal{E}'_0, \mathsf{Init}'_0, k'_0 \right\rangle$, $\{\![P_1]\!\} = \langle \mathcal{E}_1, \mathsf{Init}_1, k_1 \rangle$, and $\left\{\!\!\left[P'_1\right]\!\!\right\} = \left\langle \mathcal{E}'_1, \mathsf{Init}'_1, k'_1 \right\rangle$. Then we have isomorphisms $f_0 : \mathcal{E}_0 \to \mathcal{E}'_0$ and $f_1 : \mathcal{E}_1 \to \mathcal{E}'_1$ and transitions $\mathsf{Init}_0 \xrightarrow{e_0} X_0$ and $\mathsf{Init}_1 \xrightarrow{e_1} X_1$ such that $\lambda_0(e_0) = a(x)$, $\lambda_1(e_1) = \overline{a}(x)$, $f_0 \circ k'_0 = k_0[e_0 \mapsto m]$, $f_1 \circ k'_1 = k_1[e_1 \mapsto m]$, $f_0(X_0) = \mathsf{Init}'_0$, and $f_1(X_1) = \mathsf{Init}'_1$. We then define our isomorphism

   $$f(e') = \begin{cases} (f_0(e'_0), *) & \text{if } e' = (e'_0, *) \\ (*, f_1(e'_1)) & \text{if } e' = (*, e'_1) \\ (f_0(e'_0), f_1(e'_1)) & \text{if } e' = (e'_0, e'_1) \end{cases}$$

   and $e = (e_0, e_1)$. Since $\mathsf{sbn}(P_0) = \mathsf{sbn}(P'_0)$ and the existence of $(f_0(e_0), f_1(e_1)) \in \mathsf{Init}'$ and $a(x)[m]$ and $\overline{a}(x)[m]$ prevents $(\nu x)$ from affecting $\mathcal{E}'$, $f$ is an isomorphism, and since $\mathsf{no}(\tau) = \emptyset$, we have a transition $\mathsf{Init} \xrightarrow{e}$. The other conditions are clearly fulfilled.

6. Suppose $P = P_0 + P_1$, $P' = P_0' + P_1$, $P_0 \xrightarrow{\mu[m]} P_0'$, and $\mathsf{std}(P_1)$. Then let $\{\!|P_0|\!\} = \langle \mathcal{E}_0, \mathsf{Init}_0, k_0 \rangle$, $\{\!|P_0'|\!\} = \langle \mathcal{E}_0', \mathsf{Init}_0', k_0' \rangle$, and $\{\!|P_1|\!\} = \langle \mathcal{E}_1, \mathsf{Init}_1, k_1 \rangle$. We then have an isomorphism $f_0 : \mathcal{E}_0 \to \mathcal{E}_0'$ and transition $\mathsf{Init}_0 \xrightarrow{e_0} X_0$ such that $\lambda_0(e_0) = \mu$, $f_0 \circ k_0' = k_0[e_0 \mapsto m]$, and $f_0(X_0) = \mathsf{Init}_0'$. We define out isomorphism
$$f((i, e_i)) = \begin{cases} (0, f_0(e_0)) & \text{if i=0} \\ (1, e_1) & \text{if i=1} \end{cases}$$
and $e = (0, e_0)$. Isomorphism is preserved by the coproduct, and the remaining conditions are clearly fulfilled.
7. Suppose $P = (\nu x)Q$, $P' = (\nu x)Q'$, $Q \xrightarrow{\mu[m]} Q'$, and $x \notin n(\mu)$. Then let $\{\!|Q|\!\} = \langle \mathcal{E}_Q, \mathsf{Init}_Q, k_Q \rangle$ and $\{\!|Q'|\!\} = \left\langle \mathcal{E}_Q', \mathsf{Init}_Q', k_Q' \right\rangle$. We have an isomorphism $f_Q$ : $\mathcal{E}_Q \to \mathcal{E}_Q'$ and a transition $\mathsf{Init}_Q \xrightarrow{e_Q} X_Q$ such that $\lambda_Q(e_Q) = \mu$, $f \circ k_Q' = k_Q[e_Q \mapsto m]$, and $f(X_Q) = \mathsf{Init}_Q'$. Either there exist past actions $b(a)[m]$ and $\overline{b}(a)[m]$ in $P$ which are not guarded by a restriction $(\nu a)$ in $P$ or not. If such $b(a)[m]$ and $\overline{b}(a)[m]$ exist, then $\langle \mathcal{E}, \mathsf{Init}, k \rangle = \langle \mathcal{E}_Q, \mathsf{Init}_Q, k_Q \rangle$ and $\langle \mathcal{E}', \mathsf{Init}', k' \rangle = \left\langle \mathcal{E}_Q', \mathsf{Init}_Q', k_Q' \right\rangle$, and the rest follows trivially. Otherwise restriction preserves morphisms, and clearly does not affect $e = e_Q$.

