Highlights

Event structure semantics of (controlled) reversible CCS

Eva Graversen, Iain Phillips, Nobuko Yoshida

- Event structure semantics of uncontrolled and controlled statically reversible CCS.
- Causal reversible bundle event structure semantics of uncontrolled reversible CCS.
- Statically reversible CCS controlled by rollback.
- Non-causal extended bundle event structure semantics of reversible CCS with rollback.

Event structure semantics of (controlled) reversible CCS

Eva Graversen, Iain Phillips, Nobuko Yoshida

Imperial College London

Abstract

CCSK is a reversible form of CCS which is causal, meaning that actions can be reversed if and only if each action caused by them has already been reversed; there is no control on whether or when a computation reverses. We propose an event structure semantics for CCSK. For this purpose we define a category of reversible bundle event structures, and use the causal subcategory to model CCSK. We then modify CCSK to control the reversibility with a rollback primitive, which reverses a specific action and all actions caused by it. To define the event structure semantics of rollback, we change our reversible bundle event structures by making the conflict relation asymmetric rather than symmetric, and we exploit their capacity for non-causal reversibility.

Keywords: Reversible Computations, CCS, Event Structures, Static Reversibility, Denotational Semantics

1. Introduction

Reversible process calculi have been studied in works such as [7, 9, 11, 18, 19, 26]. One feature of such reversible processes compared to forward-only processes is their sensitivity to true concurrency distinctions [25]. For instance, using CCS notation, the processes $a|b$ and $a.b + b.a$, which are respectively a parallel composition and a choice between two orderings of events, are equivalent under interleaving semantics; however in a reversible setting we can distinguish them by noting that $a|b$ allows us to perform a followed by b and then to reverse a, which is impossible for $a.b + b.a$. This motivates us to use event structures [24] to formulate a truly concurrent semantics of a reversible process calculus.

Two reversible forms of CCS have been proposed, both using uncontrolled reversibility: RCCS [9] and CCSK [26]. RCCS creates separate memories to store past (executed) actions, what is known as *dynamic* reversibility, while CCSK annotates past actions with keys within the processes themselves, known as *static* reversibility. We formulate an event structure semantics for CCSK rather than RCCS, since the semantics for past and future actions can be defined in a similar manner, rather than having to encompass both processes and memories. We note that Lanese *et al.* [16, 22] showed that RCCS and CCSK can be encoded in one another, meaning one can use their encoding in conjunction with our event structure semantics to obtain an event structure semantics for RCCS.

Event structures are a model of true concurrency and have been used for modelling forward-only process calculi [3, 6, 31]. Describing reversible processes as event structures gives us a simple representation of the causal relationships between actions and also yields 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 rather than how the parallel actions interleave [25]. Knowing the causal relationships between actions in concurrent processes is also important when using causal-consistent debugging [20] to find bugs created by interactions between processes.

Cristescu *et al.* [8] used rigid families [4], related to event structures, to describe the semantics of Rπ [7]. However, their semantics requires 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 RCCS processes without auto-concurrency, auto-conflict, or recursion as configuration structures. By contrast, we map a CCSK process (with auto-concurrency, auto-conflict, and recursion) with past actions directly to a (reversible) event structure in a strictly denotational fashion.

Reversible forms of prime [27], asymmetric [27], and general [29] event structures have already been defined, but the usual way of handling parallel composition of forward-only prime (PES) and asymmetric event structures (AES) [30] does not translate into a reversible setting, and general event structures are far more expressive than is necessary for modelling reversible CCSK. We also considered using a reversible variant of flow event structures [3], but found that the additional expressiveness of flow event structures was unnecessary, and in fact created problems when it came to defining a category of the forward-only flow event structures [5]. We therefore chose to use bundle event structures (BESs) [21].

BESs were created with the specific purpose of allowing the same event to have multiple conflicting causes, thereby making it possible to model parallel composition without creating multiple copies of events. They do this by associating events with bundles of conflicting events, $X \mapsto e$, where in order for event e to happen one of the events of X must have already happened.

This approach can be used for modelling cases such as Example 1.1 below, where an action a has multiple options for synchronisation, either of which would allow the process to continue with the action b. If we model each synchronisation or lack thereof as a separate event then we clearly need to let b have multiple possible causes, which we can accomplish using BESs, but not using PESs. Having multiple copies of events depending on which causes we use is not possible in a reversible PES, as we do not know when performing an event what will cause it to reverse. If in Example 1.1 instead (the event labelled) b can reverse only from configurations containing either a or τ, we can have a situation where we do not have a or τ in the configuration we want to add b to, and then we do not know whether to add the b that can reverse when a is present or the b that can reverse when τ is present. In a reversible setting, bundles therefore not only simplify our event structures, but become necessary when we have events causing each other to reverse.

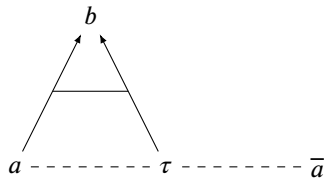


Figure 1: The event structure described in Example 1.1.

Example 1.1 (Process represented as a BES)**.** *The CCS process $a.b \mid \overline{a}$ can be described by a BES with the events $a, \tau, \overline{a}, b$, the bundle $\{a, \tau\} \mapsto b$, and the conflicts $a \sharp \tau$ and $\overline{a} \sharp \tau$ as seen in Figure 1 where a dashed line indicates conflict and connected arrows indicate a bundle. The process cannot be represented by a PES or AES without splitting some events into multiple events, due to b having multiple possible causes.*

We therefore define a category of reversible BESs (RBESs) in Section 3. Since the reversibility allowed in CCSK (as in RCCS) is *causal*, meaning that actions can be reversed if and only if every action caused by them has already been reversed, we use the causal subcategory of RBESs for defining a denotational semantics of CCSK in Section 4.

Causal reversibility has the drawback of allowing a process to get into a loop doing and undoing the same action indefinitely; there is no control on whether or when a computation reverses. We modify CCSK to control reversibility by adding the *rollback* introduced for roll-π in [17]. In Roll-CCSK every action receives a tag γ, and the process only reverses when reaching a $\mathsf{roll}\ \gamma$ primitive, upon which the action tagged with γ, together with all actions caused by it, are reversed. As in roll-π, the rollback in Roll-CCSK is *maximally permissive*, meaning that any subset of reached rollbacks may be executed, even if one of them rolls back the actions leading to another.

The operational semantics of rollback works somewhat differently in Roll-CCSK from roll-π, since roll-π has a set of memories describing past actions in addition to a π-calculus process, while CCSK has the past actions incorporated into the structure of the process, meaning that it is harder to know whether one has found all the actions necessary to reverse. Since roll-π is based on higher order-π, it can create recursion by sending processes. Roll-CCSK on the other hand has explicit recursion, and therefore needs to use bindings on tags to avoid ambiguity about which tag a roll is associated with.

As in roll-π, we also describe a more distributed semantics of rollback. This version of the semantics reverses each action marked for rollback individually, rather than performing the entire roll in one step. However, unlike roll-π, we mark all past actions in one step. We do this because propagating a marking of past actions would otherwise require the action being marked to be able to look at previous actions further back in the structure of the process to find markings it can propagate. In roll-π this is not an issue, since one has a set of parallel memories all at the same level, which can easily be compared to find out which order they are in. We describe both of these operational semantics of Roll-CCSK in Section 6.

Rollback has also been defined for μKLAIM, a tuple-based language with shared

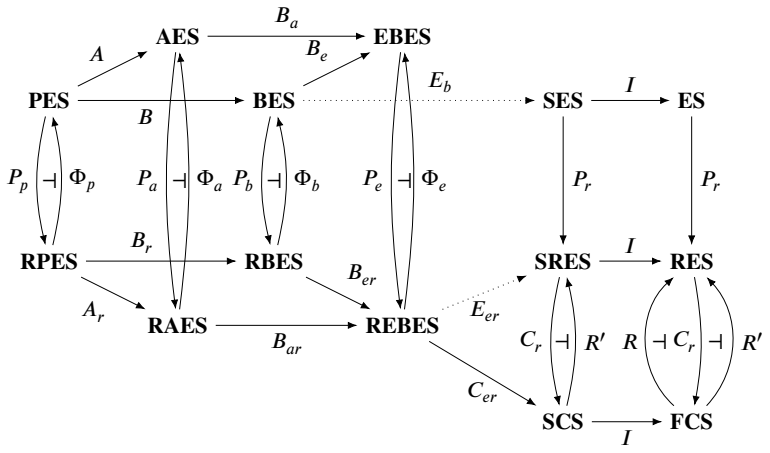


Figure 2: Event structure and configuration system categories and the functors between them. The categories **RBES** and **REBES** are new to this work, along with the functors to and from them. While BESs and EBESs are not new to this work, the morphisms in the categories **BES** and **EBES** and the functors going to and from them are. The remaining categories and functors were defined in [15].

memory [13]. This was done similarly to roll-π, giving the locations of shared memories keys, which change when a process interacts with the location. Another variant of CCS with rollback, CCS_{roll} was defined in [23]. CCS_{roll}, like roll-π, has memories in parallel with the process and equips the process with an ordering of keys, which it uses to determine which actions have been caused by the action being rolled back. This makes the semantics less compositional than Roll-CCSK. The causal-consistent reversible debugger for Erlang, CauDEr [20], allows the user to roll back not only to a checkpoint, but to other past events such as receiving or sending a specific message.

Once a $\mathsf{roll}\ \gamma$ event has happened, we need to ensure that not only are the events caused by the γ-tagged action a_γ able to reverse, but they cannot re-occur until the rollback is complete, at which point the $\mathsf{roll}\ \gamma$ event is reversed. This requires us to model *asymmetric* conflict between $\mathsf{roll}\ \gamma$ and events caused by a_γ (apart from $\mathsf{roll}\ \gamma$ itself). Asymmetric conflict is allowed in *extended* BESs (EBESs) [21]. We define a category of reversible EBESs (REBESs) in Section 5 and use them to give an event structure semantics of rollback in Section 7. Note that we do not restrict ourselves to the causal subcategory of REBESs, since reversibility in Roll-CCSK is not necessarily causal: an action a_γ tagged with γ is a cause of $\mathsf{roll}\ \gamma$, but we want a_γ to reverse before $\mathsf{roll}\ \gamma$ does.

Outline. Section 2 recalls CCSK. Section 3 describes (reversible) bundle event structures and their categories. Section 4 defines the event structure semantics of CCSK. Section 5 describes (reversible) extended bundle event structures. Section 6 defines Roll-CCSK, a version of CCSK where reversibility is controlled by rollback, and its operational semantics, and Section 7 uses reversible extended bundle event structures to describe the event structure semantics of Roll-CCSK.

Changes from conference version [14].

- We include proofs of all results.
- Sections 3 and 5 include categorical definitions of the forward-only bundle and extended bundle event structures.
- Sections 3 and 5 include functors between the newly introduced categories and their forward-only counterparts and between **RBES** and **REBES** and other categories of reversible event structures introduced in [15] creating the categories of reversible event structures and functors between them shown in Figure 2.
- Section 2 includes Proposition 4.8 showing that our ordering on event structures is a complete partial order and Lemmas 4.9, 4.10, 4.11, and 4.12 showing that the operations we define on the event structures are monotonic.
- Section 5 includes full definitions of REBES-morphisms, a functor from **REBES** to **CS**, and causal and cause-respecting REBESs. It also includes characterisations of products and coproducts of REBESs.
- We correct Definitions 6.1 and 6.2 so that they deal with keys rather than tags, as tags are not necessarily unique. As a consequence we change rollbacks to be performed on tags after they find their associated action and therefore do not need the concept of bound tags.
- We add a more distributed small-step variant of the rollback semantics to Section 6. This semantics marks all the actions needing to be reversed and afterwards reverses them individually, rather than reversing them all at once when the rollback is performed. While the marked actions are being reversed, subprocesses not preceded by marked actions can continue to perform forward actions.
- Section 7 includes intermediate results Lemmas 7.17 and 7.18.
- We have added Examples 3.5, 3.22, 3.26, 6.3, 6.7, 6.14, 7.2, and 7.5.

2. CCSK

CCSK was defined in [26], and distinguishes itself from most reversible process calculi by retaining the structure of the process when actions are performed, and annotating past actions with keys instead of generating memories. This means we get $a.P \mid \overline{a}.Q \xrightarrow{\tau[n]} a[n].P \mid \overline{a}[n].Q$, with the shared key n denoting that a and $\overline{a}$ have previously communicated, and we therefore cannot reverse one without reversing the other.

We call the set of action of CCSK $\mathcal{A}$ and let a, b, c range over $\mathcal{A}$, α, β range over $\mathcal{A} \cup \overline{\mathcal{A}}$, and μ range over $\mathcal{A} \cup \overline{\mathcal{A}} \cup \{\tau\}$. We let $\mathcal{K}$ be an infinite set of communication keys and let m, n range over $\mathcal{K}$.

CCSK then has the following syntax, very similar to CCS:

$$P ::= \alpha.P \mid \alpha[n].P \mid P_0 + P_1 \mid P_0 | P_1 \mid P \setminus A \mid P[f]$$

Here $P \setminus A$ restricts communication on actions in $A \cup \overline{A}$ and $P[f]$ applies a function $f : \mathcal{A} \to \mathcal{A}$ to the labels of transitions performed by P.

Table 1 shows the forwards rules of the operational semantics of CCSK. As CCSK is causal, the reverse rules can be derived from these. We use $\twoheadrightarrow$ to denote a reverse action, $\mathsf{std}(P)$ to denote that P is a standard process, meaning it contains no past actions, $\mathsf{keys}(P)$ to denote the set of keys used in P, and $\mathtt{fsh}[n](P)$ to denote that the key n is fresh for P. We use $\rightarrowtail$ to denote that an action may be forwards or reverse. The rules are slightly reformulated compared to [26] in that we use structural congruence $\equiv$. The rules for structural congruence are:

$$P \mid 0 \equiv P \qquad P_0 \mid P_1 \equiv P_1 \mid P_0 \qquad P_0 \mid (P_1 \mid P_2) \equiv (P_0 \mid P_1) \mid P_2$$
$$P + 0 \equiv P \qquad P_0 + P_1 \equiv P_1 + P_0 \qquad P_0 + (P_1 + P_2) \equiv (P_0 + P_1) + P_2$$

We extend CCSK with recursion as follows. We add process constants $A\left\langle \tilde{b} \right\rangle$, together with definitions $A(\tilde{a}) = P_A$, where P_A is a standard process and $\tilde{a}$ is a tuple containing the actions of P_A. This leads us to expand our definition of structural congruence with $A\left\langle \tilde{b} \right\rangle \equiv P_A\{^{\tilde{b}}/_{\tilde{a}}\}$.

Definition 2.1 (Reachability). A process P is *reachable* if there exists a standard process Q such that $Q \rightarrowtail^* P$, and *forwards-reachable* if there exists a standard process Q such that $Q \to^* P$.

Since CCSK is causal, all reachable processes are forwards-reachable ([26], Proposition 5.15; the proof still applies with recursion added).

$$\frac{\mathsf{std}(P)}{\alpha.P \xrightarrow{\alpha[n]} \alpha[n].P} \qquad \frac{P \xrightarrow{\mu[m]} P' \quad m \neq n}{\alpha[n].P \xrightarrow{\mu[m]} \alpha[n].P'} \qquad \frac{P \equiv Q \xrightarrow{\mu[n]} Q' \equiv P'}{P \xrightarrow{\mu[n]} P'}$$

$$\frac{P_0 \xrightarrow{\mu[n]} P_0' \quad \mathsf{fsh}[n](P_1)}{P_0 \mid P_1 \xrightarrow{\mu[n]} P_0' \mid P_1} \qquad \frac{P_0 \xrightarrow{\alpha[n]} P_0' \quad P_1 \xrightarrow{\overline{\alpha}[n]} P_1'}{P_0 \mid P_1 \xrightarrow{\tau[n]} P_0' \mid P_1'}$$

$$\frac{P_0 \xrightarrow{\mu[n]} P_0' \quad \mathsf{std}(P_1)}{P_0 + P_1 \xrightarrow{\mu[n]} P_0' + P_1} \qquad \frac{P \xrightarrow{\mu[n]} P' \quad \mu, \overline{\mu} \notin A}{P \setminus A \xrightarrow{\mu[n]} P' \setminus A} \qquad \frac{P \xrightarrow{\mu[n]} P'}{P[f] \xrightarrow{f(\mu)[n]} P'[f]}$$

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

$$\frac{P_0 \overset{\mu[n]}{\rightsquigarrow} P_0' \quad \mathsf{fsh}[n](P_1)}{P_0 \mid P_1 \overset{\mu[n]}{\rightsquigarrow} P_0' \mid P_1} \qquad \frac{P_0 \overset{\alpha[n]}{\rightsquigarrow} P_0' \quad P_1 \overset{\overline{\alpha}[n]}{\rightsquigarrow} P_1'}{P_0 \mid P_1 \overset{\tau[n]}{\rightsquigarrow} P_0' \mid P_1'}$$

$$\frac{P_0 \overset{\mu[n]}{\rightsquigarrow} P_0' \quad \mathsf{std}(P_1)}{P_0 + P_1 \overset{\mu[n]}{\rightsquigarrow} P_0' + P_1} \qquad \frac{P \overset{\mu[n]}{\rightsquigarrow} P' \quad \mu, \overline{\mu} \notin A}{P \setminus A \overset{\mu[n]}{\rightsquigarrow} P' \setminus A} \qquad \frac{P \overset{\mu[n]}{\rightsquigarrow} P'}{P[f] \overset{f(\mu)[n]}{\rightsquigarrow} P'[f]}$$

Table 1: Semantics of CCSK [28]

3. Reversible Bundle Event Structures

In this section we define the reversible bundle event structures which we intend to use for defining denotational true concurrency semantics of CCSK in Section 4. For this we want a categorical definition, since we can use morphisms to determine relationships between the event structures generated by different processes and use products and coproducts to define parallel composition and choice operators. Forward-only bundle event structures were introduced by [21], but have not yet been defined categorically. We therefore start by giving a categorical formulation of bundle event structures in Section 3.1 before moving on to reversible bundle event structures in Section 3.2.

3.1. Bundle event structures

Bundle event structures (BES) (Definition 3.1) extend prime event structures by allowing multiple possible causes for the same event. They do this by replacing the causal relation with a bundle set, so that if $X \mapsto e$ then one of the events in X must have happened before e can happen. This gives us the configurations described in Definition 3.3.

Definition 3.1 (Bundle Event Structure [21]). A bundle event structure (BES) is a triple $\mathcal{E} = (E, \mapsto, \sharp)$ where:

1. E is the set of events;
2. $\sharp \subseteq E \times E$ is the irreflexive and symmetric conflict relation;
3. $\mapsto \subseteq 2^E \times E$ is the bundle set, satisfying $X \mapsto e \Rightarrow \forall e_1, e_2 \in X.(e_1 \neq e_2 \Rightarrow e_1 \sharp e_2)$.

BESs allow events to have infinitely many causes, as there is no limit on the number of bundles per event, which enables them to model certain behaviours that general event structures cannot. We therefore define a subcategory of finitely caused bundle event structures in Definition 3.2, which can be modelled by general event structures.

Definition 3.2 (Finitely Caused Bundle Event Structure). A finitely caused BES (FCBES) is a BES $\mathcal{E} = (E, \mapsto, \sharp)$ where for any $e \in E$, $\{X \subseteq E \mid X \mapsto e\}$ is finite.

Definition 3.3 (BES configuration [21]). Given a BES $\mathcal{E} = (E, \mapsto, \sharp)$, a configuration of $\mathcal{E}$ is a set $X \subseteq E$ such that:

1. X is conflict-free, that is, no events $e, e' \in X$ exist such that $e \sharp e'$;
2. there exists a sequence $e_1, \ldots, e_n$ ($n \geq 0$), such that $X = \{e_1, \ldots, e_n\}$ and for all i, $1 \leq i \leq n$, if $Y \mapsto e_{i+1}$ then $\{e_1, \ldots, e_i\} \cap Y \neq \emptyset$.

A category of BESs has not, to our knowledge, been defined, and so we define a BES morphism in Definition 3.4. We want to say that the events of E_0 can behave the same way as those they synchonise with in E_1, but the bundle sets mean this is somewhat harder to describe than in other event structures. If we said that $f(X) \mapsto f(e)$ implies $X \mapsto e$, we would be requiring $X' \mapsto e$ for every $X' = X \cup X''$ where $e \in X'' \Rightarrow f(e) = \bot$, and by extension $e \sharp e'$ if $f(e) = f(e') = \bot$. As this is not what we want, we instead adopt the constraint seen in Definition 3.4.

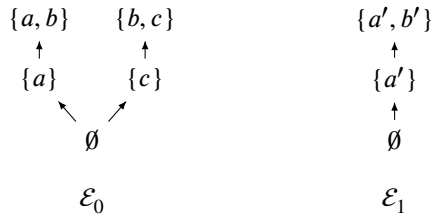


Figure 3: The configurations of the BESs discussed in Example 3.5.

Definition 3.4 (BES morphism). Given BESs $\mathcal{E}_0 = (E_0, \mapsto_0, \sharp_0)$ and $\mathcal{E}_1 = (E_1, \mapsto_1, \sharp_1)$, a BES morphism from $\mathcal{E}_0$ to $\mathcal{E}_1$ is a partial function $f : E_0 \to E_1$ such that for all $e, e' \in E_0$:

1. if $f(e) \sharp_1 f(e')$ then $e \sharp_0 e'$;
2. if $f(e) = f(e') \neq \bot$ and $e \neq e'$ then $e \sharp_0 e'$;
3. for $X_1 \subseteq E_1$ if $X_1 \mapsto_1 f(e)$ then there exists $X_0 \subseteq E_0$ such that $X_0 \mapsto_0 e$, $f(X_0) \subseteq X_1$, and if $e' \in X_0$ then $f(e') \neq \bot$.

Example 3.5 (BES morphism)**.** *Consider the two BESs $\mathcal{E}_0 = (E_0, \mapsto_0, \sharp_0)$ and $\mathcal{E}_1 = (E_1, \mapsto_1, \sharp_1)$ where $E_0 = \{a, b, c\}$, $a \sharp c$ and $\{a, c\} \mapsto b$, and $E_1 = \{a', b'\}$ and $\{a'\} \mapsto b'$, with the configurations seen in Figure 3, where an arrow from X to X' indicates that X contains an event from every bundle associated with the events in $X' \setminus X$ and $X \cup X'$ is conflict-free.*

We can define morphisms f from $\mathcal{E}_0$ to $\mathcal{E}_1$ and f' from $\mathcal{E}_1$ to $\mathcal{E}_0$ as

$$f(e) = \begin{cases} a' & \text{if } e \in \{a, c\} \\ b' & \text{if } e = b \end{cases} \qquad f'(e) = \begin{cases} a & \text{if } e = a' \\ b & \text{if } e = b' \end{cases}$$

We show that BES morphisms preserve configurations.

Proposition 3.6. *Given BESs $\mathcal{E}_0 = (E_0, \mapsto_0, \sharp_0)$ and $\mathcal{E}_1 = (E_1, \mapsto_1, \sharp_1)$ and a morphism $f : E_0 \to E_1$, if $X \subseteq E_0$ is a configuration of $\mathcal{E}_0$, then $f(X)$ is a configuration of $\mathcal{E}_1$.*

Proof. We show that $f(X)$ fulfils the conditions of Definition 3.3:

1. For any $e, e' \in X_0$, if $f(e) \sharp_1 f(e')$, then $e \sharp_0 e'$, and therefore if X_0 is conflict-free then $f(X_0)$ is conflict-free.
2. There exists a sequence $e_1, \ldots, e_n$ ($n \geq 0$), such that $X_0 = \{e_1, \ldots, e_n\}$ and for all i, $1 \leq i \leq n$, if $Y \mapsto e_{i+1}$ then $\{e_1, \ldots, e_i\} \cap Y \neq \emptyset$. Obviously $f(X_0) = \{f(e_1), \ldots, f(e_0)\}$, and for all i, if $Y_1 \mapsto f(e_{i+1})$, then there exists Y_0 such that $Y_0 \mapsto e_{i+1}$, $f(Y_0) \subseteq Y_1$, and if $e' \in Y_0$, then $f(e') \neq \bot$. Since $Y_0 \cap \{e_1, \ldots, e_i\} \neq \emptyset$, we obviously get that $Y_1 \cap \{f(e_1), \ldots, f(e_i)\} \neq \emptyset$. □

Proposition 3.7. ***BES*** *consisting of BESs and BES morphisms is a category.*

Proof. Composition of partial functions is associative and $f(e) = e$ functions as an identity arrow, and so we need only show that the morphisms are composable:

If $\mathcal{E}_0 = (E_0, \mapsto_0, \sharp_0)$, $\mathcal{E}_1 = (E_1, \mapsto_1, \sharp_1)$, and $\mathcal{E}_2 = (E_2, \mapsto_2, \sharp_2)$ are BESs and $f : E_0 \to E_1$ and $g : E_1 \to E_2$ are morphisms, we show that $f \circ g : E_0 \to E_2$ is also a morphism:

1. If $g(f(e)) \sharp_2 g(f(e'))$ then $f(e) \sharp_1 f(e')$, and therefore $e \sharp_0 e'$.
2. If $g(f(e)) = g(f(e'))$ and $e \neq e'$, then either $f(e) = f(e')$, in which case $e \sharp_0 e'$, or $f(e) \neq f(e')$, in which case $f(e) \sharp_1 f(e')$, and therefore $e \sharp_0 e'$.
3. If $X_2 \mapsto_2 g(f(e))$ then there exist $X_1 \subseteq E_1$ and $X_0 \subseteq E_0$ such that $X_1 \mapsto_1 f(e)$, $X_0 \mapsto_0 e$, $g(X_1) \subseteq X_2$, $f(X_0) \subseteq X_1$ and if $e_1 \in X_1$ then $g(e_1) \neq \bot$ and if $e_0 \in X_0$ then $f(e_0) \neq \bot$. This means that $g(f(X_0)) \subseteq X_2$, and if $e_0 \in X_0$ then $g(f(e_0)) \neq \bot$. □

We also construct a product in this category in Definition 3.8. Having products in our categories is useful for defining parallel composition in our semantics.

Definition 3.8 (BES product). Given BESs $\mathcal{E}_0 = (E_0, \mapsto_0, \sharp_0)$ and $\mathcal{E}_1 = (E_1, \mapsto_1, \sharp_1)$, we construct $\mathcal{E}_0 \times \mathcal{E}_1 = (E, \mapsto, \sharp)$ with projections π_0, π_1 where:

1. $E = E_0 \times_* E_1 = \{(e, *) \mid e \in E_0\} \cup \{(*, e) \mid e \in E_1\} \cup \{(e, e') \mid e \in E_0 \text{ and } e' \in E_1\}$;
2. for $(e_0, e_1) \in E$, $\pi_i(e_0, e_1) = e_i$;
3. for any $e \in E$, $X \subseteq E$, $X \mapsto e$ iff there exists $i \in \{0, 1\}$ and $X_i \subseteq E_i$ such that $X_i \mapsto \pi_i(e)$ and $X = \{e' \in E \mid \pi_i(e') \in X_i\}$;
4. 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 $\pi_{1-i}(e) \neq \pi_{1-i}(e')$.

Example 3.9 (Product)**.** *Consider the BESs $\mathcal{E}_0$ with events a, b and $\mathcal{E}_1$ with event c such that $\{a\} \mapsto b$. Then $\mathcal{E}_0 \times \mathcal{E}_1$ has the bundles $\{(a, *), (a, c), \} \mapsto (b, *)$ and $\{(a, *), (a, c), \} \mapsto (b, c)$ and conflict $(a, *) \sharp (a, c)$, $(b, *) \sharp (b, c)$, $(*, c) \sharp (a, c)$, $(*, c) \sharp (b, c)$, and $(a, c) \sharp (b, c)$.*

Proposition 3.10. *Given BESs $\mathcal{E}_0 = (E_0, \mapsto_0, \sharp_0)$ and $\mathcal{E}_1 = (E_1, \mapsto_1, \sharp_1)$, we have that $\mathcal{E}_0 \times \mathcal{E}_1 = (E, \mapsto, \sharp)$ is their product in the category **BES**.*

Proof. We define f as $f(e) = (f_0(e), f_1(e))$ and prove that f, π_0, and π_1 are morphisms in Appendix B.1. □

Proposition 3.11. *Given FCBESs $\mathcal{E}_0$ and $\mathcal{E}_1$, we have that $\mathcal{E}_0 \times \mathcal{E}_1$ is a FCBES.*

Proof. We say that $\mathcal{E}_0 = (E_0, \mapsto_0, \sharp_0)$ and $\mathcal{E}_1 = (E_1, \mapsto_1, \sharp_1)$.

For any $(e_0, e_1) \in E$, clearly $\{X \subseteq E \mid X \mapsto (e_0, e_1)\} = \{\{e' \in E \mid \pi_0(e') \in X_0\} \mid X_0 \mapsto_0 e_0\} \cup \{\{e' \in E \mid \pi_1(e') \in X_1\} \mid X_1 \mapsto_1 e_1\}$, which is finite because $\{X_0 \mid X_0 \mapsto_0 e_0\}$ and $\{X_1 \mid X_1 \mapsto_1 e_1\}$ are finite. □

We construct a coproduct of BESs in Definition 3.12. We will later be able to use the coproducts in our categories when modelling choices in CCSK.

Definition 3.12 (BES coproduct). Given BESs $\mathcal{E}_0 = (E_0, \mapsto_0, \sharp_0)$ and $\mathcal{E}_1 = (E_1, \mapsto_1, \sharp_1)$, we construct $\mathcal{E}_0 + \mathcal{E}_1 = (E, \mapsto, \sharp)$ with injections ι_0, ι_1 where:

- $E = \{(0, e) \mid e \in E_0\} \cup \{(1, e) \mid e \in E_1\}$;
- for $e \in E_j$, $\iota_j(e) = (j, e)$ for $j \in \{0, 1\}$;
- $X \mapsto (j, e)$ iff for all $(j', e') \in X$, $j = j'$ and $\iota_j(X) \mapsto_j e$;
- $(j, e) \sharp (j', e')$ iff $j \neq j'$ or $e \sharp_j e'$.

Proposition 3.13. *Given BESs $\mathcal{E}_0$ and $\mathcal{E}_1$, we have that $\mathcal{E}_0 + \mathcal{E}_1$ is their coproduct in the category **BES**.*

Proof. We define f as $f(j, e) = f_j(e)$ and prove that f, ι_0, and ι_1 are morphisms in Appendix B.2. □

Proposition 3.14. *Given FCBESs $\mathcal{E}_0$ and $\mathcal{E}_1$, we have that $\mathcal{E}_0 + \mathcal{E}_1$ is a FCBES.*

Proof. Follows straightforwardly from Definitions 3.2 and 3.12. □

3.2. Reversible bundle event structures

We define reversible bundle structures (RBES) by extending the bundle relation to map to reverse events, denoted $\underline{e}$, and adding a prevention relation, such that if $e \rhd \underline{e'}$ then e' cannot be reversed from configurations containing e. We use e^* to denote either e or $\underline{e}$.

Definition 3.15 (Reversible Bundle Event Structure). An RBES is a 5-tuple $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd)$ where:

1. E is the set of events;
2. $F \subseteq E$ is the set of reversible events;
3. the conflict relation, $\sharp \subseteq E \times E$, is symmetric and irreflexive;
4. 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 $e \in F$, $\{e\} \mapsto \underline{e}$;
5. $\rhd \subseteq E \times \underline{F}$ is the prevention relation.

Definition 3.16 (Finitely Caused Reversible Bundle Event Structure). A finitely caused RBES (FCRBES) is an RBES $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd)$ where for any $e^* \in E \cup \underline{F}$, $\{X \subseteq E \mid X \mapsto e^*\}$ is finite.

Example 3.17 shows the configurations of an RBES. The configuration $\{b, c\}$ is reachable despite b being required for c to happen, and c being a possible cause of b. In future examples we will leave out bundles on the form $\{e\} \mapsto \underline{e}$, since they can be assumed to exist for any $e \in F$.

Example 3.17 (RBES). *An RBES $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd)$ where $E = \{a, b, c\}$, $F = \{a, b\}$, $a \sharp c$, $\{a, c\} \mapsto b$, $\{b\} \mapsto c$ $\{a\} \mapsto \underline{a}$, $\{b\} \mapsto \underline{a}$, and $\{b\} \mapsto \underline{b}$, has the configurations seen in Figure 4 where we use the dotted arrow to indicate a reverse bundle.*

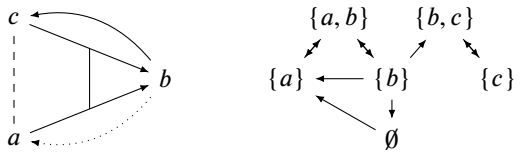


Figure 4: The configurations of the RBES described in Example 3.17.

Once again, in order to get a categorical definition of RBESs, we define a morphism in Definition 3.18. It is very similar to Definition 3.4, and treats prevention in the same way as conflict.

Definition 3.18 (RBES morphism). Given RBESs $\mathcal{E}_0 = (E_0, F_0, \mapsto_0, \sharp_0, \rhd_0)$ and $\mathcal{E}_1 = (E_1, F_1, \mapsto_1, \sharp_1, \rhd_1)$, an RBES morphism from $\mathcal{E}_0$ to $\mathcal{E}_1$ is a partial function $f : E_0 \to E_1$ such that $f(F_0) \subseteq F_1$ and for all $e, e' \in E_0$:

1. if $f(e) \sharp_1 f(e')$ then $e \sharp_0 e'$;
2. if $f(e) = f(e') \neq \bot$ and $e \neq e'$ then $e \sharp_0 e'$;
3. for $X_1 \subseteq E_1$ if $X_1 \mapsto_1 f(e)^*$ then there exists $X_0 \subseteq E_0$ such that $X_0 \mapsto_0 e^*$, $f(X_0) \subseteq X_1$, and if $e' \in X_0$ then $f(e') \neq \bot$;
4. if $f(e) \rhd_1 \underline{f(e')}$ then $e \rhd_0 \underline{e'}$.

Proposition 3.19. ***RBES*** *consisting of RBESs and RBES morphisms is a category.*

Proof. Composition of partial functions is associative, and $f(e) = e$ functions as an identity arrow, and the morphisms are obviously composable. □

As we did for **BES**, we construct a product of RBESs in Definition 3.20.

Definition 3.20 (RBES product). Given RBESs $\mathcal{E}_0 = (E_0, F_0, \mapsto_0, \sharp_0, \rhd_0)$ and $\mathcal{E}_1 = (E_1, F_1, \mapsto_1, \sharp_1, \rhd_1)$, we construct $\mathcal{E}_0 \times \mathcal{E}_1 = (E, F, \mapsto, \sharp, \rhd)$ with projections π_0, π_1 where:

1. $E = E_0 \times_* E_1 = \{(e, *) \mid e \in E_0\} \cup \{(*, e) \mid e \in E_1\} \cup \{(e, e') \mid e \in E_0 \text{ and } e' \in E_1\}$;
2. $F = F_0 \times_* F_1 = \{(e, *) \mid e \in F_0\} \cup \{(*, e) \mid e \in F_1\} \cup \{(e, e') \mid e \in F_0 \text{ and } e' \in F_1\}$;
3. for $(e_0, e_1) \in E$, $\pi_i(e_0, e_1) = e_i$;
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 $\pi_i(e) \lhd_i \underline{\pi_i(e')}$.

Proposition 3.21. *Given RBESs $\mathcal{E}_0 = (E_0, F_0, \mapsto_0, \sharp_0, \rhd_0)$ and $\mathcal{E}_1 = (E_1, F_1, \mapsto_1, \sharp_1, \rhd_1)$, we have that $\mathcal{E}_0 \times \mathcal{E}_1$ is their product in the category **RBES**.*

Proof. Similar to the proof of Proposition 3.10. □

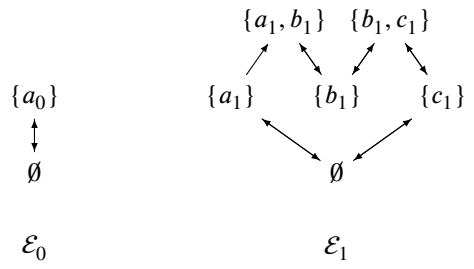


Figure 5: The configurations of the event structures discussed in Example 3.22.

Example 3.22. *Consider the RBESs* $\mathcal{E}_0 = (E_0, F_0, \mapsto_0, \sharp_0, \rhd_0)$ *and* $\mathcal{E}_1 = (E_1, F_1, \mapsto_1, \sharp_1, \rhd_1)$ *where* $E_0 = F_0 = \{a_0\}$ *and* $E_1 = \{a_1, b_1, c_1\}$, $F_1 = \{a_1, b_1\}$, $a_1 \sharp_1 c_1$, $\{a_1, c_1\} \mapsto_1 b_1$, *and* $\{c_1\} \mapsto_1 \underline{b_1}$, *with the configurations seen in Figure 5.*

Then $\mathcal{E}_0 \times \mathcal{E}_1 = (E, F, \mapsto, \overline{\sharp}, \rhd)$ *where*

$E = \{(a_0, *), (a_0, a_1), (a_0, b_1), (a_0, c_1), (*, a_1), (*, b_1), (*, c_1)\}$

$F = \{(a_0, *), (a_0, a_1), (a_0, b_1), (*, a_1), (*, b_1)\}$

$(a_0, a_1) \sharp (a_0, b_1)$ $\quad$ $(a_0, b_1) \sharp (a_0, c_1)$

$(a_0, a_1) \sharp (a_0, c_1)$ $\quad$ $(a_0, a_1) \sharp (a_0, *)$

$(a_0, b_1) \sharp (a_0, *)$ $\quad$ $(a_0, c_1) \sharp (a_0, *)$

$(a_0, a_1) \sharp (*, a_1)$ $\quad$ $(a_0, b_1) \sharp (*, b_1)$

$(a_0, c_1) \sharp (*, c_1)$ $\quad$ $(a_0, a_1) \sharp (a_0, c_1)$

$(*, a_1) \sharp (*, c_1)$ $\quad$ $(a_0, a_1) \sharp (*, c_1)$

$(*, a_1) \sharp (a_0, c_1)$

$\{(a_0, a_1), (a_0, c_1), (*, a_1), (*, c_1)\} \mapsto (a_0, b_1)$ $\quad$ $\{(a_0, c_1), (*, c_1)\} \mapsto \underline{(*, b_1)}$

$\{(a_0, a_1), (a_0, c_1), (*, a_1), (*, c_1)\} \mapsto (*, b_1)$ $\quad$ $\{(a_0, c_1), (*, c_1)\} \mapsto \underline{(a_0, b_1)}$

Proposition 3.23. *Given FCRBESs* $\mathcal{E}_0$ *and* $\mathcal{E}_1$, *we have that* $\mathcal{E}_0 \times \mathcal{E}_1$ *is a FCRBES.*

Proof. Similar to the proof of Proposition 3.11. □

Definition 3.24 (RBES coproduct). Given RBESs $\mathcal{E}_0 = (E_0, F_0, \mapsto_0, \sharp_0, \rhd_0)$ and $\mathcal{E}_1 = (E_1, F_1, \mapsto_1, \sharp_1, \rhd_1)$, we construct $\mathcal{E}_0 + \mathcal{E}_1 = (E, F, \mapsto, \sharp, \rhd)$ with injections ι_0, ι_1 where:

- $E = \{(0, e) \mid e \in E_0\} \cup \{(1, e) \mid e \in E_1\}$
- $F = \{(0, e) \mid e \in F_0\} \cup \{(1, e) \mid e \in F_1\}$
- for $e \in E_j$, $\iota_j(e) = (j, e)$ for $j \in \{0, 1\}$
- $X \mapsto (j, e)^*$ iff for all $(j', e') \in X$, $j = j'$ and $\iota_j(X) \mapsto_j e^*$
- $(j, e) \sharp (j', e')$ iff $j \neq j'$ or $e \sharp_j e'$
- $(j, e) \rhd \underline{(j', e')}$ iff $j \neq j'$ or $e \rhd_j \underline{e'}$

Proposition 3.25. *Given RBESs* $\mathcal{E}_0$ *and* $\mathcal{E}_1$, *we have that* $\mathcal{E}_0 + \mathcal{E}_1$ *is their coproduct in the category* ***RBES****.*

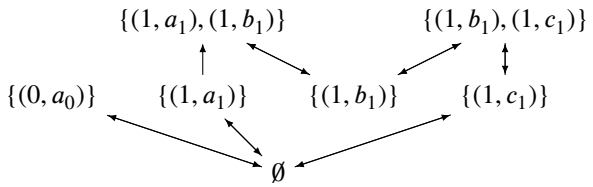


Figure 6: The Configurations of the RBES discussed in Example 3.26.

Proof. Similar to the BES coproduct (Proposition 3.13). □

Example 3.26. *Consider again the RBESs $\mathcal{E}_0$ and $\mathcal{E}_1$ from Example 3.22. We can also find $\mathcal{E}_0 + \mathcal{E}_1 = (E, F, \mapsto, \sharp, \rhd)$ where*

$E = \{(0, a_0), (0, b_0), (1, a_1), (1, b_1), (1, c_1)\}$
$F = \{(0, a_0), (0, b_0), (1, a_1), (1, b_1)\}$
$(1, a_1) \sharp (1, c_1)$
$\{(0, a_0)\} \mapsto (0, b_0)$
$\{(1, a_1), (1, c_1)\} \mapsto (1, b_1)$
$\{(1, c_1)\} \mapsto \underline{(1, b_1)}$
$(0, b_0) \rhd \underline{(0, a_0)}$

with the configurations seen in Figure 6.

Proposition 3.27. *Given FCRBESs $\mathcal{E}_0$ and $\mathcal{E}_1$, we have that $\mathcal{E}_0 + \mathcal{E}_1$ is a FCRBES.*

Proof. Follows straightforwardly from Definitions 3.16 and 3.24. □

We want to model RBESs as configuration systems (CSs), and therefore define a functor from one category to the other in Definition 3.29. A CS consists of a set of events, some of which are reversible, configurations of these events, and labelled transitions between them, as described in Definition 3.28. We will later use the CSs corresponding to our event structure semantics to describe the operational correspondence between our event structure semantics and the operational semantics of CCSK.