*Proof (Proof of Theorem 6.4).* We prove this by structural induction on $P$:

- Suppose $P = 0$. Then $E = \emptyset$, and no transition $\mathsf{Init} \xrightarrow{\{e\}} X$ exists.
- Suppose $P = \overline{a}(x).Q$. Let $\{\!|Q|\!\}_{\mathcal{N}} = \langle \mathcal{E}_Q, \mathsf{Init}_Q, k_Q \rangle$, $\mathcal{E}_Q = (E_Q, F_Q, \mapsto_Q, \sharp_Q, \rhd_Q, \lambda_Q, \mathsf{Act}_Q)$, and $\mathcal{E}' = (E', F', \mapsto', \sharp', \rhd', \lambda', \mathsf{Act}')$. Then there exists $e$ such that $E \setminus E_Q = \{e\}$, and for all $e' \in E$, if $e' \neq e$ then $\{e\} \mapsto e'$. Therefore this is the only possible $e$ such that $\mathsf{Init} \xrightarrow{\{e\}}$. Additionally we have $\lambda(e) = \overline{a}(x)$ and $P \xrightarrow{\overline{a}(x)[m]} \overline{a}(x)[m].Q$ for any key $m$, and the rest of the case is straightforward.
- Suppose $P = a(x).Q$. Then there must exist $b \in \mathcal{N} \setminus \mathsf{sbn}(P)$ such that $\lambda(e) = a(b)$, and for all $e' \in E$ either $e = e'$, $e \,\sharp\, e'$, or $\{e\} \mapsto e'$. There then exists a transition $P \xrightarrow{a(b)[m]} a(b)[m].Q[x := b_{[m]}]$ and the rest of the case is straightforward.
- Suppose $P = \overline{a}(x)[n].Q$. Let $\{\!|Q|\!\}_{\mathcal{N}} = \langle \mathcal{E}_Q, \mathsf{Init}_Q, k_Q \rangle$, $\mathcal{E}_Q = (E_Q, F_Q, \mapsto_Q, \sharp_Q, \rhd_Q, \lambda_Q, \mathsf{Act}_Q)$, and $\mathcal{E}' = (E', F', \mapsto', \sharp', \rhd', \lambda', \mathsf{Act}')$. Then $\mathsf{Init} \xrightarrow{\{e\}} X$ implies $\mathsf{Init}_Q \xrightarrow{\{e\}} X \cap E_Q$. We therefore have a transition $Q \xrightarrow{\mu[m]} Q'$ such that $\{\!|Q'|\!\}_{\mathcal{N}} = \left\langle \mathcal{E}_Q', \mathsf{Init}_Q', k_Q' \right\rangle$ and an isomorphism $f_Q : \mathcal{E}_Q \to \mathcal{E}_Q'$ such that $\lambda_Q(e) = \mu$, $f_Q \circ k_Q' = k_Q[e \mapsto m]$, and $f_Q(X \cap E_Q) = \mathsf{Init}_Q'$. This gives us a transition $P \xrightarrow{\mu[m]} \overline{a}(x)[n].Q'$ and the rest of the case is straightforward.
- Suppose $P = a(x)[n].Q$. This case is a combination of the previous two.