Definition 3.28 (Configuration system [27]). A configuration system (CS) is a quadruple $\mathcal{C} = (E, F, \mathsf{C}, \rightarrow)$ 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:

- $A \cap X = \emptyset$; $B \subseteq X \cap F$; $Y = (X \setminus B) \cup A$;
- and 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 3.29 (From RBES to CS). The functor $C_{br} : \mathbf{RBES} \rightarrow \mathbf{CS}$ is defined as:

1. $C_{br}((E, F, \mapsto, \sharp, \rhd)) = (E, F, \mathsf{C}, \rightarrow)$ where

(a) $X \in \mathsf{C}$ if X is conflict-free;

(b) for $X, Y \in \mathsf{C}$, $A \subseteq E$, and $B \subseteq F$, there exists a transition $X \xrightarrow{A \cup \underline{B}} Y$ if

i. $Y = (X \setminus B) \cup A$;
ii. $X \cap A = \emptyset$;
iii. $B \subseteq X$;
iv. $X \cup A$ is conflict-free;
v. for all $e \in B$, if $e' \rhd \underline{e}$ then $e' \notin X \cup A$;
vi. for all $e \in A$ and $X' \subseteq E$, if $X' \mapsto e$ then $X' \cap (X \setminus B) \neq \emptyset$;
vii. for all $e \in B$ and $X' \subseteq E$, if $X' \mapsto \underline{e}$ then $X' \cap (X \setminus (B \setminus \{e\})) \neq \emptyset$;

2. $C_{br}(f) = f$.

Proposition 3.30. *C_{br} is a functor from **RBES** to **CS**.*

Proof. The definition of a CS morphism and proof can be seen in Appendix B.3. □

We define cause-respecting and causal variants of RBES in Definition 3.31. In a cause-respecting RPES events cannot reverse if they have caused a subsequent event and in a causal RPES events can reverse if and only if they have not caused a subsequent event. We also define the categories **crRBES** and **CRBES**, consisting of respectively cause-respecting RBESs and causal RBESs and the morphisms between them.

Definition 3.31 (cause-respecting and causal RBES). We say that $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd)$ is a cause-respecting RBES (crRBES) if whenever $X \mapsto e$ and $e' \in X \cap F$, then $e \rhd \underline{e'}$.

We say that $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd)$ is a causal RBES (CRBES) if (1) if $e \rhd \underline{e'}$ then either $e \sharp e'$ or there exists an $X \subseteq E$ such that $X \mapsto e$ and $e' \in X$, (2) if $X \mapsto e$ and $e' \in X \cap F$, then $e \rhd \underline{e'}$, and (3) if $X \mapsto \underline{e}$ then $e \in X$.

Proposition 3.32. *Given a crRBES, $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd)$ with a corresponding CS $C_{rb}(\mathcal{E}) = (E, F, \mathsf{C}, \rightarrow)$, any reachable $C \in \mathsf{C}$ is forwards-reachable.*

Proof. There exists a trace $\emptyset \xrightarrow{\{e_0^*\}} C_0 \xrightarrow{\{e_1^*\}} C_1 \ldots \xrightarrow{\{e_n^*\}} C_n$ where $C_n = C$. Clearly e_0^* is a forward event e_0, and C_0 is forwards-reachable, and we will show that if C_j is forwards-reachable for $0 \leq j \leq i$, then C_{i+1} is forwards-reachable.

If e_{i+1}^* is a forwards event, then obviously C_{i+1} is forwards-reachable.

If $e_{i+1}^* = \underline{e_i}$, then $C_{i+1} = C_{i-1}$, which is obviously forwards-reachable.

If $e_{i+1}^* = \underline{e_j}$ for some $0 \leq j < i$, then for all $0 \leq j' \leq i$, since $e_{j'} \not\rhd \underline{e_j}$, there does not exist $X \subseteq E$ such that $e_j \in X$ and $X \mapsto e_{j'}$. This means $\emptyset \xrightarrow{\{e_0^*\}} C_0 \xrightarrow{\{e_1^*\}} C_1 \ldots \xrightarrow{\{e_{j-1}^*\}} C_{j-1} \xrightarrow{\{e_{j+1}^*\}} C_{j+1} \setminus \{e_j\} \ldots \xrightarrow{\{e_i^*\}} C_{i+1}$. □

Proposition 3.33.

1. *If $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd)$ is a crRBES and $C_{br}(\mathcal{E}) = (E, F, \mathsf{C}, \rightarrow)$ then whenever $X \in \mathsf{C}$ is a reachable configuration and $X \xrightarrow{\underline{B}} Y$, there exists a transition $Y \xrightarrow{B} X$.*

2. *If* $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd)$ *is a CRBES and* $C_{br}(\mathcal{E}) = (E, F, \mathsf{C}, \rightarrow)$ *then whenever* $X \in \mathsf{C}$, $X \xrightarrow{A \cup \underline{B}} Y$ *and* $A \cup B \subseteq F$, *there exists a transition* $Y \xrightarrow{B \cup \underline{A}} X$.

Proof.

1. By Proposition 3.32, X is forwards reachable, meaning for every $e \in X$, if $X' \mapsto e$ then there exists e' such that $X' \cap X = \{e'\}$. For each $e_b \in B$, if $X_b \mapsto e_b$ and $X_b \cap X = \{e'_b\}$ then either $e'_b \notin F$ or $e_b \rhd \underline{e'_b}$, meaning $e'_b \notin B$, and therefore clearly $Y \xrightarrow{B} X$.
2. For any forwards-reachable configuration $X \in \mathsf{C}$ and $A, B \subseteq F$, if $X \xrightarrow{A \cup \underline{B}} (X \cup A) \setminus B$ then $(X \cup A) \setminus B \xrightarrow{B \cup \underline{A}} X$ according to Definition 3.29:
   i to iv follow from $X \xrightarrow{A \cup \underline{B}} (X \cup A) \setminus B$ being a transition.
   v. For all $e \in A$ and $e' \in E$, if $e' \rhd \underline{e}$, then either $e' \sharp e$, or there exists $X' \subseteq E$ such that $X' \mapsto e'$ and $e \in X'$.
   If $e' \sharp e$, then, as $X \cup A$ is conflict-free, $e' \notin X \cup A$.
   If there exists $X' \subseteq E$ such that $X' \mapsto e'$ and $e \in X'$ then for all $e'' in X' \setminus \{e\}$ we know $e'' \sharp e$, meaning $e'' \notin X \cup A$. This means $X \cap X' = \emptyset$, and therefore for all $X'' \subseteq X$, $X'' \not\xrightarrow{e'}$, and consequently $e' \notin X \cup A$.
   vi. For all $e \in B$ and $X' \subseteq E$, if $X' \mapsto e$, then, since X is forwards-reachable, $X' \cap X \neq \emptyset$. If $X' \cap X \setminus B = \emptyset$, then there exists $e' \in X' \cap B$. But this means $e \rhd \underline{e'}$, conflicting with $X \xrightarrow{A \cup \underline{B}} (X \cup A) \setminus B$.
   vii. For all $e \in A$ and $X' \subseteq E$, if $X' \mapsto \underline{e}$, then $e \in X'$. □

We define functors between **BES** and **RBES**, and show that they form an adjunction, meaning that applying first Φ_b going from **RBES** to **BES** and then P_b going from **BES** to **RBES** to an RBES always yields an under-estimation of the original, in that it is the original with all actions made irreversible. We shall rely on the following characterisation of adjunctions, based on [2, Definition 9.1].

Definition 3.34 (Adjunction). Let $\mathbf{C}$ and $\mathbf{D}$ be categories, and let $F : \mathbf{D} \rightarrow \mathbf{C}$ and $G : \mathbf{C} \rightarrow \mathbf{D}$ be functors. Then F is a *left adjoint* of G, $F \dashv G$, if there exists a natural transformation $\eta : I_{\mathbf{D}} \rightarrow F \circ G$ (the *unit*) such that for any $c \in \mathbf{C}$, $d \in \mathbf{D}$ and morphism $g : d \rightarrow G(c)$ there is a unique morphism $f : F(d) \rightarrow c$ such that $g = \eta_d \circ G(f)$, where η_d is the component of η at d.

Definition 3.35 (BES to RBES). The functor $P_b : \mathbf{BES} \rightarrow \mathbf{RBES}$ is defined as:

1. $P_b((E, \mapsto, \sharp)) = (E, \emptyset, \mapsto, \sharp, \emptyset \times \emptyset)$;
2. $P_b(f) = f$.

Definition 3.36 (RBES to BES). The functor $\Phi_b : \mathbf{RBES} \rightarrow \mathbf{BES}$ is defined as:

1. $\Phi_b((E, F, \mapsto, \sharp, \rhd)) = (E, \mapsto \cap (E^2 \times E), \sharp)$;
2. $\Phi_b(f) = f$.

Proposition 3.37. $P_b \dashv \Phi_b$.

Proof. For any RBES $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd)$, clearly $P_b(\Phi_b(\mathcal{E})) = (E, \emptyset, \sharp, \mapsto_E, \emptyset \times \emptyset)$ with $\mapsto_E = \mapsto \cap (E^2 \times E)$, and we define $\eta : P_b(\Phi_b(\mathcal{E})) \to \mathcal{E}$ such that for all $e \in E$ $\eta(e) = e$, and prove that it is an RBES morphism according to Definition 3.18

1. P_b and Φ_b do not change conflict.
2. If $f(e) = f(e')$ then $e = e'$.
3. This means $f(e)^* = f(e)$, and clearly if $X \mapsto_E f(e)$, then $X = f(X) \mapsto e$.
4. There are no $e' \in \emptyset$.

We then show that given a BES $\mathcal{E}_A = (E_A, \mapsto_A, \sharp_A)$, an RBES $\mathcal{E}_B = (E_B, F_B, \mapsto_B, \sharp_B, \rhd_B)$, and an RBES morphism $g : P_p(A) \to B$, $f : A \to \Phi_p(B)$ is a BES morphism according to Definition 3.4:

1. P_b and Φ_b do not change conflict.
2. If $f(e) = f(e')$ then $e = e'$.
3. P_p and Φ_p do not affect $\mapsto_{A_E}$ or $\mapsto_{B_E}$. □

We also wish to relate RBESs to previously defined reversible event structures as shown in Figure 2 in Section 1, reversible prime event structures (RPESs) [27] and reversible stable general event structures (SRESs) [29], so we define functors from **RPES** to **RBES** and from **FCRBES** to **SRES** in Appendix A.

Since our motivation for defining RBESs was modelling reversible processes, we need to be able to label our events with a corresponding action from a process. For this we use a *labelled RBES* (LRBES).

Definition 3.38 (Labelled Reversible Bundle Event Structure). A labelled reversible bundle event structure $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$ consists of an RBES $(E, F, \mapsto, \sharp, \rhd)$, a set of labels Act, and a surjective labelling function $\lambda : E \to \mathsf{Act}$.

Definition 3.39 (LRBES morphism). Let $\mathcal{E}_0 = (E_0, F_0, \mapsto_0, \sharp_0, \rhd_0, \lambda_0, \mathsf{Act}_0)$ and $\mathcal{E}_1 = (E_1, F_1, \mapsto_1, \sharp_1, \rhd_1, \lambda_1, \mathsf{Act}_1)$ be LRBESs. An LRBES morphism $f : \mathcal{E}_0 \to \mathcal{E}_1$ is a partial function $f : E_0 \to E_1$ such that $f : (E_0, F_0, \mapsto_0, \sharp_0, \rhd_0) \to (E_1, F_1, \mapsto_1, \sharp_1, \rhd_1)$ is an RBES morphism and for all $e \in E_0$, either $f(e) = \bot$ or $\lambda_0(e) = \lambda_1(f(e))$.

4. Event Structure Semantics of CCSK

Having defined RBESs, we will now use them to describe the semantics of CCSK. Unlike the event structure semantics of CCS [3, 31], our semantics will generate both an event structure and an *initial configuration* containing all the events corresponding to past actions. This means that for any CCSK processes P and P', if $P \to P'$ then P and P' will be described by the same, or at least isomorphic, event structures with different initial states.

First we define the operators we will use in the semantics, particularly restriction, parallel composition, choice, and action prefixes. Restriction is done by simply removing any events associated with the restricted action.

Definition 4.1 (Restriction). Given an LRBES $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$, restricting $\mathcal{E}$ to $E' \subseteq E$ creates $\mathcal{E} \upharpoonright E' = (E', F', \mapsto', \sharp', \rhd', \lambda', \mathsf{Act}')$ 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. $\lambda' = \lambda \upharpoonright_{E'}$;
6. $\mathsf{Act} = \mathsf{ran}(\lambda')$.

Parallel composition uses the product of RBESs, labels any event corresponding to a synchronisation τ, and removes any invalid events describing an impossible synchronisation.

Definition 4.2 (Parallel). 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 is $\mathcal{E}_0 || \mathcal{E}_1 = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act}) \upharpoonright \{e \mid \lambda(e) \neq 0\}$ where:

1. $(E, F, \mapsto, \sharp, \rhd) = (E_0, F_0, \mapsto_0, \sharp_0, \rhd_0) \times (E_1, F_1, \mapsto_1, \sharp_1, \rhd_1)$
2. $$\lambda(e) = \begin{cases} \lambda_0(e_0) & \text{if } e = (e_0, *) \\ \lambda_1(e_1) & \text{if } e = (*, e_1) \\ \tau & \text{if } e = (e_0, e_1) \text{ and } \lambda_0(e_0) = \overline{\lambda_1(e_1)} \\ 0 & \text{if } e = (e_0, e_1) \text{ and } \lambda_0(e_0) \neq \overline{\lambda_1(e_1)} \end{cases}$$
3. $\mathsf{Act} = \mathsf{Act}_0 \cup \mathsf{Act}_1 \cup \{0, \tau\}$

Choice, which act as a coproduct of LRBESs, simply uses the coproduct of RBESs, and defines the labels as expected.

Definition 4.3 (Choice). Given LRBESs $\mathcal{E}_0 = (E_0, F_0, \mapsto_0, \sharp_0, \rhd_0, \lambda_0, \mathsf{Act}_0)$ and $\mathcal{E}_1 = (E_1, F_1, \mapsto_1, \sharp_1, \rhd_1, \lambda_1, \mathsf{Act}_1)$;, the choice between them is $\mathcal{E}_0 + \mathcal{E}_1 = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$ where:

1. $(E, F, \mapsto, \sharp, \rhd) = (E_0, F_0, \mapsto_0, \sharp_0, \rhd_0) + (E_1, F_1, \mapsto_1, \sharp_1, \rhd_1)$;
2. $\lambda(j, e) = \lambda_j(e)$;
3. $\mathsf{Act} = \mathsf{Act}_0 \cup \mathsf{Act}_1$.

Proposition 4.4. *If $\mathcal{E}_0$ and $\mathcal{E}_1$ are LRBESs, then $\mathcal{E} = \mathcal{E}_0 + \mathcal{E}_1$ with injections ι_0 and ι_1 such that $\iota_j(j, e) = e$ is their coproduct.*

Proof. Obviously $\mathcal{E}$ is an LRBES, and ι_0 and ι_1 are morphisms, and so we simply need to prove that if there exists an LRBES $\mathcal{E}_2 = (E_2, F_2, \mapsto_2, \sharp_2, \rhd_2, \lambda_2, \mathsf{Act}_2)$ and morphisms $f_0 : \mathcal{E}_0 \to \mathcal{E}_2$ and $f_1 : \mathcal{E}_1 \to \mathcal{E}_2$, then there exists a unique LRBES-morphism $f : \mathcal{E} \to \mathcal{E}_2$ such that $f_0 = f \circ \iota_0$ and $f_1 = f \circ \iota_1$.

Since $E_0 + E_1$, ι_0, and ι_1 make up a coproduct in the category of sets and partial functions, f must be unique.

We define f as $f(j, e) = f_j(e)$ and prove it to be a morphism. Since $(E, F, \mapsto, \sharp, \rhd) = (E_0, F_0, \mapsto_0, \sharp_0, \rhd_0) + (E_1, F_1, \mapsto_1, \sharp_1, \rhd_1)$, we know $f : (E, F, \mapsto, \sharp, \rhd) \to (E_2, F_2, \mapsto_2, \sharp_2, \rhd_2)$ is an RBES-morphism, and clearly $\lambda(e, j) = \lambda_j(e) = \lambda_2(f_j(e))$. □

Causally prefixing an action onto an event structure means we add a new event such that the new event causes all other events and is prevented from reversing by all other events and has the designated label.

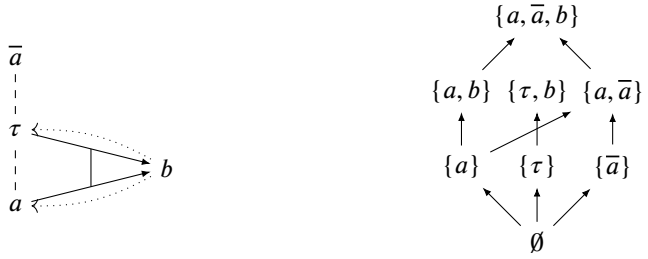


Figure 7: LRBES and configurations of the process in Example 4.6.

Definition 4.5 (Causal Prefix). Given an LRBES $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$, an event $e \notin E$, and a label α, we add e labelled α to the beginning of $\mathcal{E}$ to get $\alpha(e).\mathcal{E} = (E', F', \mapsto', \sharp', \rhd', \lambda', \mathsf{Act}')$ 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. $\lambda' = \lambda \cup \{(e, \alpha)\}$;
7. $\mathsf{Act}' = \mathsf{Act} \cup \{\alpha\}$.

Now that we have defined the main operations of the process calculus, we define the event structure semantics in Table 2. We do this using rules of the form $\{\!|P|\!\}_l = \langle \mathcal{E}, \mathsf{Init}, k \rangle$ wherein l is the level of unfolding, which we use to model recursion, $\mathcal{E}$ is an LRBES, Init is the initial configuration, 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 Init.

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.

Example 4.6. *The CCSK process $a.b \mid \overline{a}$ can be represented by the RBES with events labelled a, $\overline{a}$, τ, and b, the bundle $\{a, \tau\} \mapsto b$, the conflicts $a \sharp \tau$ and $\overline{a} \sharp \tau$, and the preventions $b \rhd \underline{a}$ and $b \rhd \underline{\tau}$, creating the LRBES seen in Figure 7 where we label events with their labels.*

We say that $\{\!|P|\!\} = \sup_{l \in \mathcal{N}} \{\!|P|\!\}_l$. This means we need to show that there exists such a least upper bound of the levels of unfolding. As shown in [12], ordering closed BESs by restriction produces a complete partial order. Since our LRBESs do not have overlapping bundles ($X \mapsto e^*$ and $X' \mapsto e^*$ implies $X \neq X'$ or $X \cap X' = \emptyset$) they are closed, and we can use a similar ordering.

Definition 4.7 (Ordering of LRBESs). Given LRBESs $\mathcal{E}_0 = (E_0, F_0, \mapsto_0, \sharp_0, \rhd_0, \lambda_0, \mathsf{Act}_0)$ and $\mathcal{E}_1$, we say that $\mathcal{E}_0 \leq \mathcal{E}_1$ if $\mathcal{E}_0 = \mathcal{E}_1 \upharpoonright E_0$.

We can see that $\leq$ is a partial order with the empty LRBES as its minimum.

Proposition 4.8. *Any ω-chain $\mathcal{E}_0 \leq \mathcal{E}_1 \leq \mathcal{E}_2 \ldots$ has a least upper bound $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$ where:*

$$
\begin{array}{ll}
\{\!|0|\!\}_I = & \langle(\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\emptyset),\emptyset,\emptyset\rangle \\
\{\!|P_0+P_1|\!\}_I = & \langle \mathcal{E}_0+\mathcal{E}_1, \mathsf{Init}, k\rangle \text{ where} \\
& \text{For } i\in\{0,1\},\ \{\!|P_i|\!\}_I = \langle \mathcal{E}_i, \mathsf{Init}_i, k_i\rangle \\
& \mathsf{Init} = \{(j,e) \mid j\in\{0,1\} \text{ and } e\in \mathsf{Init}_j\} \\
& k(j,e) = k_j(e) \text{ if } e\in \mathsf{Init}_j \\
\{\!|\alpha.P|\!\}_I = & \left\langle \alpha(e).(E,F,\mapsto,\sharp,\rhd,\lambda,\mathsf{Act}), \mathsf{Init}, k\right\rangle \text{ for } e \text{ fresh for } E \text{ where} \\
& \{\!|P|\!\}_I = \left\langle (E,F,\mapsto,\sharp,\rhd,\lambda,\mathsf{Act}), \mathsf{Init}, k\right\rangle \\
\{\!|\alpha[m].P|\!\}_I = & \left\langle \alpha(e).(E,F,\mapsto,\sharp,\rhd,\lambda,\mathsf{Act}), \mathsf{Init}', k'\right\rangle \text{ for } e \text{ fresh for } E \text{ where} \\
& \{\!|P|\!\}_I = \left\langle (E,F,\mapsto,\sharp,\rhd,\lambda,\mathsf{Act}), \mathsf{Init}, k\right\rangle \\
& \mathsf{Init}' = \mathsf{Init}\cup\{e\} \\
& k' = k\cup\{(e,m)\} \\
\{\!|P_0 \mid P_1|\!\}_I = & \left\langle (E,F,\mapsto,\sharp,\rhd,\lambda,\mathsf{Act}), \mathsf{Init}, k\right\rangle \text{ where} \\
& \text{For } i\in\{0,1\},\ \{\!|P_i|\!\}_I = \langle \mathcal{E}_i, \mathsf{Init}_i, k_i\rangle \\
& (E,F,\mapsto,\sharp,\rhd,\lambda,\mathsf{Act}) = \mathcal{E}_0||\mathcal{E}_1 \\
& \mathsf{Init} = \{(e_0,e_1) \mid e_0\in\mathsf{Init}_0, e_1\in\mathsf{Init}_1, k_0(e_0)=k_1(e_1)\}\cup \\
& \left\{(*,e_1)\,\middle|\, \begin{array}{l} e_1\in\mathsf{Init}_1 \text{ and} \\ \nexists e_0\in\mathsf{Init}_0.\lambda_0(e_0)=\overline{\lambda_1(e_1)} \text{ and } k_0(e_0)=k_1(e_1)\end{array}\right\}\cup \\
& \left\{(e_0,*)\,\middle|\, \begin{array}{l} e_0\in\mathsf{Init}_0 \text{ and} \\ \nexists e_1\in\mathsf{Init}_1.\lambda_0(e_0)=\overline{\lambda_1(e_1)} \text{ and } k_0(e_0)=k_1(e_1)\end{array}\right\} \\
& 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) \quad \textit{– note that } k_0(e_0)=k_1(e_1)\end{cases} \\
\{\!|P\setminus A|\!\}_I = & \left\langle \mathcal{E}\upharpoonright\{e\mid\lambda(e)\notin\mathcal{A}\}, \mathsf{Init}', k\upharpoonright\{e\mid\lambda(e)\notin\mathcal{A}\}\right\rangle \text{ where} \\
& \{\!|P|\!\}_I = \langle\mathcal{E},\mathsf{Init},k\rangle \\
& \mathsf{Init}' = \mathsf{Init}\cap\{e\mid\lambda(e)\notin\mathcal{A}\} \\
& \mathcal{A} = A\cup\overline{A} \\
\{\!|P[f]|\!\}_I = & \left\langle (E,F,\mapsto,\sharp,\rhd,\lambda,\mathsf{Act}), \mathsf{Init}, k\right\rangle \text{ where} \\
& \{\!|P|\!\}_I = \left\langle (E,F,\mapsto,\sharp,\rhd,\lambda',\mathsf{Act}'), \mathsf{Init}, k\right\rangle \\
& \mathsf{Act} = f(\mathsf{Act}') \\
& \lambda = f\circ\lambda' \\
\left\{\!\!\left| A\left\langle\tilde{b}\right\rangle\right|\!\!\right\}_0 = & \langle(\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\emptyset),\emptyset,\emptyset\rangle \\
\left\{\!\!\left| A\left\langle\tilde{b}\right\rangle\right|\!\!\right\}_I = & \left\{\!\!\left| P_A\{^{\tilde{b}}/_{\tilde{a}}\}\right|\!\!\right\}_{I-1} \text{ where } A(\tilde{a}) = P_A
\end{array}
$$

Table 2: RBES-semantics of CCSK.