- Suppose $P = Q + R$ and let $\{\!|Q|\!\}_{\mathcal{N}} = \left\langle \mathcal{E}_Q, \mathsf{Init}_Q, k_Q \right\rangle$, $\mathcal{E}_Q = (E_Q, F_Q, \mapsto_Q, \sharp_Q, \rhd_Q, \lambda_Q, \mathsf{Act}_Q)$, $\{\!|R|\!\}_{\mathcal{N}} = \langle \mathcal{E}_R, \mathsf{Init}_R, k_R \rangle$, $\mathcal{E}_R = (E_R, F_R, \mapsto_R, \sharp_R, \rhd_R, \lambda_R, \mathsf{Act}_R)$, and $\mathcal{E}' = (E', F', \mapsto', \sharp', \rhd', \lambda', \mathsf{Act}')$. Either $e = (0, e_Q)$ or $e = (1, e_R)$. In the first case we get a transition transition $Q \xrightarrow{\mu[m]} Q'$ such that $\{\!|Q'|\!\}_{\mathcal{N}} = \left\langle \mathcal{E}'_Q, \mathsf{Init}'_Q, k'_Q \right\rangle$ and an isomorphism $f_Q : \mathcal{E}_Q \to \mathcal{E}'_Q$ such that $\lambda_Q(e_Q) = \mu$, $f_Q \circ k'_Q = k_Q[e_Q \mapsto m]$, and $f_Q(\{e'_Q | (0, e'_Q) \in X\}) = \mathsf{Init}'_Q$. We therefore define

$$f(e) = \begin{cases} (0, f_Q(e_Q)) & \text{if } e = (0, e_Q) \\ e & \text{otherwise} \end{cases}$$

  and the rest of the case is straightforward. If $e = (1, e_R)$, the proof is similar.
- Suppose $P = Q|R$ and let $\{\!|Q|\!\}_{\mathcal{N}} = \left\langle \mathcal{E}_Q, \mathsf{Init}_Q, k_Q \right\rangle$, $\mathcal{E}_Q = (E_Q, F_Q, \mapsto_Q, \sharp_Q, \rhd_Q, \lambda_Q, \mathsf{Act}_Q)$, $\{\!|R|\!\}_{\mathcal{N}} = \langle \mathcal{E}_R, \mathsf{Init}_R, k_R \rangle$, $\mathcal{E}_R = (E_R, F_R, \mapsto_R, \sharp_R, \rhd_R, \lambda_R, \mathsf{Act}_R)$, and $\mathcal{E}' = (E', F', \mapsto', \sharp', \rhd', \lambda', \mathsf{Act}')$. Either $e = (e_Q, *)$, $e = (*, e_R)$, or $e = (e_Q, e_R)$. If $e = (e_Q, *)$ then we have a transition $Q \xrightarrow{\mu[m]} Q'$ such that $\{\!|Q'|\!\}_{\mathcal{N}} = \left\langle \mathcal{E}'_Q, \mathsf{Init}'_Q, k'_Q \right\rangle$ and an isomorphism $f_Q : \mathcal{E}_Q \to \mathcal{E}'_Q$ such that $\lambda_Q(e_Q) = \mu$, $f_Q \circ k'_Q = k_Q[e_Q \mapsto m]$, and $f_Q(\{e'_Q | (e'_Q, *) \in X \text{ or } (e'_Q, e'_R) \in X\}) = \mathsf{Init}'_Q$. We therefore get $P \xrightarrow{\mu[m]} Q'|R$ so long as $\mathsf{fsh}[m](R)$, and if not we can do the same with a different $m$. We can then define

$$f(e') = \begin{cases} (f_0(e'_0), *) & \text{if } e' = (e'_0, *) \\ (*, e'_1) & \text{if } e' = (*, e'_1) \\ (f_0(e'_0), e'_1) & \text{if } e' = (e'_0, e'_1) \end{cases}$$