1. $E = \bigcup_{n\in\omega} E_n$;
2. $F = \bigcup_{n\in\omega} F_n$;
3. $X \mapsto e^*$ *if for all* $n\in\omega$ *such that* $e\in E_n$, $(X\cap E_n)\mapsto e^*$;
4. $\sharp = \bigcup_{n\in\omega} \sharp_n$;
5. $\rhd = \bigcup_{n\in\omega} \rhd_n$;

6. $\lambda(e) = l$ *if there exists* $n \in \omega$ *such that* $\lambda_n(e) = l$;
7. $\mathsf{Act} = \bigcup_{n \in \omega} \mathsf{Act}_n$.

Proof. Clearly $\mathcal{E}$ is an LRBES, and for all $i \in \omega$, $\mathcal{E}_i \leq \mathcal{E}$. We therefore know that $\mathcal{E}$ is an upper bound of the chain. We now show that $\mathcal{E}$ is the least upper bound of the chain: Given an upper bound $\mathcal{E}'$, it is obvious that $E \subseteq E'$, $F = E \cap F'$, and if $X \mapsto e^*$ then, for all $n \in \omega$, if $e \in E_n$ then $X \cap E_n \mapsto_n e^*$, meaning since $\mathcal{E}_n \leq \mathcal{E}'$ there exists $X'_n \subseteq E'$ such that $X'_n \mapsto e^*$ and $X'_n \cap E_n = X_n$, and since for all $n' \geq n$, $\mathcal{E}_{n'} \leq \mathcal{E}'$ and $e \in E_{n'}$, $X'_n \cap E_{n'} \mapsto_{n'} e^*$. This means clearly $X'_n \cap E = \bigcup_{n \in \omega} X_n = X$. And for $e \in E$, if $X' \mapsto' e'^*$, then for all $n \in \omega$ such that $e \in E_n$ $X' \cap E_n \mapsto_n e^*$, meaning $X' \cap E \mapsto e^*$. Similar arguments apply to $\sharp$, $\rhd$, λ, and Act.

Therefore, clearly $\mathcal{E} \leq \mathcal{E}'$, and $\mathcal{E}$ is the least upper bound of the chain. □

This means that, given a set of events A, with $\mathbb{E}_A$ being the set of LRBESs $(E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$ such that $E \subseteq A$, we have that $(\mathbb{E}_A, \leq)$ is a complete partial order. We then need to show that our operations are monotonic.

Lemma 4.9.

1. *If* $\mathcal{E}_0 \leq \mathcal{E}_1$ *and* $E \subseteq E_1$, *then* $\mathcal{E}_0 \upharpoonright (E \cap E_0) \leq \mathcal{E}_1 \upharpoonright E$.
2. *For any* ω*-chain* $\mathcal{E}_0 \leq \mathcal{E}_1 \leq \mathcal{E}_2 \ldots$ *and* $E \subseteq \bigcup_{n \in \omega} E_n$, *we have* $(\bigsqcup_{n \in \omega} \mathcal{E}_n) \upharpoonright E = \bigsqcup_{n \in \omega} (\mathcal{E}_n \upharpoonright (E \cap E_n))$.

Proof. Straightforward from Definition 4.1. □

Lemma 4.10.

1. *Given LRBESs* $\mathcal{E}_0 \leq \mathcal{E}_1$ *and* $\mathcal{E}$, $\mathcal{E}_0 || \mathcal{E} \leq \mathcal{E}_1 || \mathcal{E}$.
2. *For any* ω*-chain* $\mathcal{E}_0 \leq \mathcal{E}_1 \leq \mathcal{E}_2 \ldots$ *and any LRBES* $\mathcal{E}$, *we have* $(\bigsqcup_{n \in \omega} \mathcal{E}_n) || \mathcal{E} = \bigsqcup_{n \in \omega} (\mathcal{E}_n || \mathcal{E})$.

Proof. Straightforward from Definition 4.2. □

Lemma 4.11.

1. *Given LRBESs* $\mathcal{E}_0 \leq \mathcal{E}_1$ *and* $\mathcal{E}$, $\mathcal{E}_0 + \mathcal{E} \leq \mathcal{E}_1 + \mathcal{E}$.
2. *For any* ω*-chain* $\mathcal{E}_0 \leq \mathcal{E}_1 \leq \mathcal{E}_2 \ldots$ *and LRBES* $\mathcal{E}$, *we have* $(\bigsqcup_{n \in \omega} \mathcal{E}_n) + \mathcal{E} = \bigsqcup_{n \in \omega} (\mathcal{E}_n + \mathcal{E})$.

Proof. Straightforward from Definition 4.3. □

Lemma 4.12.
1. *Given LRBESs* $\mathcal{E}_0 \leq \mathcal{E}_1$, *an event* $e \notin E_1$, *and a label* α, $\alpha(e).\mathcal{E}_0 \leq \alpha(e).\mathcal{E}_1$.
2. *For any* ω*-chain* $\mathcal{E}_0 \leq \mathcal{E}_1 \leq \mathcal{E}_2 \ldots$ *an event* $e \notin E_1$, *and a label* α, *we have* $\alpha(e).(\bigsqcup_{n \in \omega} \mathcal{E}_n) = \bigsqcup_{n \in \omega} (\alpha(e).\mathcal{E}_n)$.

Proof. Straightforward from Definition 4.5. □

Proposition 4.13 (Unfolding)**.** *Given a forwards reachable process P and a level of unfolding k, if $\{\!|P|\!\}_k = \langle \mathcal{E}, \mathsf{Init}, k\rangle$ and $\{\!|P|\!\}_{k-1} = \langle \mathcal{E}', \mathsf{Init}', k'\rangle$, then $\mathcal{E}' \leq \mathcal{E}$, $\mathsf{Init} = \mathsf{Init}'$, and $k = k'$.*

Proof. We have proved in Lemmas 4.9 to 4.12 that all the operations used for defining $\mathcal{E}$ and $\mathcal{E}'$ are monotonic, so clearly $\mathcal{E}' \leq \mathcal{E}$, and since P has been generated from a standard process, we cannot have any $\alpha[m]$ inside a recursion, as it would have to have been unfolded first. □

In order to prove that our event structure semantics corresponds with the operational semantics for CCSK defined in [28] we first show that event structures generated by our semantics are causal according to Definition 3.31.

Proposition 4.14. *Given a process P such that $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k\rangle$, $\mathcal{E}$ is causal.*

Proof. We say that $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$ and prove this by induction on P:

- Suppose $P = 0$. Then $\mathcal{E}$ is empty, and therefore obviously causal.
- Suppose $P = P_0 + P_1$, $\{\!|P_i|\!\} = \left\langle (E_i, F_i, \mapsto_i, \sharp_i, \rhd_i, \lambda_i, \mathsf{Act}_i), \mathsf{Init}_i, k_i \right\rangle$, $e \in E$ and $e' \in F$. Then if $e \rhd \underline{e'}$, then there exists $i \in \{0, 1\}$ such that either $e \rhd_i e'$ or $e \in E_i$ and $e' \in F_{1-i}$. By induction, $e \rhd_i e'$ this means there exists an $X_i \subseteq E_i$ such that $X_i \mapsto_i e$ and $e' \in X_i$. As $X_i \mapsto_i e$, we get $X_i \mapsto e$. And $E_i \times E_{1-i} \subseteq \sharp$.

  If there exists an $X \subseteq E$ such that $X \mapsto e$ and $e' \in X$, then there exists an $i \in \{0, 1\}$ such that $X \mapsto_i e$. Then by induction we get $e \rhd_i \underline{e'}$, implying $e \rhd \underline{e'}$.

  We have $X \mapsto \underline{e'}$ if and only if there exists an $i \in \{0, 1\}$ such that $X \mapsto_i \underline{e'}$. By induction, this means $e' \in X$.
- Suppose $P = \alpha.P'$, $\{\!|P'|\!\} = \left\langle (E', F', \mapsto', \sharp', \rhd', \lambda', \mathsf{Act}'), \mathsf{Init}', k' \right\rangle$, $e \in E$ and $e' \in F$. Then if $e \rhd \underline{e'}$ then either $e \rhd' \underline{e'}$, or $e' = e_\alpha$ and $e \in E'$. If $e \rhd' \underline{e'}$, then by induction there exists an $X \subseteq E'$ such that $X \mapsto' e$ and $e' \in X$, and $X \mapsto e$. If $e' = e_\alpha$ and $e \in E'$ then we know $\{e_\alpha\} \mapsto e$.

  If there exists an $X \subseteq E$ such that $X \mapsto e$ and $e' \in X$, then either $X \mapsto' e$, or $X = \{e_\alpha\}$ and $e \in E'$. If $X \mapsto' e$, then by induction we get $e \rhd' \underline{e'}$, and therefore $e \rhd \underline{e'}$. If $X = \{e_\alpha\}$ and $e \in E'$, then we know $e \rhd \underline{e_\alpha}$.

  We have $X \mapsto \underline{e'}$ if and only if $X \mapsto' \underline{e'}$ or $e' = e_\alpha$ and $X = \{e_\alpha\}$. By induction, if $X \mapsto' \underline{e'}$ then $e' \in X$.
- Suppose $P = \alpha[m].P'$. Then the proof is similar to the previous case.
- Suppose $P = P_0 \mid P_1$, $\{\!|P_i|\!\} = \left\langle (E_i, F_i, \mapsto_i, \sharp_i, \rhd_i, \lambda_i, \mathsf{Act}_i), \mathsf{Init}_i, k_i \right\rangle$, $e \in E$ and $e' \in F$. Then if $e \rhd \underline{e'}$, then there exists an $i \in \{0, 1\}$, such that $\pi_i(e) \rhd_i \pi_i(e')$. By induction, this means there exists an $X_i \subseteq E_i$ such that $X_i \mapsto_i \pi_i(e)$ and $\pi_i(e') \in X_i$. This means $\{e'' \in E \mid \pi_i(e'') \in X_i\} \mapsto e$, and obviously $e' \in \{e'' \in E \mid \pi_i(e'') \in X_i\}$.

If there exists an $X \subseteq E$ such that $X \mapsto e$ and $e' \in X$, then there exists an $i \in \{0,1\}$ and $X_i \subseteq E_i$ such that $X_i \mapsto_i \pi_i(e)$ and $X = \{e'' \in E \mid \pi_i(e'') \in X_i\}$, meaning $\pi_i(e') \in X_i$. By induction we get $\pi_i(e) \rhd_i \underline{\pi_i(e')}$, and therefore $e \rhd \underline{e'}$.

We have $X \mapsto \underline{e'}$ if and only if there exists $i \in \{0,1\}$ and $X_i \subseteq E_i$ such that $X_i \mapsto_i \pi_i(e')$ and $X = \{e'' \in E \mid \pi_i(e'') \in X_i\}$. By induction, since $X_i \mapsto_i \pi_i(e')$ we know $\pi_i(e') \in X_i$, meaning clearly $e' \in X$.

- Suppose $P = P' \setminus A$, $[\![P']\!] = \left\langle (E', F', \mapsto', \sharp', \rhd', \lambda', \mathsf{Act}'), \mathsf{Init}', k' \right\rangle$, $e \in E$ and $e' \in F$. Then $X \mapsto e'^*$ if and only if $X \mapsto' e'^*$ and $e \rhd \underline{e'}$ if and only if $e \rhd' \underline{e'}$. The rest of the case follows from induction.

- Suppose $P = P'[f]$. Then the result follows from induction. □

We then show that structurally congruent processes will generate isomorphic event structures.

Proposition 4.15 (Structural Congruence)**.** *Given processes P and P' such that $P \equiv P'$, $[\![P]\!] = \langle \mathcal{E}, \mathsf{Init}, k \rangle$, and $[\![P']\!] = \left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$, 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:

$P = P' \mid 0$**:** The function $f(e, *) = e$ fulfils the conditions.

$P = P_0 \mid P_1$ **and** $P' = P_1 \mid P_0$**:** 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.

$P = P_0 \mid (P_1 \mid P_2)$ **and** $P' = (P_0 \mid P_1) \mid P_2$**:** Products are associative up to isomorphism, and $f(e_0, (e_1, e_2)) = ((e_0, e_1), e_2)$ clearly fulfills the other conditions.

$P = P' + 0$**:** Clearly $\mathcal{E}_P = (\{0\} \times E', \{0\} \times F', \mapsto' \cup \emptyset, \sharp' \cup \emptyset, \rhd' \cup \emptyset, \lambda', \mathsf{Act}' \cup \emptyset)$, $\mathsf{Init} = \{0\} \times \mathsf{Init}'$, and $k = \{0\} \times k'$, meaning $f(0, e) = 0$.

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

$$f(e) = \begin{cases} (0, e') & \text{if } e = (0, (0, e')) \\ (1, (0, e')) & \text{if } e = (0, (1, e')) \\ (1, (1, e')) & \text{if } e = (1, e') \end{cases}$$

clearly fulfils the other conditions.

$P = A\left\langle \tilde{b} \right\rangle$ **and** $P' = P_A\{^{\tilde{b}}/_{\tilde{a}}\}$ **where** $A\langle \tilde{a} \rangle = P_A$**:** Follows from Proposition 4.13. □

Before we show our correspondence between actions in the process and in the event structure, we show that a reachable process has an empty initial state if and only if it does not contain any past action.

Lemma 4.16 (Standard)**.** *Given a process P such that $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k \rangle$ and there exists a standard process Q such that $Q \to^* P$, we have* $\mathsf{std}(P)$ *if and only if* $\mathsf{Init} = \emptyset$*.*

Proof. As the only rule which can add events to an empty Init is $\{\!|\alpha[m].P|\!\}$, clearly $\mathsf{Init} = \emptyset$ if $\mathsf{std}(P)$.

If $\mathsf{Init} = \emptyset$, then clearly we cannot have any $\alpha[m]$ in P, which are not guarded by a restriction on α. But if such a restricted communication has occurred in P, then there must exist a parallel $\overline{a}[m]$ inside the same restriction, meaning the corresponding event $(e_\alpha, e_{\overline{\alpha}})$ has the label τ, not α, and would therefore be in Init. Therefore we must have $\mathsf{std}(P)$. □

We now show that reachable processes have conflict-free initial states.

Lemma 4.17 (Reachable)**.** *If P is forwards-reachable and $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k \rangle$, then* Init *is conflict-free in $\mathcal{E}$.*

Proof. We show this by induction on P.

- Suppose $P = 0$. Then $\mathsf{Init} = \emptyset$.
- Suppose $P = \alpha.P'$ and $\{\!|P'|\!\} = \left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$. Then $\mathsf{Init} = \mathsf{Init}'$, and therefore Init is conflict-free.
- Suppose $P = \alpha[m].P'$ and $\{\!|P'|\!\} = \left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$. Then $\mathsf{Init} = \mathsf{Init}' \cup \{e_\alpha\}$, Init' is conflict-free, and therefore Init is clearly conflict-free.
- Suppose $P = P_1 + P_2$ and $\{\!|P_i|\!\} = \langle \mathcal{E}_i, \mathsf{Init}_i, k_i \rangle$. Then $\mathsf{Init} = (\{0\} \times \mathsf{Init}_1) \cup (\{1\} \times \mathsf{Init}_2)$ and, since P is reachable, either $\mathsf{Init}_1 = \emptyset$ or $\mathsf{Init}_2 = \emptyset$, and both Init_1 and Init_2 are conflict-free. Therefore, Init is conflict-free.
- Suppose $P = P_1 \mid P_2$ and $\{\!|P_i|\!\} = \left\langle (E_i, F_i, \mapsto_i, \sharp_i, \rhd_i, \lambda_i, \mathsf{Act}_i), \mathsf{Init}_i, k_i \right\rangle$. Then, since P is reachable from a standard process, each key appears at most once in P_1 and once in P_2. Additionally, Init_1 is conflict-free and Init_2 is conflict-free, meaning $\mathsf{Init} = \{(e_0, *) \mid e_0 \in \mathsf{Init}_0 \text{ and } \nexists e_1 \in \mathsf{Init}_1 . \lambda_0(e_0) = \overline{\lambda_1(e_1)} \text{ and } k_0(e_0) = k_1(e_1)\} \cup \{(*, e_1) \mid e_1 \in \mathsf{Init}_1 \text{ and } \nexists e_0 \in \mathsf{Init}_0 . \lambda_0(e_0) = \overline{\lambda_1(e_1)} \text{ and } k_0(e_0) = k_1(e_1)\} \cup \{(e_0, e_1) \mid e_0 \in \mathsf{Init}_0, e_1 \in \mathsf{Init}_1, k_0(e_0) = k_1(e_1)\}$ is conflict-free.
- Suppose $P = P' \setminus A$ and $\{\!|P'|\!\} = \left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$. Then Init' is conflict-free, and $\mathsf{Init} \subseteq \mathsf{Init}'$, meaning Init is conflict-free.
- Suppose $P = P'[f]$. Then $\mathsf{Init} = \mathsf{Init}'$, which is conflict-free. □

Finally we show in Theorems 4.18 and 4.19 that given a process P with a conflict-free initial state, including any reachable process, there exists a transition $P \xrightarrow{\mu} P'$ if and only if the event structure corresponding to P is isomorphic to the event structure corresponding to P' and an event e labelled μ exists such that e is available in P's initial state, and P''s initial state is P's initial state with e added.

Theorem 4.18. *If $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k\rangle$, $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$, $C_{br}(\mathcal{E}) = (E, F, \mathsf{C}, \rightarrow)$, Init is conflict-free, and there exists a transition $P \xrightarrow{\mu[m]} P'$ such that $\{\!|P'|\!\} = \left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$, then there exists an isomorphism $f : \mathcal{E} \to \mathcal{E}'$ and a transition $\mathsf{Init} \xrightarrow{\{e\}} X$ such that $\lambda(e) = \mu$, $f \circ k' = k \cup \{(e, m)\}$, and $f(X) = \mathsf{Init}'$.*

Proof. We prove this by induction on $P \xrightarrow{\mu[m]} P'$. The full proof can be seen in Appendix C.1 □

Having shown that each forwards transition in the operational semantics corresponds to one in the generated event structure, we now show the converse.

Theorem 4.19. *Let P be a reachable process. If $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k\rangle$, $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$, $C_{br}(\mathcal{E}) = (E, F, \mathsf{C}, \rightarrow)$, Init is conflict-free, and there exists a transition $\mathsf{Init} \xrightarrow{e} X$ in $C_{br}(\mathcal{E})$, then there exists a key m and a transition $P \xrightarrow{\lambda(e)[m]} P'$, such that $\{\!|P'|\!\} = \left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$ and there exists isomorphism $f : \mathcal{E} \to \mathcal{E}'$ such that $f \circ k' = k \cup \{(e, m)\}$ and $f(X) = \mathsf{Init}'$.*

Proof. We prove the theorem by induction on P in Appendix C.2. □

Corollary 4.20. *Given a process P such that $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k\rangle$, Init is forwards-reachable in $\mathcal{E}$ if and only if P is forwards-reachable.*

Since we showed in Proposition 4.14 that any event structures generated by processes are causal, it follows that we get a similar correspondence between the reverse transitions of processes and event structures.