  and the rest of the case is straightforward. If $e = (*, e_R)$, the case is similar. If $e = (e_Q, e_R)$, then we have transition $Q \xrightarrow{\alpha[m]} Q'$ such that $\{\!|Q'|\!\}_{\mathcal{N}} = \left\langle \mathcal{E}'_Q, \mathsf{Init}'_Q, k'_Q \right\rangle$ and isomorphism $f_Q : \mathcal{E}_Q \to \mathcal{E}'_Q$ such that $\lambda_Q(e_Q) = \alpha$, $f_Q \circ k'_Q = k_Q[e_Q \mapsto m]$, and $f_Q(\{e'_Q | (e'_Q, *) \in X \text{ or } (e'_Q, e'_R) \in X\}) = \mathsf{Init}'_Q$, and transition $R \xrightarrow{\alpha'[m]} R'$ such that $\{\!|R'|\!\}_{\mathcal{N}} = \left\langle \mathcal{E}'_R, \mathsf{Init}'_R, k'_R \right\rangle$ and isomorphism $f_R : \mathcal{E}_R \to \mathcal{E}'_R$ such that $\lambda_R(e_R) = \alpha'$, $f_R \circ k'_R = k_R[e_R \mapsto m]$, and $f_R(\{e'_R | (*, e'_R) \in X \text{ or } (e'_Q, e'_R) \in X\}) = \mathsf{Init}'_R$ and there exist names $a, b$ such that either $\alpha = a(b)$ and $\alpha' = \overline{a}(b)$ or $\alpha' = a(b)$ and $\alpha = \overline{a}(b)$. We therefore get a transition $P \xrightarrow{\tau[m]} (\nu b)(Q'|R')$ and define

$$f(e) = \begin{cases} (f_0(e'_0), *) & \text{if } e = (e'_0, *) \\ (*, f_1(e'_1)) & \text{if } e = (*, e'_1) \\ (f_0(e'_0), f_1(e'_1)) & \text{if } e = (e'_0, e'_1) \end{cases}$$

  and the rest of the case is straightforward.
- Suppose $P = (\nu a)Q$. Let $\{\!|Q|\!\}_{\mathcal{N}} = \left\langle \mathcal{E}_Q, \mathsf{Init}_Q, k_Q \right\rangle$, $\mathcal{E}_Q = (E_Q, F_Q, \mapsto_Q, \sharp_Q, \rhd_Q, \lambda_Q, \mathsf{Act}_Q)$, and $\mathcal{E}' = (E', F', \mapsto', \sharp', \rhd', \lambda', \mathsf{Act}')$. Then either there exist past

actions $b(a)[m]$ and $\overline{b}(a)[m]$ in $Q$ which are not guarded by a restriction $(\nu a)$ in $Q$ or not. If such $b(a)[m]$ and $\overline{b}(a)[m]$ do exist, then they must be in parallel, and therefore there exists an event $e' \in E \setminus \mathsf{Init}$ such that $\lambda(e) = \overline{b}(a)$, and for all other events $e'' \in E$, if $\lambda(e')$ outputs $a$ then $e' = e$, and if $a \in \mathsf{no}(\lambda(e'))$ then $\{e'\} \mapsto e''$. Additionally there exists $e''' \in \mathsf{Init}$ such that $e''' \sharp e'$ and $\lambda(e''') = \tau$. We therefore get that $a \notin n(e)$. Additionally $\mathsf{Init}_Q = \mathsf{Init} \xrightarrow{\{e\}} X$ and by induction we have a transition $Q \xrightarrow{\mu[m]} Q'$ such that $\{\!|Q'|\!\}_{\mathcal{N}} = \left\langle \mathcal{E}'_Q, \mathsf{Init}'_Q, k'_Q \right\rangle$ and an isomorphism $f_Q : \mathcal{E}_Q \to \mathcal{E}'_Q$ such that $\lambda_Q(e) = \mu$, $f_Q \circ k'_Q = k_Q[e \mapsto m]$, and $f_Q(X) = \mathsf{Init}'_Q = \mathsf{Init}'$. We define $f = f_Q$ and the result follows. If no such $b(a)[m]$ and $\overline{b}(a)[m]$ exist in $Q$ then clearly $a \notin n(\lambda(e))$, and restriction preserves morphisms, meaning the proof is straightforward.