Theorem 4.21. *Let P be a CCSK process. If $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k\rangle$, $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$, $C_{br}(\mathcal{E}) = (E, F, \mathsf{C}, \rightarrow)$, Init is conflict-free, and there exists a transition $P \overset{\mu[m]}{\rightsquigarrow} P'$ such that $\{\!|P'|\!\} = \left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$, then there exists isomorphism $f : \mathcal{E} \to \mathcal{E}'$ and a transition $\mathsf{Init} \xrightarrow{\{e\}} X$ such that $\lambda(e) = \mu$, $f \circ k' = k \cup \{(e, m)\}$, and $f(X) = \mathsf{Init}'$.*

Proof. Implied by Proposition 4.14, Theorem 4.18, and Corollary 4.20. □

Theorem 4.22. *Let P be a CCSK process. If $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k\rangle$, $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$, $C_{br}(\mathcal{E}) = (E, F, \mathsf{C}, \rightarrow)$, Init is conflict-free, and there exists a transition $\mathsf{Init} \xrightarrow{e} X$ in $C_{br}(\mathcal{E})$, then there exists a a key m and a transition $P \overset{\lambda(e)[m]}{\rightsquigarrow} P'$, such that $\{\!|P'|\!\} = \left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$ and there exists isomorphism $f : \mathcal{E} \to \mathcal{E}'$ such that $f \circ k' = k \cup \{(e, m)\}$ and $f(X) = \mathsf{Init}'$.*

Proof. Implied by Proposition 4.14, Theorem 4.19, and Corollary 4.20. □

We have now proved operational correspondence between the operational semantics of CCSK and the event structure semantics presented in this section. Proving this correspondence on reverse actions was made easy by both the calculus and the generated event structures having uncontrolled causal reversibility. However, uncontrolled reversibility comes with problems, in that it allows a process or event structure to do and undo the same action indefinitely. We would therefore like CCSK to have a way to control its reversibility.

5. Reversible Extended Bundle Event Structures

Suppose one wishes to model a program consisting of multiple parallel processes, but rather than allowing the process to do and undo actions whenever as in CCSK, it might be preferable to have one action that causes all actions, or all actions since the last safe state, to be reversed before the process can continue, similar to the roll command of [17]. RBESs can easily ensure that this roll event is required for other events to reverse. We simply say that $\{\mathsf{roll}\} \mapsto \underline{e}$ for all e, but preventing events from happening in RBESs requires symmetric conflict, which would mean the other events also prevent roll from occurring. To solve this problem in sequential processes, Phillips and Ulidowski [27] use reversible asymmetric event structures, which replace symmetric conflict with asymmetric. But since these use the same notion of causality as reversible prime event structures, they have trouble modelling concurrent processes with synchronisation, as shown in Example 1.1.

Extended bundle event structures (EBES) (Definition 5.2) add asymmetric conflict, similar to that of AESs to bundle event structures, and so defining a reversible variant of these will allow us to model the above scenario easily. In this section we therefore define a category of reversible bundle event structures, similar to the category of RBESs we defined in Section 3.

Example 5.1 (The necessity of REBESs for modelling rollback)**.** *Consider a process* $a.b \mid \overline{a}_\gamma.\mathsf{roll}\ \gamma$*, where* $\mathsf{roll}\ \gamma$ *means undo the action labelled* γ*, that is* $\overline{a}$*, and everything caused by it before continuing. To model this we would need to expand the RBES from Example 4.6 with a new event* $\mathsf{roll}\ \gamma$*, and split* b *into two different events depending on whether it needs to be reversed during the rollback or not. This would give us an RBES* $(\{a, \tau, \overline{a}, b_a, b_\tau, \mathsf{roll}\ \gamma\}, \{a, \tau, \overline{a}, b_a, b_\tau, \mathsf{roll}\ \gamma\}, \mapsto, \sharp, \rhd)$ *where* $\{a\} \mapsto b_a$*,* $\{\tau\} \mapsto b_\tau$*,* $\{\overline{a}, \tau\} \mapsto \mathsf{roll}\ \gamma$*,* $\{\mathsf{roll}\ \gamma\} \mapsto \underline{\tau}$*,* $\{\mathsf{roll}\ \gamma\} \mapsto \underline{\overline{a}}$*,* $\{\mathsf{roll}\ \gamma\} \mapsto \underline{b_\tau}$*,* $a \sharp \tau$*,* $\overline{a} \sharp \tau$*,* $b_a \rhd \underline{a}$*,* $b_\tau \rhd \underline{\tau}$*,* $\overline{a} \rhd \mathsf{roll}\ \gamma$*, and* $\tau \rhd \mathsf{roll}\ \gamma$*. This would indeed ensure that* $\overline{\overline{a}}$ *and the events caused by it could only reverse if one of the* roll *events had occurred, but it would not force them to do so before doing anything else. For this we use asymmetric conflict:* $\mathsf{roll}\ \gamma \rhd \overline{a}$*,* $\mathsf{roll}\ \gamma \rhd \tau$*,* $\mathsf{roll}\ \gamma \rhd b_\tau$*, giving us a CS with the* reachable configurations *shown in Figure 8.*

Definition 5.2 (Extended Bundle Event Structure [21]). An extended BES (EBES) is a triple $(E, \mapsto, \rhd)$ where:

1. E is the set of events;
2. $\mapsto \subseteq 2^E \times E$ is the bundle set, satisfying $X \mapsto e \Rightarrow \forall e_1, e_2 \in X.(e_1 \neq e_2 \Rightarrow e_1 \lhd e_2)$;

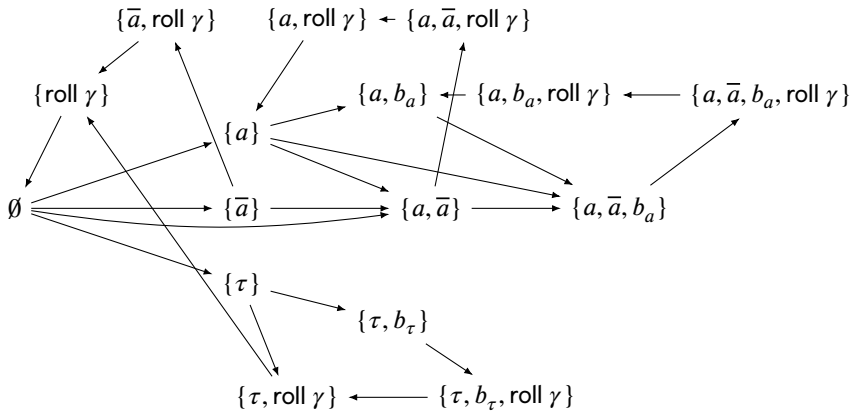


Figure 8: The reachable configurations of the REBES described in Example 5.1.

3. $\lhd \subseteq E \times E$ is the asymmetric conflict relation, which is irreflexive.

As we did for BESs, we define configurations, product, coproduct, and a finitely caused subcategory of EBESs. The full details can be found in Appendix D.

We again define a reversible version of EBESs in Definition 5.3, simply extending the asymmetric conflict and bundles to act on reversible events similarly to RBESs.

Definition 5.3 (Reversible Extended Bundle Event Structure). A reversible EBES (REBES) is a triple $\mathcal{E} = (E, F, \mapsto, \rhd)$ where:

1. E is the set of events;
2. $F \subseteq E$ is the set of reversible events;
3. $\mapsto \subseteq 2^E \times (E \cup \underline{F})$ is the bundle set, satisfying $X \mapsto e \Rightarrow \forall e_1, e_2 \in X.(e_1 \neq e_2 \Rightarrow e_1 \lhd e_2)$, and for all $e \in F$, $\{e\} \mapsto \underline{e}$;
4. $\lhd \subseteq (E \cup \underline{F}) \times E$ is the asymmetric conflict relation, which is irreflexive.

Unlike the forward-only general event structures, reversible general event structures (RESs) can model asymmetric conflict, by using their preventing set. This means that when we create a subcategory of finitely caused REBESs, they can be modelled by RESs.

Definition 5.4 (Finitely Caused Reversible Extended Bundle Event Structure). A finitely caused REBES (FCREBES) is an REBES $\mathcal{E} = (E, F, \mapsto, \rhd)$ where for any $e^* \in E \cup \underline{F}$, the set $\{X \subseteq E \mid X \mapsto e^*\}$ is finite.

Example 5.5 shows an REBES, which cannot be represented by an RBES, since one gets a transition $\emptyset \to \{a\}$, but no $\{b\} \to \{a, b\}$, despite $\{a, b\}$ being a configuration.

Example 5.5 (REBES)**.** *An REBES $\mathcal{E} = (E, F, \mapsto, \rhd)$ where $E = \{a, b, c\}$, $F = \{a, b\}$, $\{a, c\} \mapsto b$, $\{b\} \mapsto c$ $\{a\} \mapsto \underline{a}$, $\{b\} \mapsto \underline{a}$, $\{b\} \mapsto \underline{b}$, $a \rhd c$, $c \rhd a$, and $b \rhd a$ gives the CS $C_{er}(\mathcal{E})$ in Figure 9.*

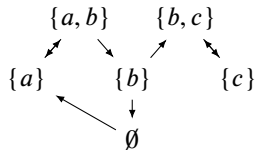


Figure 9: The configurations of the REBES described in Example 5.5.

In order to define REBES morphisms, we extend the RBES definition in the obvious way, as seen in Definition 5.6.

Definition 5.6 (REBES morphism). Given REBESs $\mathcal{E}_0 = (E_0, F_0, \mapsto_0, \rhd_0)$ and $\mathcal{E}_1 = (E_1, F_1, \mapsto_1, \rhd_1)$, an REBES morphism from $\mathcal{E}_0$ to $\mathcal{E}_1$ is a partial function $f : E_0 \to E_1$ such that $f(F_0) \subseteq F_1$ and for all $e, e' \in E_0$:

1. if $f(e) = f(e')$ and $e \neq e'$ then $e \sharp_0 e'$;
2. for $X_1 \subseteq E_1$ if $X_1 \mapsto_1 f(e)^*$ then there exists $X_0 \subseteq E_0$ such that $f(X_0) \subseteq X_1$, if $e' \in X_0$ then $f(e') \neq \bot$, and $X \mapsto_0 e^*$;
3. if $f(e) \lhd_1 f(e')^*$ then $e \lhd_0 \underline{e'}^*$.

We again construct a product of REBESs in Definition 5.7.

Definition 5.7 (REBES product). Given REBESs $\mathcal{E}_0 = (E_0, F_0, \mapsto_0, \rhd_0)$ and $\mathcal{E}_1 = (E_1, F_1, \mapsto_1, \rhd_1)$, we construct $\mathcal{E}_0 \times \mathcal{E}_1 = (E, F, \mapsto, \rhd)$ with projections π_0, π_1 where:

1. $E = E_0 \times_* E_1 = \{(e, *) \mid e \in E_0\} \cup \{(*, e) \mid e \in E_1\} \cup \{(e, e') \mid e \in E_0 \text{ and } e' \in E_1\}$;
2. $F = F_0 \times_* F_1 = \{(e, *) \mid e \in F_0\} \cup \{(*, e) \mid e \in F_1\} \cup \{(e, e') \mid e \in F_0 \text{ and } e' \in F_1\}$;
3. for $(e_0, e_1) \in E$, $\pi_i(e_0, e_1) = e_i$;
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 $e \in E$, $e'^* \in E \cup \underline{F}$, $e \rhd e'^*$ iff there exists $i \in \{0, 1\}$ such that $\pi_i(e) \lhd_i \pi_i(e')^*$.

Proposition 5.8. *Given REBESs $\mathcal{E}_0 = (E_0, F_0, \mapsto_0, \rhd_0)$ and $\mathcal{E}_1 = (E_1, F_1, \mapsto_1, \rhd_1)$, we have that $\mathcal{E}_0 \times \mathcal{E}_1 = (E, F, \mapsto, \rhd)$ is their product in the category **REBES**.*

Proof. Similar to the proof of Proposition D.7 seen in Appendix D. □

Proposition 5.9. *Given FCREBESs $\mathcal{E}_0 = (E_0, F_0, \mapsto_0, \rhd_0)$ and $\mathcal{E}_1 = (E_1, F_1, \mapsto_1, \rhd_1)$, we have that $\mathcal{E}_0 \times \mathcal{E}_1 = (E, F, \mapsto, \rhd)$ is an FCREBES.*

Proof. Similar to the proof of Proposition 3.11 seen in Appendix D. □

Definition 5.10 (REBES coproduct). Given REBESs $\mathcal{E}_0 = (E_0, F_0, \mapsto_0, \rhd_0)$ and $\mathcal{E}_1 = (E_1, F_1, \mapsto_1, \rhd_1)$, we construct $\mathcal{E}_0 + \mathcal{E}_1 = (E, F, \mapsto, \rhd)$ with injections ι_0, ι_1 where:

- $E = \{(0, e) \mid e \in E_0\} \cup \{(1, e) \mid e \in E_1\}$;

- $F = \{(0, e) \mid e \in F_0\} \cup \{(1, e) \mid e \in F_1\}$;
- for $e \in E_j$, $\iota_j(e) = (j, e)$ for $j \in \{0, 1\}$;
- $X \mapsto (j, e)^*$ iff for all $(j', e') \in X$, $j = j'$ and $\iota_j(X) \mapsto_j e^*$;
- $(j, e)^* \lhd (j', e')$ iff $j \neq j'$ or $e^* \lhd_j e'$.

Proposition 5.11. *If $\mathcal{E}_0$ and $\mathcal{E}_1$ are REBESs, then $\mathcal{E}_0 + \mathcal{E}_1$ is their coproduct in the category **REBES**.*

Proof. Similar to that of BES coproduct (Proposition 3.13 on Appendix D). □

Proposition 5.12. *Given FCREBESs $\mathcal{E}_0 = (E_0, F_0, \mapsto_0, \rhd_0)$ and $\mathcal{E}_1 = (E_1, F_1, \mapsto_1, \rhd_1)$, we have that $\mathcal{E}_0 + \mathcal{E}_1 = (E, F, \mapsto, \rhd)$ is an FCREBES.*

Proof. Similar to Proposition 3.27. □

We again model REBESs as CSs, defining a functor in Definition 5.13.

Definition 5.13 (From REBES to CS). The functor C_{er} : **REBES** $\to$ **CS** is defined as:

1. $C_{er}((E, F, \mapsto, \rhd)) = (E, F, \mathsf{C}, \to)$ where:
   (a) $X \in \mathsf{C}$ if $\lhd$ is well-founded on X;
   (b) For $X, Y \in \mathsf{C}$, $A \subseteq E$, and $B \subseteq F$, there exists a transition $X \xrightarrow{A \cup \underline{B}} Y$ if:
      i. $Y = (X \setminus B) \cup A$;
      ii. $X \cap A = \emptyset$;
      iii. $B \subseteq X$;
      iv. for all $e^* \in A \cup \underline{B}$, if $e' \rhd e^*$ then $e' \notin X \cup A$;
      v. for all $e \in A$ and $X' \subseteq E$, if $X' \mapsto e$ then $X' \cap (X \setminus B) \neq \emptyset$;
      vi. for all $e \in B$ and $X' \subseteq E$, if $X' \mapsto \underline{e}$ then $X' \cap (X \setminus (B \setminus \{e\})) \neq \emptyset$.
2. $C_{er}(f) = f$.

The definition of a cause-respecting and causal REBES (Definition 5.14) is of course practically identical to that of a crRBES and CRBES (Definition 3.31).

Definition 5.14 (Cause-respecting and Causal REBES). We say that $\mathcal{E} = (E, F, \mapsto, \rhd)$ is a cause-respecting REBES (crREBES) if whenever $X \mapsto e$ and $e' \in X$ or $e \rhd e'$, then $e \rhd \underline{e'}$.

We say that $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd, \lambda, \mathsf{Act})$ is a causal REBES (CREBES) if (1) if $e \rhd \underline{e'}$ then either $e \sharp e'$ or there exists an $X \subseteq E$ such that $X \mapsto e$ and $e' \in X$, (2) if $e \rhd e'$ or $X \mapsto e$ and $e' \in X \cap F$, then $e \rhd \underline{e'}$, and (3) if $X \mapsto \underline{e}$ then $e \in X$.

Proposition 5.15. *Given a CREBES, $\mathcal{E} = (E, F, \mapsto, \rhd)$ and corresponding CS $C_{re}(\mathcal{E}) = (E, F, \mathsf{C}, \to)$, any reachable $C \in \mathsf{C}$ is forwards-reachable.*

Proof. Similar to the corresponding proof for RBESs (Proposition 3.32). □

Proposition 5.16.

1. *If* $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd)$ *is a crREBES and* $C_{br}(\mathcal{E}) = (E, F, \mathsf{C}, \rightarrow)$ *then whenever* $X \in \mathsf{C}$ *is a reachable configuration,* $X \xrightarrow{\underline{B}} Y$, $Y \xrightarrow{B} X$
2. *If* $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd)$ *is a CREBES and* $C_{br}(\mathcal{E}) = (E, F, \mathsf{C}, \rightarrow)$ *then whenever* $X \in \mathsf{C}$, $X \xrightarrow{A \cup \underline{B}} Y$ *and* $A \cup B \subseteq F$, *there exists a transition* $Y \xrightarrow{B \cup \underline{A}} X$.

Proof. Similar to the corresponding proofs for RAESs [27] and RBESs (Proposition 3.33). □

We define a simple functor from **RBES** to **REBES**.

Definition 5.17 (RBES to REBES). The functor B_e : **RBES** $\rightarrow$ **REBES** is defined as:

1. $B_e((E, F, \mapsto, \rhd)) = (E, F, \mapsto, \lhd')$, where $e \lhd' e'$ if $e \sharp e'$ and $\underline{e} \lhd' e'$ if $e' \rhd \underline{e}$;
2. $B_e(f) = f$.

We define functors between **EBES** and **REBES**, which form an adjunction.

Definition 5.18 (EBES to REBES). The functor P_e : **EBES** $\rightarrow$ **REBES** is defined as:

1. $P_e((E, \mapsto, \rhd)) = (E, \emptyset, \mapsto, \lhd)$;
2. $P_e(f) = f$.

Definition 5.19 (REBES to EBES). The functor Φ_e : **REBES** $\rightarrow$ **EBES** is defined as:

1. $\Phi_e((E, F, \mapsto, \rhd)) = (E, \mapsto_E, \lhd_E)$;
2. $\Phi_e(f) = f$.

Proposition 5.20. $P_e \dashv \Phi_e$.

Proof. Similar to the proof of Proposition 3.37. □

As REBESs use asymmetric conflict, we create a functor from **RAES** to **REBES** in Definition E.3. As before, we also define a functor from **FCREBES** to **SRES** in Appendix E. Of course this causes the same problem as a functor from **RAES** to **SRES**, namely that a set of events X containing an infinite chain $e_1 \lhd e_2 \lhd \ldots$ does not have any finite subsets containing such an infinite chain, unless X also contains a finite $\lhd$-loop. This means that we may have some configurations of the generated SRES, which were not configurations of the original **FCREBES**.

Since we intend to use our REBESs for modelling the semantics of Rollback in CCSK, we need a labelled variant, much as we have of RBESs.

Definition 5.21 (Labelled Reversible Extended Bundle Event Structure). A labelled REBES (LREBES) $\mathcal{E} = (E, F, \mapsto, \rhd, \lambda, \mathsf{Act})$ consists of an REBES $(E, F, \mapsto, \rhd)$, a set of labels Act, and a surjective labelling function $\lambda : E \rightarrow \mathsf{Act}$.

Definition 5.22 (LREBES morphism). Given two labelled REBESs $\mathcal{E}_0 = (E_0, F_0, \mapsto_0, \sharp_0, \rhd_0, \lambda_0, \mathsf{Act}_0)$ and $\mathcal{E}_1 = (E_1, F_1, \mapsto_1, \sharp_1, \rhd_1, \lambda_1, \mathsf{Act}_1)$, an LREBES morphism $f : \mathcal{E}_0 \rightarrow \mathcal{E}_1$ is a partial function $f : E_0 \rightarrow E_1$ such that $f : (E_0, F_0, \mapsto_0, \rhd_0) \rightarrow (E_1, F_1, \mapsto_1, \rhd_1)$ is an REBES morphism and for all $e \in E_0$, either $f(e) = \perp$ or $\lambda_0(e) = \lambda_1(f(e))$.

6. Roll-CCSK

In this section we add a rollback operator, similar to that of roll-π [17], to CCSK. The semantics of roll-π is not directly translatable to CCSK, as it makes use of the fact that one can know, when looking at a memory, whether the communication it was associated with was with another process or not, and therefore, for a given subprocess P and a memory m, one knows whether all the memories and subprocesses caused by m are part of P. In CCSK, this is not the case, as the rollback in the subprocess $\alpha_\gamma[n].\mathsf{roll}\,\gamma$, where γ is a *tag* denoting which rollback rolls back which action, may or may not require rolling back the other end of the α communication, and all actions caused by it. For example, in the process $P = a_\gamma[n].\mathsf{roll}\,\gamma \mid b[m]$, rolling back γ does not involve reversing $b[m]$, but if we have a larger process, $\overline{a}[n].\overline{b}[m] \mid (a_\gamma[n].\mathsf{roll}\,\gamma \mid b[m])$, of which P is a subprocess, then $b[m]$ does need to get reversed. In roll-π, since all the memories are together and their tags make their causal relationships clear, this is not an issue. This makes it significantly harder to know when to stop rolling back, as (unless all the actions needing to be rolled back were τ-actions) we do not know that the rollback is complete. We therefore need to check at every instance of parallel composition whether any communication with actions being rolled back has taken place, and if so roll back those actions and all actions caused by them. This may include rolling back additional actions from the subprocess containing the rollback, as in $a[n_1].\overline{b}[n_2] \mid c[n_3].(\overline{a}_\gamma[n_1].\mathsf{roll}\,\gamma \mid b[n_2])$, where it does not become clear that $b[n_2]$ needs to be reversed during the rollback until the outer parallel composition. Additionally, roll-π uses asymmetric communication, meaning the memories that need to be reversed form a sequence, as opposed to one communication potentially causing more actions at both ends. This causes further problems when trying to define distributed semantics, which we do later in this section. To get around these problems, our distributed semantics marks all involved actions for rollback at once, but reverses them individually by using similar reversal rules to CCSK without rollback but only applying them to marked actions.

The syntax of Roll-CCSK is as follows:

$$\begin{aligned} P ::= \quad & \alpha_\gamma.P \mid \alpha_\gamma[n].P \mid P_0 + P_1 \mid P_0|P_1 \mid P \setminus A \\ & \mid P[f] \mid A\left\langle \tilde{b}, \tilde{\gamma} \right\rangle \mid 0 \mid \mathsf{roll}\,\gamma \mid \mathsf{rolling}\,\gamma \end{aligned}$$

Most of the syntax is the same as CCSK and CCS, but adding tags and rolls as described above, and $\mathsf{rolling}\,\gamma$, which denotes a rollback in progress, the necessity of which is justified later. From now on we will use $\alpha.P$ to denote $\alpha_\gamma.P$ where no $\mathsf{roll}\,\gamma$ exists in P. We also add a tuple of tags to our process constants, which get substituted just like the actions in a new structural congruence rule:

$$A\left\langle \tilde{b}, \tilde{\delta} \right\rangle \equiv P_A\{{}^{\tilde{b}}/_{\tilde{a}}\, {}^{\tilde{\delta}}/_{\tilde{\gamma}}\} \text{ if } A(\tilde{a}, \tilde{\gamma}) = P_A$$

Before defining our semantics of rollback, we first define causal dependence and projection similarly to [17], which we will use to define our semantics.

Definition 6.1 (Causal dependence). Let P be a process. Then the binary relation $\leq_P$ is the smallest relation on $\mathsf{keys}(P)$ satisfying:

- If there exists a process P' and past actions $\alpha_\gamma[n]$ and $\beta_{\gamma'}[m]$ such that $\alpha_\gamma[n].P'$ is a subprocess of P and $\beta_{\gamma'}[m]$ occurs in P' then $n \leq_P m$.
- $\leq_P$ is reflexively and transitively closed.

Definition 6.2 (Projection). Given a process P and a set of keys C, $P_{\natural C}$ is defined as:

$(\alpha_\gamma[n].P)_{\natural C} = \alpha_\gamma[n].(P_{\natural C})$ if $n \notin C$
$(\alpha_\gamma[n].P)_{\natural C} = \alpha_\gamma.(P_{\natural C}\{^{\mathsf{roll}\ \gamma}/_{\mathsf{rolling}\ \gamma}\})$ if $n \in C$
$\mathsf{roll}\ \gamma_{\natural C} = \mathsf{roll}\ \gamma$
$(P_0 \mid P_1)_{\natural C} = P_{0_{\natural C}} \mid P_{1_{\natural C}}$
$(P[f])_{\natural C} = P_{\natural C}[f]$
$(\alpha_\gamma.P)_{\natural C} = \alpha_\gamma.(P_{\natural C})$
$0_{\natural C} = 0$
$(P \setminus A)_{\natural C} = (P_{\natural C}) \setminus A$
$\mathsf{rolling}\ \gamma_{\natural C} = \mathsf{rolling}\ \gamma$
$A\left\langle \tilde{b}, \tilde{\gamma} \right\rangle_{\natural C} = A\left\langle \tilde{b}, \tilde{\gamma} \right\rangle$
$(P_0 + P_1)_{\natural C} = P_{0_{\natural C}} + P_{1_{\natural C}}$

Much as in [17] we carry out our rollback in two steps, the first triggering the rollback, and the second actually performing the roll, in order to ensure that we can start multiple rollbacks at the same time. For example, in the process $(a_\gamma.(\overline{d}.0 \mid c.\mathsf{roll}\ \gamma) \mid b_{\gamma'}.(\overline{c} \mid d.\mathsf{roll}\ \gamma') \mid \overline{a} \mid \overline{b}) \setminus \{a, b, c, d\}$, similar to an example from [17], we will otherwise never be able to roll all the way back to the beginning, as rolling back a_γ will stop us from reaching $\mathsf{roll}\ \gamma'$ and vice versa. Tables 3 and 4 show that new rules for reversing actions in Roll-CCSK. The rules for forward actions are still the same as in Table 1.

Example 6.3. *Consider the process* $(a_\gamma[n_a].(\overline{d}[n_d].0 \mid c[n_c].\mathsf{roll}\ \gamma) \mid b_{\gamma'}[n_b].(\overline{c}[n_c] \mid d[n_d].\mathsf{roll}\ \gamma') \mid \overline{a}[n_a] \mid \overline{b}[n_b]) \setminus \{a, b, c, d\}$. *We would like to roll this process back to its initial state, but performing one of the rolls would disable the other, since it would reverse the communications on* c *and* d. *We therefore say that we first need to trigger the rolls,*

$$a_\gamma[n_a].(\overline{d}[n_d].0 \mid c[n_c].\mathsf{roll}\ \gamma) \mid b_{\gamma'}[n_b].(\overline{c}[n_c] \mid d[n_d].\mathsf{roll}\ \gamma') \mid \overline{a}[n_a] \mid \overline{b}[n_b]) \setminus \{a, b, c, d\}$$
$$\xrightarrow{\mathsf{start\ roll}\ \gamma\ \ \mathsf{start\ roll}\ \gamma'}$$
$$(a_\gamma[n_a].(\overline{d}[n_d].0 \mid c[n_c].\mathsf{rolling}\ \gamma) \mid b_{\gamma'}[n_b].(\overline{c}[n_c] \mid d[n_d].\mathsf{rolling}\ \gamma') \mid \overline{a}[n_a] \mid \overline{b}[n_b]) \setminus \{a, b, c, d\}$$

meaning that the rolls can now be executed even if the preceding actions have been reversed, so we can do

$$a_\gamma[n_a].(\overline{d}[n_d].0 \mid c[n_c].\mathsf{rolling}\ \gamma) \mid b_{\gamma'}[n_b].(\overline{c}[n_c] \mid d[n_d].\mathsf{rolling}\ \gamma') \mid \overline{a}[n_a] \mid \overline{b}[n_b]) \setminus \{a, b, c, d\}$$
$$\xrightarrow{\mathsf{roll}\ n_b} (a_\gamma[n_a].(\overline{d}.0 \mid c.\mathsf{rolling}\ \gamma) \mid b_{\gamma'}.(\overline{c} \mid d.\mathsf{roll}\ \gamma') \mid \overline{a}[n_a] \mid \overline{b}) \setminus \{a, b, c, d\}$$

Since $\mathsf{rolling}\ \gamma$ *has been triggered it can now be executed despite* c *having been reversed,*

$$a_\gamma[n_a].(\overline{d}.0 \mid c.\mathsf{rolling}\ \gamma) \mid b_{\gamma'}.(\overline{c} \mid d.\mathsf{roll}\ \gamma') \mid \overline{a}[n_a] \mid \overline{b}) \setminus \{a, b, c, d\}$$
$$\xrightarrow{\mathsf{roll}\ n_a} (a_\gamma.(\overline{d}.0 \mid c.\mathsf{roll}\ \gamma) \mid b_{\gamma'}.(\overline{c} \mid d.\mathsf{roll}\ \gamma') \mid \overline{a} \mid \overline{b}) \setminus \{a, b, c, d\}$$

In addition, to ensure every rollback is associated with exactly one action, we define a *consistent process*.

(start ROLL) $\mathsf{roll}\ \gamma \overset{\mathsf{start\ roll}\ \gamma}{\rightsquigarrow} \mathsf{rolling}\ \gamma$ (par ROLL) $\dfrac{P_0 \overset{\mathsf{roll}\ n}{\rightsquigarrow} P_0' \quad C = \{m \mid n \leq_{P_0'\mid P_1} m\}}{P_0 \mid P_1 \overset{\mathsf{roll}\ n}{\rightsquigarrow} (P_0 \mid P_1)_{\not\downarrow C}}$

(ROLL) $\mathsf{rolling}\ \gamma \overset{\mathsf{roll}\ \gamma}{\rightsquigarrow} \mathsf{roll}\ \gamma$ (act ROLL) $\dfrac{P \overset{\mathsf{roll}\ \gamma}{\rightsquigarrow} P' \quad C = \{m \mid n \leq_{\alpha_\gamma[n].P} m\}}{\alpha_\gamma[n].P \overset{\mathsf{roll}\ n}{\rightsquigarrow} \alpha_\gamma.P'_{\not\downarrow C}}$

Table 3: The main rules for rollback in the operational semantics of Roll-CCSK.

Definition 6.4 (Consistent process). A roll-CCSK process P is consistent if

1. there exists a standard process Q with no subprocess $\mathsf{rolling}\ \gamma$ such that $Q \to^* P$;
2. any $\mathsf{roll}\ \gamma$ or $\mathsf{rolling}\ \gamma$ in P is part of a subprocess, $\alpha_\gamma.P'$ or $\alpha_\gamma[n].P'$;
3. any subprocess of P, $\alpha_\gamma.P'$ or $\alpha_\gamma[n].P'$ contains at most one instance of $\mathsf{roll}\ \gamma$ or $\mathsf{rolling}\ \gamma$ not part of a subprocess of P', $\alpha'_\gamma.P''$ or $\alpha'_\gamma[n].P''$;
4. if $A\langle \tilde{a}, \tilde{\gamma}\rangle$ is a subprocess of P defined as $A(\tilde{b}, \tilde{\delta}) = P_A$, then P_A is consistent.

Proposition 6.5. *Let P be a consistent process and $P \equiv P'$. Then P' is consistent.*

Proof. The tags and definitions of process constants will not change between P and P'. □

Proposition 6.6. *Let P be a consistent process and $P \xrightarrow{\mu[n]} P'$. Then P' is consistent.*

Proof. The tags and definitions of process constants will not change between P and P'. □

Example 6.7 (Recursion)**.** *Consider the process $a_\gamma.A\langle a;\gamma\rangle$ where we have a recursive definition $A(b;\delta) = b_\delta.(A\langle b;\delta\rangle \mid \mathsf{roll}\ \delta)$. This process is consistent and we show that we do not have problems of confusing which actions are supposed to be reversed by which instance of* $\mathsf{roll}\ \gamma$*. After doing some actions and unfoldings we get*

$$a_\gamma[n_0].a_\gamma[n_1].(a_\gamma[n_2].(A\langle a;\gamma\rangle \mid \mathsf{roll}\ \gamma) \mid \mathsf{rolling}\ \gamma)$$

At this point, if we execute the outer $\mathsf{rolling}\ \gamma$ *we will get*

$$\dfrac{\dfrac{\dfrac{}{\mathsf{rolling}\ \gamma \overset{\mathsf{roll}\ \gamma}{\rightsquigarrow} \mathsf{roll}\ \gamma}}{(a_\gamma[n_2].(A\langle a;\gamma\rangle \mid \mathsf{roll}\ \gamma) \mid \mathsf{rolling}\ \gamma) \overset{\mathsf{roll}\ \gamma}{\rightsquigarrow} (a_\gamma[n_2].(A\langle a;\gamma\rangle \mid \mathsf{roll}\ \gamma) \mid \mathsf{roll}\ \gamma)}}{a_\gamma[n_1].(a_\gamma[n_2].(A\langle a;\gamma\rangle \mid \mathsf{roll}\ \gamma) \mid \mathsf{rolling}\ \gamma) \overset{\mathsf{roll}\ n_1}{\rightsquigarrow} a_\gamma.(a_\gamma.(A\langle a;\gamma\rangle \mid \mathsf{roll}\ \gamma) \mid \mathsf{roll}\ \gamma)}$$

$$a_\gamma[n_0].a_\gamma[n_1].(a_\gamma[n_2].(A\langle a;\gamma\rangle \mid \mathsf{roll}\ \gamma) \mid \mathsf{rolling}\ \gamma) \overset{\mathsf{roll}\ n_1}{\rightsquigarrow} a_\gamma[n_0].a_\gamma.(a_\gamma.(A\langle a;\gamma\rangle \mid \mathsf{roll}\ \gamma) \mid \mathsf{roll}\ \gamma)$$

And we see that switching the rollback from the tag to the key once we reach the first γ-tagged action means that is the action we roll back to.

(prop ROLL 1)	$\dfrac{P \overset{\text{roll } \gamma}{\rightsquigarrow} P' \quad \gamma \neq \gamma'}{\beta_{\gamma'}[m].P \overset{\text{roll } \gamma}{\rightsquigarrow} \beta_{\gamma'}[m].P'}$	(prop ROLL start 1)	$\dfrac{P \overset{\text{start roll } \gamma}{\rightsquigarrow} P'}{\alpha_{\gamma'}[n].P \overset{\text{start roll } \gamma}{\rightsquigarrow} \alpha_{\gamma'}[n].P'}$
(prop ROLL 2)	$\dfrac{P \overset{\text{roll } \gamma}{\rightsquigarrow} P'}{\beta_{\gamma'}.P \overset{\text{roll } \gamma}{\rightsquigarrow} \beta.\gamma'.P'}$	(prop ROLL start 2)	$\dfrac{P_0 \overset{\text{start roll } \gamma}{\rightsquigarrow} P_0' \quad \mathsf{std}(P_1)}{P_0 + P_1 \overset{\text{start roll } \gamma}{\rightsquigarrow} P_0' + P_1}$
(prop ROLL 3)	$\dfrac{P_0 \overset{\text{roll } \gamma}{\rightsquigarrow} P_0'}{P_0 + P_1 \overset{\text{roll } \gamma}{\rightsquigarrow} P_0' + P_1}$	(prop ROLL start 3)	$\dfrac{P_0 \overset{\text{start roll } \gamma}{\rightsquigarrow} P_0'}{P_0 \mid P_1 \overset{\text{start roll } \gamma}{\rightsquigarrow} P_0' \mid P_1}$
(prop ROLL 4)	$\dfrac{P \overset{\text{roll } \gamma}{\rightsquigarrow} P'}{P \setminus A \overset{\text{roll } \gamma}{\rightsquigarrow} P' \setminus A}$	(prop ROLL start 4)	$\dfrac{P \to P'}{P \setminus A \overset{\text{start roll } \gamma}{\rightsquigarrow} P' \setminus A}$
(prop ROLL 5)	$\dfrac{P \overset{\text{roll } \gamma}{\rightsquigarrow} P'}{P[f] \overset{\text{roll } \gamma}{\rightsquigarrow} P'[f]}$	(prop ROLL start 5)	$\dfrac{P \overset{\text{start roll } \gamma}{\rightsquigarrow} P'}{P[f] \overset{\text{start roll } \gamma}{\rightsquigarrow} P'[f]}$
(prop ROLL 6)	$\dfrac{P \equiv Q \overset{\text{roll } \gamma}{\rightsquigarrow} Q' \equiv P'}{P \overset{\text{roll } \gamma}{\rightsquigarrow} P'}$	(prop ROLL start 6)	$\dfrac{P \equiv Q \overset{\text{start roll } \gamma}{\rightsquigarrow} Q' \equiv P'}{P \overset{\text{start roll } \gamma}{\rightsquigarrow} P'}$
(prop ROLL Key 1)	$\dfrac{P \overset{\text{roll } n}{\rightsquigarrow} P' \quad n \neq m}{\beta_\gamma[m].P \overset{\text{roll } n}{\rightsquigarrow} \beta_\gamma[m].P'}$	(prop ROLL Key 2)	$\dfrac{P \overset{\text{roll } n}{\rightsquigarrow} P'}{\beta_\gamma.P \overset{\text{roll } n}{\rightsquigarrow} \beta.\gamma.P'}$
(prop ROLL Key 3)	$\dfrac{P_0 \overset{\text{roll } n}{\rightsquigarrow} P_0'}{P_0 + P_1 \overset{\text{roll } n}{\rightsquigarrow} P_0' + P_1}$	(prop ROLL Key 4)	$\dfrac{P \overset{\text{roll } n}{\rightsquigarrow} P'}{P \setminus A \overset{\text{roll } n}{\rightsquigarrow} P' \setminus A}$
(prop ROLL Key 5)	$\dfrac{P \overset{\text{roll } n}{\rightsquigarrow} P'}{P[f] \overset{\text{roll } n}{\rightsquigarrow} P'[f]}$	(prop ROLL Key 6)	$\dfrac{P \equiv Q \overset{\text{roll } n}{\rightsquigarrow} Q' \equiv P'}{P \overset{\text{roll } n}{\rightsquigarrow} P'}$

Table 4: The operational semantics for propagation of rolls in Roll-CCSK.

We are then ready to prove Theorem 6.9, stating that for consistent subprocesses, any rollback can be undone by a sequence of forwards actions. For this we use Lemma 6.8, which states that projecting on an upwards-closed set of keys always results in a possible previous state of the process.

Lemma 6.8. *Given a consistent process P with no subprocess* rolling γ *and a set of keys C such that if $n \in C$ and $n \leq_P m$ then $m \in C$, we have a sequence of transitions $P_{\not\downarrow C} \to^* P$.*

Proof. We prove this by induction on the size of C.

Suppose $C = \emptyset$. Then $P_{\not\downarrow C} = P$.

Suppose $P_{\not\downarrow C'} \to^* P$ and $C = C' \cup \{n\}$ for some n such that if $m \leq_P n$ then $m \notin C'$. Then if there does not exist an action α and a tag γ such that $\alpha_\gamma[n]$ occurs in P, we get $P_{\not\downarrow C} = P_{\not\downarrow C'} \to^* P$. If there exists a process P', an action α and tag γ such that $\alpha_\gamma[n].P'$ is a subprocess of P then all past actions of P' are in C', meaning $P'_{\not\downarrow C} = \mathsf{rt}(P')$, and since $m \leq_P n \Rightarrow m \notin C'$, we get $P_{\not\downarrow C} \xrightarrow{\alpha_\gamma[n]} P_{\not\downarrow C'} \to^* P$. □

Theorem 6.9 (Loop (Soundness))**.** *Given consistent processes P_0 and P_1 containing no subprocesses* rolling γ*, such that $P_0 \overset{\text{start roll } \gamma}{\rightsquigarrow} P'_0 \overset{\text{roll n}}{\rightsquigarrow} P_1$, we have $P_1 \to^* P_0$.*

Proof. Follows from Lemma 6.8. □

We will from now on use $\to_{CCSK}$ and $\rightsquigarrow_{CCSK}$ to distinguish CCSK-transitions from roll-CCSK transitions. The last thing we need to prove about our rollback operational semantics before moving on to event structure semantics is Theorem 6.12, stating that (1) our rollbacks only reverse the actions caused by the action we are rolling back according to CCSK, and (2) our rollbacks are maximally permissive, meaning that any subset of reached rollbacks may be successfully executed.

In order to define our notion of completeness, we first need a way to translate roll-CCSK to CCSK.

Definition 6.10 (Transforming roll-CCSK to CCSK). We define a function, ϕ, which translates a roll-CCSK process into CCSK:

$\phi(0) = 0$	$\phi(P_0 + P_1) = \phi(P_0) + \phi(P_1)$	$\phi(P_0 \mid P_1) = \phi(P_0) \mid \phi(P_1)$
$\phi(\alpha_\gamma.P) = \alpha.\phi(P)$	$\phi(\alpha_\gamma[n].P) = \alpha[n].\phi(P)$	$\phi(P \setminus A) = \phi(P) \setminus A$
$\phi(P[f]) = \phi(P)[f]$	$\phi(\text{roll } \gamma) = 0$	

Definition 6.11 (Reversing Upto Keys). Let P be a CCSK process and $T = \{m_0, m_1, \dots m_n\}$ be a set of keys. We say that $P \rightsquigarrow_T P'$ if there exists actions μ, ν and a key m such that $P \overset{\mu[m]}{\rightsquigarrow}_{CCSK} P'$ and $\nu[m_i] \leq_P \mu[m]$ for some $m_i \in T$.

Theorem 6.12 (Completeness)**.** *Let P be a consistent roll-CCSK process with subprocesses with rolls, $\alpha_{0_{\gamma_0}}[m_0] \dots$ roll γ_0, $\alpha_{1_{\gamma_1}}[m_1] \dots$ roll γ_1, $\dots$, $\alpha_{n_{\gamma_n}}[m_n] \dots$ roll γ_n. Then for all $T \subseteq \{m_0, m_1, \dots m_n\}$, if $\phi(P) \rightsquigarrow_T^* P' \not\rightsquigarrow_T$ then there exists a roll-CCSK process P'' such that $\phi(P'') = P'$ and $P \rightsquigarrow^* P''$.*

(start ROLL) $\mathsf{roll}\ \gamma \overset{\text{start roll } \gamma}{\rightsquigarrow}_{\text{SM}} \mathsf{rolling}\ \gamma$ (par ROLL) $\dfrac{P_0 \overset{\text{roll } n}{\rightsquigarrow}_{\text{SM}} P_0' \quad C = \{m \mid n \leq_{P_0 \mid P_1} m\}}{P_0 \mid P_1 \overset{\text{roll } n}{\rightsquigarrow}_{\text{SM}} \mathsf{mark}_C(P_0 \mid P_1)}$

(ROLL) $\mathsf{rolling}\ \gamma \overset{\text{roll } \gamma}{\rightsquigarrow}_{\text{SM}} \mathsf{roll}\ \gamma$ (act ROLL) $\dfrac{P \overset{\text{roll } \gamma}{\rightsquigarrow}_{\text{SM}} P' \quad C = \{m \mid n \leq_{\alpha_\gamma[n].P} m\}}{\alpha_\gamma[n].P \overset{\text{roll } n}{\rightsquigarrow}_{\text{SM}} \alpha_\gamma[n]^\bullet.\mathsf{mark}_C(P')}$

Table 5: The main rules for semi-distributed rollback of Roll-CCSK.

Proof. To get P'' we apply first $\overset{\text{start roll } \gamma_i}{\rightsquigarrow}$ for every $m_i \in T$ and then $\overset{\text{roll } m_i}{\rightsquigarrow}$ for every $m_i \in T$ to P. We show that this is the correct P''.

Let P_r be P with every $\mathsf{rolling}\ \gamma$ replaced with $\mathsf{roll}\ \gamma$. Then $P_r \rightsquigarrow^* P_r''$, where P_r'' is P'' with every $\mathsf{rolling}\ \gamma$ replaced with $\mathsf{roll}\ \gamma$ using the same rules as $P \rightsquigarrow^* P''$.

By the loop theorem we get $P_r'' \rightarrow^* P_r$. And since $\phi(P_r) = \phi(P)$ and $\phi(P_r'') = \phi(P'')$, we can translate this computation into CCSK: $\phi(P'') \rightarrow^*_{CCSK} \phi(P)$. From the loop lemma of CCSK, this gives us $\phi(P) \rightsquigarrow^*_{CCSK} \phi(P'')$. And obviously $\phi(P'') \not\rightsquigarrow_T$.

We then only need to show that if $\phi(P) \rightsquigarrow^*_T P' \not\rightsquigarrow_T$, then $P' = \phi(P'')$. Since they both reverse all the keys causally dependent on keys in T, this follows from Proposition 5.16 of [26]. □

With Theorems 6.9 and 6.12, we have shown that our semantics captures the behaviour of a rollback, but we would like to be able to roll back the actions one at a time in a distributed manner. For this purpose we define a new semantics, in which the rollback marks the actions needing to be rolled back with the key associated with the rollback, so they can be rolled back individually. We refer to this as single-mark semantics and annotate its transitions with SM; we call the semantics described above high-level semantics, which we annotate with HL.

To mark the actions we define a new function, mark, similar to $\not\downarrow$, which marks all the actions with a specific set of keys, C, with $\bullet$.

Definition 6.13 (Marking Function). Given a process P and a set of keys C, $\mathsf{mark}^\bullet_C(P)$ is defined as:

$\mathsf{mark}_C(\alpha_\gamma[m].P) = \alpha_\gamma[m].\mathsf{mark}_C(P)$ if $m \notin C$ $\quad \mathsf{mark}_C(0) = 0$

$\mathsf{mark}_C(P \setminus A) = \mathsf{mark}_C(P) \setminus A$ $\quad \mathsf{mark}_C(\mathsf{roll}\ \gamma) = \mathsf{roll}\ \gamma$

$\mathsf{mark}_C(\alpha_\gamma[m].P) = \alpha_\gamma[m]^\bullet.\mathsf{mark}_C(P)$ if $m \in C$ $\quad \mathsf{mark}_C(\mathsf{rolling}\ \gamma) = \mathsf{rolling}\ \gamma$

$\mathsf{mark}_C(P_0 \mid P_1) = \mathsf{mark}_C(P_0) \mid \mathsf{mark}_C(P_1)$ $\quad \mathsf{mark}_C(A\left\langle \tilde{b}, \tilde{\gamma} \right\rangle) = A\left\langle \tilde{b}, \tilde{\gamma} \right\rangle$

$\mathsf{mark}_C(P_0 + P_1) = \mathsf{mark}_C(P_0) + \mathsf{mark}_C(P_1)$ $\quad \mathsf{mark}_C(\alpha_\gamma.P) = \alpha_\gamma.\mathsf{mark}_C(P)$

$\mathsf{mark}_C(P[f]) = \mathsf{mark}_C(P)[f]$

We call a process containing actions marked with $\bullet$ a *marked* process.

Note that this marking is not defined on already marked actions, meaning all markings must be removed before a new rollback can start.

The forwards rules are still the same, though note that a forwards action cannot propagate past marked actions. The roll rules (Tables 4 and 5) are the same as in the previous semantics, with the exception of replacing instances of $P_{\not\downarrow C}$ with $\mathsf{mark}_C(P)$

$$\frac{\mathsf{std}(P)}{\alpha[n]^\bullet.P \overset{\alpha[n]}{\rightsquigarrow}_{\mathrm{SM}} \alpha.P} \qquad \frac{P \overset{\mu[m]}{\rightsquigarrow}_{\mathrm{SM}} P' \quad m \neq n}{\alpha[n].P \overset{\mu[m]}{\rightsquigarrow}_{\mathrm{SM}} \alpha[n].P'} \qquad \frac{P \overset{\mu[m]}{\rightsquigarrow}_{\mathrm{SM}} P' \quad m \neq n}{\alpha[n]^\bullet.P \overset{\mu[m]}{\rightsquigarrow}_{\mathrm{SM}} \alpha[n]^\bullet.P'}$$

$$\frac{P \equiv Q \overset{\mu[n]}{\rightsquigarrow}_{\mathrm{SM}} Q' \equiv P'}{P \overset{\mu[n]}{\rightsquigarrow}_{\mathrm{SM}} P'} \qquad \frac{P_0 \overset{\mu[n]}{\rightsquigarrow}_{\mathrm{SM}} P_0' \quad \mathsf{fsh}[n](P_1)}{P_0 \mid P_1 \overset{\mu[n]}{\rightsquigarrow}_{\mathrm{SM}} P_0' \mid P_1} \qquad \frac{P_0 \overset{\alpha[n]}{\rightsquigarrow}_{\mathrm{SM}} P_0' \quad P_1 \overset{\overline{\alpha}[n]}{\rightsquigarrow}_{\mathrm{SM}} P_1'}{P_0 \mid P_1 \overset{\tau[n]}{\rightsquigarrow}_{\mathrm{SM}} P_0' \mid P_1'}$$

$$\frac{P_0 \overset{\mu[n]}{\rightsquigarrow}_{\mathrm{SM}} P_0' \quad \mathsf{std}(P_1)}{P_0 + P_1 \overset{\mu[n]}{\rightsquigarrow}_{\mathrm{SM}} P_0' + P_1} \qquad \frac{P \overset{\mu[n]}{\rightsquigarrow}_{\mathrm{SM}} P' \quad \mu, \overline{\mu} \notin A}{P \setminus A \overset{\mu[n]}{\rightsquigarrow}_{\mathrm{SM}} P' \setminus A} \qquad \frac{P \overset{\mu[n]}{\rightsquigarrow}_{\mathrm{SM}} P'}{P[f] \overset{f(\mu)[n]}{\rightsquigarrow}_{\mathrm{SM}} P'[f]}$$

Table 6: Reversing marked actions.

and letting start roll propagate past marked actions. For actually reversing the marked actions, we introduce the rules seen in Table 6, which invert the forwards rules, but must start with a marked action.

Example 6.14. *Consider the process $a_\gamma[n].b[m].\mathsf{rolling}\ \gamma \mid (\overline{b}[m] \mid \overline{a}[n])$ in which the a's and b's have communicated and the rollback has been activated. In order to actually execute the roll, we mark the actions needing to be rolled back with n:*

$$a_\gamma[n].b[m].\mathsf{rolling}\ \gamma \mid (\overline{b}[m] \mid \overline{a}[n]) \overset{\mathsf{roll}\ n}{\rightsquigarrow}_{\mathrm{SM}} a_\gamma[n]^\bullet.b[m]^\bullet.\mathsf{roll}\ \gamma \mid (\overline{b}[m]^\bullet \mid \overline{a}[n]^\bullet)$$

We can then reverse the marked actions in a causal way:

$$\begin{array}{ll} & a_\gamma[n]^\bullet.b[m]^\bullet.\mathsf{roll}\ \gamma \mid (\overline{b}[m]^\bullet \mid \overline{a}[n]^\bullet) \\ \xrightarrow{\tau[m]}_{\mathrm{SM}} & a_\gamma[n]^\bullet.b.\mathsf{roll}\ \gamma \mid (\overline{b} \mid \overline{a}[n]^\bullet) \\ \xrightarrow{\tau[n]}_{\mathrm{SM}} & a_\gamma.b.\mathsf{roll}\ \gamma \mid (\overline{b} \mid \overline{a}) \end{array}$$

We intend to show that this new semantics is equivalent to the previous. For this purpose we define a notion of *reverse-ignoring bisimulation* which, as the name suggests ignores all reverse transitions except rolls.

Definition 6.15 (Reverse-ignoring Bisimulation). A relation $\mathcal{R}$ on processes from single-mark and high-level semantics, is a reverse-ignoring bisimulation if whenever $P\ \mathcal{R} Q$:

1. If $P \xrightarrow{\mu[n]}_{\mathrm{SM}} P'$ then $Q \xleftarrow{\mu[n]}_{\mathrm{HL}} Q'$ and $P' \mathcal{R} Q'$.
2. If $Q \xrightarrow{\mu[n]}_{\mathrm{HL}} Q'$ then $P \xleftarrow{\mu[n]}_{\mathrm{SM}} P'$ and $P' \mathcal{R} Q'$.
3. For $l \in \{\mathsf{roll}\ n \mid n \text{ is a key}\}$, if $P \overset{l}{\rightsquigarrow}_{\mathrm{SM}} P'$ then $Q \overset{l}{\hookleftarrow}_{\mathrm{HL}} Q'$ and $P' \mathcal{R} Q'$.
4. For $l \in \{\mathsf{roll}\ n \mid n \text{ is a key}\}$, if $Q \overset{l}{\rightsquigarrow}_{\mathrm{HL}} Q'$ then $P \overset{l}{\hookleftarrow}_{\mathrm{SM}} P'$ and $P' \mathcal{R} Q'$.

Where $P \overset{l}{\hookleftarrow} P'$ if $P \overset{l_1}{\rightsquigarrow} \cdots \overset{l_i}{\rightsquigarrow} \overset{l}{\rightarrow} \overset{l'_1}{\rightsquigarrow} \cdots \overset{l'_j}{\rightsquigarrow} P'$ or $P \overset{l_1}{\rightsquigarrow} \cdots \overset{l_i}{\rightsquigarrow} \overset{l}{\rightarrow} \overset{l'_1}{\rightsquigarrow} \cdots \overset{l'_j}{\rightsquigarrow} P'$ for $\{l_1, \ldots, l_i, l'_1, \ldots, l'_j\} \cap (\{\mathsf{roll}\ n \mid n \text{ is a key}\}) = \emptyset$.

For processes P and Q, if there exists a reverse-ignoring bisimulation $\mathcal{R}$ such that $P \mathcal{R} Q$ then $P \approx_{\mathrm{SM}}^{\mathrm{HL}} Q$.

We also define a mapping, M, which removes the $\bullet$ and key from all marked actions, effectively reversing them, which we will use to describe our bisimulation relation between single-mark and high-level processes.

Definition 6.16 (Mapping from single-mark to high-level semantics). Given a reachable process of the single-mark semantics, P, we define a mapping to a high-level process without markings:

$$
\begin{array}{ll}
M(\alpha_\gamma[n].P) = \alpha_\gamma[n].M(P) & M(0) = 0 \\
M(P \setminus A) = M(P) \setminus A & M(\mathsf{roll}\ \gamma) = \mathsf{roll}\ \gamma \\
M(\mathsf{rolling}\ \gamma) = \mathsf{rolling}\ \gamma & M(P_0 \mid P_1) = M(P_0) \mid M(P_1) \\
M(A\left\langle \tilde{b}, \tilde{\gamma} \right\rangle) = A\left\langle \tilde{b}, \tilde{\gamma} \right\rangle & \\
M(\alpha_\gamma[n]^\bullet.P) = \alpha_\gamma.M(P) & M(P_0 + P_1) = M(P_0) + M(P_1) \\
M(\alpha_\gamma.P) = \alpha_\gamma.M(P) & M(P[f]) = M(P)[f]
\end{array}
$$

Using those, we get a correspondence between the two semantics, as expressed in Theorem 6.17.

Theorem 6.17. *Given a process P reachable with the single-mark semantics, $P \approx_{\mathrm{HL}} M(P)$.*

Proof. We define

$$\mathcal{R} = \{(P, M(P)) \mid P \text{ is reachable } \}$$

and show that this is a reverse-ignoring bisimulation.

We do this by induction on the structure of P, and it is obvious in all cases except $P = \alpha_\gamma[n]^\bullet.P'$. In this case P can only perform reverse non-rollback actions $P \overset{\mu[m]}{\rightsquigarrow}_{\mathrm{SM}} Q$, but $M(P)$ can perform $M(P) \xrightarrow{\alpha_\gamma[n']}_{\mathrm{HL}} R$. Fortunately P can perform a series of transitions $P \overset{\mu_1[m_1]}{\rightsquigarrow}_{\mathrm{SM}} \cdots \overset{\mu_i[m_i]}{\rightsquigarrow}_{\mathrm{SM}} \overset{\alpha[n]}{\rightsquigarrow}_{\mathrm{SM}} M(P) \xrightarrow{\alpha_\gamma[n']}_{\mathrm{SM}} R$. □

7. Event Structure Semantics of Roll-CCSK

Now that we have defined operational semantics of rollback, we are ready to describe a more formal method for using LREBESs to model rollback than the ad-hoc approach we used in Example 5.1.

To model a rollback in an event structure, we have two events, one which triggers the rollback, labelled $\mathsf{start\ roll}\ \gamma$, and another which starts the actual rollback by allowing the events caused by the associated action to begin reversing. When prefixing a process P with an action α_γ, we now need to ensure that any action in P, and any $\mathsf{start\ roll}$ associated with such an action, will be reversed by any $\mathsf{roll}\ \gamma$ in P, and that the rollback does not stop, signified by the event labelled $\mathsf{roll}\ \gamma$ being reversed, until those actions have all been reversed.

When composing the LREBESs of two processes, we also create a separate event for each set of causes it might have (Definition 7.1), similarly to how products of PESs are handled or how we define enablings when converting to an RES. This allows us to say that an event can be rolled back if it was caused by a communication with one of the

events being rolled back, but not if the communication went differently. Consider the process $a_\gamma.\mathsf{roll}\ \gamma \mid \overline{a}.b \mid a_{\gamma'}.\mathsf{roll}\ \gamma'$. In this case we will want b to roll back if both (a_γ, a) and $\mathsf{roll}\ \gamma$ have happened, or if both $(a_{\gamma'}, a)$ and $\mathsf{roll}\ \gamma'$ have happened, but not if any other combination of the four events has happened, something which bundles cannot express unless b is split into multiple events. In addition, we use the sets of causes to ensure that if e is in e''s set of causes and e_{roll} can cause e to reverse, then e_{roll} can cause e' to reverse.

Definition 7.1 (Possible Causes). Given an LREBES, $\mathcal{E} = (E, F, \mapsto, \rhd, \lambda, \mathsf{Act})$, the set of possible causes for an event $e \in E$, $\mathsf{cause}(e) = X$, contains minimal sets of events such that if $x \in X$ then:

1. if $x' \mapsto e$ then there exists e' such that $x' \cap x = \{e'\}$;
2. if $e' \in x$ then there exists $x' \in \mathsf{cause}(e')$ such that $x' \subseteq x$;
3. if $e_1, e_0 \in X$ then not $e_0 \sharp e_1$.

Example 7.2 (Causes and parallel composition)**.** *Consider the process* $P = a_\gamma.\mathsf{roll}\ \gamma \mid \overline{a}_{\gamma'}.\mathsf{roll}\ \gamma'$. *This process would generate the event structure* $\{\!|P|\!\} = \langle \mathcal{E}, \emptyset, \emptyset \rangle$ *with events:*

$$\begin{array}{lllll} \{(\emptyset, (a, *)), & (\emptyset, (*, \overline{a})), & (\emptyset, (a, \overline{a})), & (\{(a, *)\}, & (\mathsf{start\ roll}\ \gamma, *)), \\ (\{(a, \overline{a})\}, & (\mathsf{start\ roll}\ \gamma, *)), & (\{(*, \overline{a})\}, & (*, \mathsf{start\ roll}\ \gamma')), & (\{(a, \overline{a})\}, \\ (*, \mathsf{start\ roll}\ \gamma')), & (\mathsf{roll}\ \gamma, *), & (*, \mathsf{roll}\ \gamma')\} & & \end{array}$$

Here all events but the rolls have a set of causes associated with them but are still labelled based only on the action.

The actions are not preceded by other actions, so their sets of causes are empty and they can happen at the start of the process.

The start roll*-labelled events are all caused by their preceding actions either happening on their own or synchronising, and whether they synchronise or not determines which of the* start roll *events happens.*

The roll *events do not have sets of causes, since we do not treat them as forwards actions needing to be reversed before the actions causing them are reversed.*

As a side effect of adding causes, we also need to change the definition of restriction to remove not only the actions associated with the restricted labels, but also the actions caused by them. We do this because we want the event structures generated by P and $0 \mid P$ to always be isomorphic, but if $P = (a.b) \setminus \{a\}$, we will otherwise get an event b, which, having no possible causes, disappears once we put P in parallel with anything, since this involves generating a b event for each set of possible causes.

Definition 7.3 (Removing labels and their dependants). Given an event structure $\mathcal{E} = (E, F, \mapsto, \rhd, \lambda, \mathsf{Act})$ and a set of labels $A \subseteq \mathsf{Act}$, we define $\rho(A) = X$ as the maximum subset of E such that

1. if $e \in X$ then $\lambda(e) \notin A$;
2. if $e \in X$ then there exists $x \in \mathsf{cause}(e)$ such that $x \subseteq X$.

We are now ready to define event structure semantics of Roll-CCSK in Table 7. Both $\mathsf{roll}\ \gamma$ and $\mathsf{rolling}\ \gamma$ generate the same event structure consisting of a start rollback event e_s and a rollback event e_r, but with the e_s in the initial state of $\mathsf{rolling}\ \gamma$. A process with an action prefix, $\alpha_\gamma.P$ has to find any $\mathsf{roll}\ \gamma$ events rolling back to α_γ, R, and any events not part of a potential roll back to α_γ, E_{roll}. To define E_{roll}, we look for whether tags occur in P. This is not strictly necessary, as we can tell for a given rollback event whether P contains its corresponding action by seeing whether it is caused by any action events (as we do when defining R). We then say that all non-rollback events in P are caused by e_α and reversed by the rolls in R. And that the rolls in R are prevented from reversing by e_α. After adding causes to action events, a parallel composition, $P_0 \mid P_1$, has to figure out which rolls are associated with which of the new events with causes. To do this, we need to find the rolls associated with either half of either the original event or one of its causes. The rest of the event structure semantics is similar to that of Section 4.

Table 7: LREBES-semantics of Roll-CCSK.

$\{\!|\mathsf{roll}\ \gamma|\!\}_l =$ $\langle(\{e_r, e_s\}, \{e_r, e_s\}, \mapsto, \rhd, \lambda, \mathsf{Act}), \emptyset, \emptyset\rangle$ where:
$\{e_r\} \mapsto \underline{e_r}$ $\{e_s\} \mapsto \underline{e_s}$ $\{e_s\} \mapsto e_r$, and $\{e_r\} \mapsto \underline{e_s}$
$e_s \rhd \underline{e_r}$ and $e_r \rhd e_s$
$\lambda(e) = \begin{cases} \mathsf{roll}\ \gamma & \text{if } e = e_r \\ \mathsf{start\ roll}\ \gamma & \text{if } e = e_s \end{cases}$ $\mathsf{Act} = \{\mathsf{roll}\ \gamma, \mathsf{start\ roll}\ \gamma\}$

$\{\!|\mathsf{rolling}\ \gamma|\!\}_l =$ $\langle(\{e_r, e_s\}, \{e_r, e_s\}, \mapsto, \rhd, \lambda, \mathsf{Act}), \{e_s\}, \emptyset\rangle$ where:
$\{e_r\} \mapsto \underline{e_r}$ $\{e_s\} \mapsto \underline{e_s}$ $\{e_s\} \mapsto e_r$, and $\{e_r\} \mapsto \underline{e_s}$
$e_s \rhd \underline{e_r}$ and $e_r \rhd e_s$
$\lambda(e) = \begin{cases} \mathsf{roll}\ \gamma & \text{if } e = e_r \\ \mathsf{start\ roll}\ \gamma & \text{if } e = e_s \end{cases}$ $\mathsf{Act} = \{\mathsf{roll}\ \gamma, \mathsf{start\ roll}\ \gamma\}$

$\{\!|0|\!\}_l =$ $\langle(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset), \emptyset, \emptyset\rangle$

$\{\!|\alpha_\gamma.P|\!\}_l =$ $\langle(E, F, \mapsto, \rhd, \lambda, \mathsf{Act}), \mathsf{Init}, k\rangle$ where:
$\{\!|P|\!\}_l = \langle(E_P, F_P, \mapsto_P, \rhd_P, \lambda_P, \mathsf{Act}_P), \mathsf{Init}, k\rangle$
$E = E_P \cup \{e_\alpha\}$ where e_α fresh
$E_{\mathsf{roll}} = \left\{ e \,\middle|\, \begin{array}{l} \lambda_P(e) \in \{\mathsf{roll}\ \gamma' \mid \gamma' \text{ is a tag } \} \text{ or} \\ \lambda_P(e) \in \{\mathsf{start\ roll}\ \gamma' \mid \nexists \beta, n.\beta_{\gamma'} \text{ or } \beta_{\gamma'}[n] \text{ occurs in } \alpha_\gamma.P\} \end{array} \right\}$
$F = F_P \cup \{e_\alpha\}$
$X \mapsto e$ if $X \mapsto_P e$ or $X = \{e_\alpha\}$, $e \in E_P$, and $\lambda_P(e) \neq \mathsf{roll}\ \gamma'$
We define the set of roll events rolling back to α_γ as:
$R = \{e \mid \lambda_P(e) = \mathsf{roll}\ \gamma \text{ and } e' \rhd_P \underline{e} \Rightarrow \lambda_P(e') = \mathsf{start\ roll}\ \gamma\}$
$X \mapsto \underline{e}$ if $X = \{e\}$, or $e = e_\alpha$ and $X = R$, or $e \in E_{\mathsf{roll}}$ and $X \mapsto_P \underline{e}$, or $e \notin E_{\mathsf{roll}}$, $\{e\} \neq X' \mapsto_P \underline{e}$, and $X = X' \cup R$
$\rhd = \rhd_P \cup ((E \setminus \{e_r \mid \exists \gamma'.\lambda(e_r) \in \{\mathsf{roll}\ \gamma', \mathsf{start\ roll}\ \gamma'\}\}) \times \{\underline{e_\alpha}\}) \cup$
$(\{e_\alpha\} \times \underline{R}) \cup (R \times (E \setminus R))$
$\mathsf{Act} = \mathsf{Act}_P \cup \{\alpha\}$
For all $e \in E$, $\lambda(e) = \begin{cases} \lambda_P(e) & \text{if } e \in E_P \\ \alpha & \text{if } e = e_\alpha \end{cases}$

Table 7: LREBES-semantics of Roll-CCSK (continued).

$\{\!|\alpha_\gamma[m].P|\!\}_l =$ $\langle (E, F, \mapsto, \rhd, \lambda, \mathsf{Act}), \mathsf{Init}, k\rangle$ where:
$\{\!|\alpha_\gamma.P|\!\}_l = \left\langle (E, F, \mapsto, \rhd, \lambda, \mathsf{Act}), \mathsf{Init}', k'\right\rangle$
$\{e_\alpha\} = \{e \in E \mid \lambda(a) = \alpha \text{ and } \nexists X \subseteq E.X \mapsto e_\alpha\}$
$\mathsf{Init} = \mathsf{Init}' \cup \{e_\alpha\} \qquad k(e) = \begin{cases} m & \text{if } e = e_\alpha \\ k'(e) & \text{otherwise} \end{cases}$

$\left\{\!\left|A\left\langle \tilde{b}, \tilde{\delta}\right\rangle\right|\!\right\}_0 =$ $\langle (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset), \emptyset, \emptyset\rangle$

$\left\{\!\left|A\left\langle \tilde{b}, \tilde{\delta}\right\rangle\right|\!\right\}_l =$ $\left\{\!\left|P_A\{^{\tilde{b},\tilde{\delta}}/_{\tilde{a},\tilde{\gamma}}\}\right|\!\right\}_{l-1}$ where $A\langle \tilde{a}, \tilde{\gamma}\rangle = P_A$ and $l \geq 1$

$\{\!|P_0 + P_1|\!\}_l =$ $\langle \mathcal{E}_0 + \mathcal{E}_1, \mathsf{Init}, k\rangle$ where
For $i \in \{0, 1\}$, $\{\!|P_i|\!\}_l = \langle \mathcal{E}_i, \mathsf{Init}_i, k_i\rangle$
$\mathsf{Init} = \{(j, e) \mid j \in \{0, 1\} \text{ and } e \in \mathsf{Init}_j\}$
$k(j, e) = k_j(e)$ if $e \in \mathsf{Init}_j$

$\{\!|P_0 \mid P_1|\!\}_l =$ $\langle (E, F, \mapsto, \rhd, \lambda, \mathsf{Act}), \mathsf{Init}, k\rangle$ where:
$\{\!|P_i|\!\}_l = \langle \mathcal{E}_i, \mathsf{Init}_i, k_i\rangle$ and $\mathcal{E}_i = (E_i, F_i, \mapsto_i, \rhd_i, \lambda_i, \mathsf{Act}_i)$ for $i \in \{0, 1\}$
$(E_\times, F_\times, \mapsto_\times, \rhd_\times, \lambda_\times, \mathsf{Act}_\times) = \mathcal{E}_0 || \mathcal{E}_1$
$\mathsf{Init}_\times = \{(e_0, e_1) \mid e_0 \in \mathsf{Init}_0, e_1 \in \mathsf{Init}_1, k_0(e_0) = k_1(e_1)\} \cup$
$\{(*, e_1) \mid e_1 \in \mathsf{Init}_1, \nexists e_0 \in \mathsf{Init}_0.\lambda_0(e_0) = \underline{\lambda_1(e_1)}, \text{ and } k_0(e_0) = k_1(e_1)\} \cup$
$\{(e_0, *) \mid e_0 \in \mathsf{Init}_0, \nexists e_1 \in \mathsf{Init}_1.\lambda_0(e_0) = \underline{\lambda_1(e_1)}, \text{ and } k_0(e_0) = k_1(e_1)\}$
$E_{\mathsf{action}} = \left\{ (X, e) \middle| \begin{array}{c} e \in E_\times, \lambda_\times(e) \notin \{\mathsf{roll}\ \gamma \mid \gamma \text{ is a tag }\}, X \in \mathsf{cause}(e) \\ \text{and } \forall e' \in X.\exists X' \in \mathsf{cause}(e').X' \subseteq X \end{array} \right\}$
$E_{\mathsf{roll}} = \{e \mid e \in E_\times \text{ and } \lambda_\times(e) \in \{\mathsf{roll}\ \gamma \mid \gamma \text{ is a tag }\}\}$
$E = E_{\mathsf{action}} \cup E_{\mathsf{roll}}$
$F_{\mathsf{action}} = \{(X, e) \in E \mid e \in F_\times\} \quad F_{\mathsf{roll}} = E_{\mathsf{roll}} \cap F_\times \quad F = F_{\mathsf{action}} \cup F_{\mathsf{roll}}$
We define π_0 and π_1 such that for $(X, (e_0, e_1)) \in E_a$, $\pi_i(X, (e_0, e_1)) = e_i$,
and for $(e_0, e_1) \in E_r$, $\pi_i(e_0, e_1) = e_i$
$\{(X, e') \mid X' \subseteq X\} \mapsto (X', (e_0, e_1))$ if $e' \in X'$
$X \mapsto (e_0, e_1)$ if there exists X' such that $X' \mapsto_\times e$ and
$X = \{e' \mid (\pi_0(e'), \pi_1(e')) \in X'\}$
$X \mapsto \underline{e}$ if $X = \{e\}$, or
$e = (e_0, e_1)$ and $X = \{e' \mid (\pi_0(e'), \pi_1(e')) \in X'\}$ for some $X' \mapsto_\times \underline{e}$, or
$e = (X', e_\times)$ and $X = \bigcup \left\{ X'' \middle| \begin{array}{l} \exists i \in \{0, 1\}, X_i \in E_i.X_i \mapsto \underline{\pi_i(e_\times)} \\ \text{or } \exists e'_\times \in X'.X_i \mapsto \pi_i(\underline{e'_\times}) \\ \text{and } e' \in X'' \text{ iff } \pi_i(\underline{e') \in X_i} \end{array} \right\}$
$e \rhd e'^*$ if there exists $i \in \{0, 1\}$ such that $\pi_i(e) \rhd_i \pi_i(e')^*$, or
$e \neq e'$, and $e \in X \mapsto \underline{e'}$, or
$\pi_i(e) = \pi_i(e') \neq \bot$, and $e \neq e'$, $e'^* = e'$, or $e'^* \neq e'$ and $e, e' \in E_{\mathsf{roll}}$
$\mathsf{Act} = \mathsf{Act}_0 \cup \mathsf{Act}_1 \cup \{\tau\}$
$\lambda(e) = \begin{cases} \tau & \text{if } e = (X, (e_0, e_1)) \\ \lambda_0(e_0) & \text{if } e = (X, (e_0, *)) \text{ or } e = (e_0, *) \\ \lambda_1(e_1) & \text{if } e = (X, (*, e_1)) \text{ or } e = (*, e_1) \end{cases}$
$\mathsf{Init} = \{(X, e) \mid X \cup \{e\} \subseteq \mathsf{Init}_\times\} \cup (E_{\mathsf{roll}} \cap \mathsf{Init}_\times)$

Table 7: LREBES-semantics of Roll-CCSK (continued).

$$k(e) = \begin{cases} k_0(e_0) & \text{if } e = (X, (e_0, *)) \\ k_1(e_1) & \text{if } e = (X, (*, e_1)) \\ k_0(e_0) & \text{if } e = (X, (e_0, e_1)) \quad \textit{– note that } k_0(e_0) = k_1(e_1) \end{cases}$$

$$\{\!|\, P \setminus A \,|\!\}_l = \left\langle \mathcal{E} \upharpoonright \rho(A \cup \overline{A}), \mathsf{Init} \cap \rho(A \cup \overline{A}), k \upharpoonright \rho(A \cup \overline{A}) \right\rangle \text{ where } \{\!|\, P \,|\!\}_l = \langle \mathcal{E}, \mathsf{Init}, k \rangle$$

Much like we did in Proposition 4.13, we need to show that there exists a least upper bound of the event structures resulting from unfolding recursion. For this we first show that our action prefix, parallel composition, and tag binding are monotonic.

Proposition 7.4 (Unfolding)**.** *Given a consistent process P and a level of unfolding k, if $\{\!|\, P \,|\!\}_k = \langle \mathcal{E}, \mathsf{Init}, k \rangle$ and $\{\!|\, P \,|\!\}_{k-1} = \left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$, then $\mathcal{E}' \leq \mathcal{E}$, $\mathsf{Init} = \mathsf{Init}'$, and $k = k'$.*

Proof. Follows from Lemmas F.1, F.2, and F.3 in Appendix F.1 and Proposition 4.13. □

Example 7.5 (Recursion)**.** *Consider the process $P = a_\gamma.A\langle a; \gamma \rangle$ with the recursive definition $A(b; \delta) = b_\delta.(A\langle b; \delta \rangle \mid \mathsf{roll}\ \delta)$ from Example 6.7. Since we have recursion $\{\!|\, P \,|\!\}$ would have an infinite number of events, but we can still get a reasonable idea of what it would look like by only unfolding twice, giving us $\{\!|\, P \,|\!\}_2 = \langle (E, F, \mapsto, \rhd), \emptyset, \emptyset \rangle$. We name events after their labels and their level of unfolding so that e.g. $\mathsf{roll}\ \gamma{:}1$ is the $\mathsf{roll}\ \gamma$-labelled event originating from the first unfolding of the recursion.*

$E = F = \{a{:}0, a{:}1, a{:}2, \mathsf{start\ roll}\ \gamma{:}1, \mathsf{start\ roll}\ \gamma{:}2, \mathsf{roll}\ \gamma{:}1, \mathsf{roll}\ \gamma{:}2\}$

$\{a{:}i\} \mapsto a{:}j$ *for* $i < j$	$\mathsf{roll}\ \gamma{:}i \rhd a{:}j$ *for* $i \leq j$
$\{a{:}i\} \mapsto \mathsf{start\ roll}\ \gamma{:}j$ *for* $i \leq j$	$\mathsf{roll}\ \gamma{:}i \rhd \mathsf{start\ roll}\ \gamma{:}j$ *for* $i \leq j$
$\{\mathsf{start\ roll}\ \gamma{:}i\} \mapsto \mathsf{roll}\ \gamma{:}i$ *for* $i \in \{1, 2\}$	$\mathsf{roll}\ \gamma{:}i \rhd \mathsf{roll}\ \gamma{:}j$ *for* $i \neq j$
$\{\mathsf{roll}\ \gamma{:}i\} \mapsto \underline{\mathsf{start\ roll}\ \gamma{:}i}$ *for* $i \in \{1, 2\}$	$a{:}i \rhd \underline{a{:}j}$ *for* $i > j$
$\{\mathsf{roll}\ \gamma{:}i\} \mapsto \underline{a{:}j}$ *for* $i \leq j$	$a{:}i \rhd \underline{\mathsf{roll}\ \gamma{:}i}$ *for* $i \in \{0, 1\}$
$\lambda(\mu{:}i) = \mu$	$\mathsf{start\ roll}\ \gamma{:}i \rhd \underline{\mathsf{roll}\ \gamma{:}i}$ *for* $i \in \{1, 2\}$
$\mathsf{Act} = \{a, \mathsf{start\ roll}\ \gamma\}$	

Obviously this can be extended to a greater level of unfolding by allowing $i > 2$.

In this case we deal with the issue of ensuring each rollback only rolls back to its most recent a_γ by using the set $\{e_r \mid \lambda_P(e_r) = \mathsf{roll}\ \gamma$ and $e_r \in X \mapsto e \Rightarrow \lambda(e) = \mathsf{start\ roll}\ \gamma\}$ as the set of rollback events rolling back to α_γ in the definition of $\{\!|\, \alpha_\gamma.P \,|\!\}$. This ensures that if the rollback event has found another γ-tagged action in P then it will be causing that to reverse and therefore not be in the set.

We then show that structurally congruent processes result in isomorphic event structures, for which we use Lemmas 7.6 to 7.11.

We first show that aside from rolls, we have causal consistency between events.

Lemma 7.6 (Causal consistency of actions)**.** *Given Roll-CCSK process P such that $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k\rangle$, for any events e, e', if $e \in X \mapsto e'$ and $e' \not\trianglerighteq \underline{e}$, then there exists γ such that $\lambda(e') \in \{\mathsf{roll}\ \gamma, \mathsf{start\ roll}\ \gamma\}$.*

Proof. We prove this by structural induction on P in Appendix F.2. □

We then show that, aside from rolls, causation is transitive.

Lemma 7.7 (Transitive causation)**.** *Given a Roll-CCSK process P such that $\{\!|P|\!\} = \langle (E, F, \mapsto, \rhd, \lambda, \mathsf{Act}), \mathsf{Init}, k\rangle$, whenever $X \mapsto e \in X' \mapsto e'$, we have $X \mapsto e'$, or there exists a γ such that $\lambda(e') = \mathsf{roll}\ \gamma$.*

Proof. We prove this by structural induction on P in Appendix F.3. □

We then show that because we split actions into multiple events, each bundle associated with an action event only contains one event.

Lemma 7.8 (Forwards bundles)**.** *Given a Roll-CCSK process P such that $\{\!|P|\!\} = \langle (E, F, \mapsto, \rhd, \lambda, \mathsf{Act}), \mathsf{Init}, k\rangle$, whenever $X \mapsto e$, either there exists e' such that $X = \{e'\}$, or there exists a γ such that $\lambda(e) = \mathsf{roll}\ \gamma$.*

Proof. We prove this by structural induction on P in Appendix F.4. □

We then show a sort of reverse causality. If an event e' causes e to reverse, then e' prevents e. This means actions being reversed as part of a rollback must wait for the rollback to be completed.

Lemma 7.9 (Reverse inverse causality)**.** *Given a Roll-CCSK process P with $\{\!|P|\!\} = \langle (E, F, \mapsto, \rhd, \lambda, \mathsf{Act}), \mathsf{Init}, k\rangle$, whenever $e' \in X \mapsto \underline{e}$ and $e \neq e'$, we get $e' \rhd e$.*

Proof. We prove this by structural induction on P in Appendix F.5. □

We then show that the reversal of an event is associated with at most one bundle. This means no action needs multiple rolls to be in progress before it can reverse.

Lemma 7.10 (Single backwards bundle)**.** *Given a Roll-CCSK process P such that $\{\!|P|\!\} = \langle (E, F, \mapsto, \rhd, \lambda, \mathsf{Act}), \mathsf{Init}, k\rangle$, for any event $e \in F$, there exists at most one bundle $X \mapsto \underline{e}$ such that $X \neq \{e\}$.*

Proof. We prove this by structural induction on P in Appendix F.6. □

The final lemma we need for structural congruence says that if e' causes an action event e then every rollback that reverses e' must also reverse e.

Lemma 7.11 (Reverse transitivity)**.** *Given a Roll-CCSK process P such that $\{\!|P|\!\} = \langle (E, F, \mapsto, \rhd, \lambda, \mathsf{Act}), \mathsf{Init}, k\rangle$, whenever $e' \in X \mapsto e$, $X' \mapsto \underline{e'}$, $X' \neq \{e'\}$, and $\lambda(e) = \mu$, there must exist $X'' \supseteq X'$ such that $X'' \mapsto \underline{e}$.*

Proof. We prove this by structural induction on P in Appendix F.7. □

Having introduced these lemmas, we are ready to prove that structurally congruent processes generate isomorphic event structures.

Proposition 7.12 (Structural Congruence)**.** *Given consistent roll-CCSK-processes P and P', if $P \equiv P'$, $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k\rangle$, and $\{\!|P'|\!\} = \left\langle \mathcal{E}', \mathsf{Init}', k'\right\rangle$, 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 prove this by case analysis on the structural congruence rules. We use Lemmas 7.7, 7.8, 7.9, 7.10 and 7.11. The full proof can be seen in Appendix F.8. □

We then prove, much like we did for our CCSK event structure semantics, that, as stated in Theorems 7.13 and 7.14, a process P has a transition $P \xrightarrow{\mu} P'$ if and only if P and P' correspond to isomorphic event structures, and there exists a μ-labelled transition from the initial state of $\{\!|P|\!\}$ to the initial state of $\{\!|P'|\!\}$.

Theorem 7.13. *Let P be a consistent roll-CCSK process. If $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k\rangle_l$, $\mathcal{E} = (E, F, \mapsto, \rhd, \lambda, \mathsf{Act})$, $C_{re}(\mathcal{E}) = (E, F, \mathsf{C}, \to)$, Init is conflict-free, and there exists a transition $P \xrightarrow{\mu[m]} P'$ such that $\{\!|P'|\!\} = \left\langle \mathcal{E}', \mathsf{Init}', k'\right\rangle_l$, then there exists isomorphism $f : \mathcal{E} \to \mathcal{E}'$ and a transition $\mathsf{Init} \xrightarrow{\{e\}} X$ such that $\lambda(e) = \mu$, $f \circ k' = k \cup \{(e, m)\}$, and $f(X) = \mathsf{Init}'$.*

Proof. We prove this by induction on $P \xrightarrow{\mu[m]} P'$ using Proposition 7.12 and Lemmas 7.7, 7.9, and 7.11. The full proof can be seen in Appendix F.9. □

Having shown that each forwards transition in the operational semantics corresponds to one in the generated event structure, we now show the converse.

Theorem 7.14. *Let P be a consistent roll-CCSK process. If $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k\rangle$, $\mathcal{E} = (E, F, \mapsto, \rhd, \lambda, \mathsf{Act})$, $C_{br}(\mathcal{E}) = (E, F, \mathsf{C}, \to)$, Init is conflict-free, and there exists a transition $\mathsf{Init} \xrightarrow{e} X$ in $C_{br}(\mathcal{E})$ such that $\lambda(e) = \mu$, then there exists a key m and a transition $P \xrightarrow{\mu[m]} P'$, such that $\{\!|P'|\!\} = \left\langle \mathcal{E}', \mathsf{Init}', k'\right\rangle$ and there exists isomorphism $f : \mathcal{E} \to \mathcal{E}'$ such that $f \circ k' = k \cup \{(e, m)\}$ and $f(X) = \mathsf{Init}'$.*

Proof. We prove this by induction on P in Appendix F.10 using Lemma 7.6. □

We then prove the same correspondence for $\mathsf{start\ roll}$ transitions.

Proposition 7.15. *Let P be a consistent roll-CCSK process. If $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k\rangle$, $\mathcal{E} = (E, F, \mapsto, \rhd, \lambda, \mathsf{Act})$, $C_{re}(\mathcal{E}) = (E, F, \mathsf{C}, \to)$, Init is conflict-free, and there exists a transition $P \xrightarrow{\mathsf{start\ roll}\ \gamma} P'$ such that $\{\!|P'|\!\} = \left\langle \mathcal{E}', \mathsf{Init}', k'\right\rangle$, then there exist isomorphisms $f : \mathcal{E} \to \mathcal{E}'$ and $g : \mathcal{E}' \to \mathcal{E}$ and a transition $\mathsf{Init} \xrightarrow{\{e\}} X$ such that $\lambda(e) = \mathsf{start\ roll}\ \gamma$, $f \circ k' = k \cup \{(e, m)\}$, and $f(X) = \mathsf{Init}'$.*

Proof. Similar to the proof of Theorem 7.13. □

Proposition 7.16. *Let P be a consistent roll-CCSK process. If $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k\rangle$, $\mathcal{E} = (E, F, \mapsto, \rhd, \lambda, \mathsf{Act})$, $C_{br}(\mathcal{E}) = (E, F, \mathsf{C}, \to)$, Init is conflict-free, and there exists a transition $\mathsf{Init} \xrightarrow{e} X$ in $C_{br}(\mathcal{E})$ such that $\lambda(e) = \mathsf{start\ roll}\ \gamma$, then there exists a a key m and a transition $P \xrightarrow{\mathsf{start\ roll}\ \gamma} P'$, such that $\{\!|P'|\!\} = \left\langle \mathcal{E}', \mathsf{Init}', k'\right\rangle$ and there exist*

isomorphisms $f : \mathcal{E} \to \mathcal{E}'$ *and* $g : \mathcal{E}' \to \mathcal{E}$ *such that* $f \circ k' = k \cup \{(e, m)\}$ *and* $f(X) = \mathsf{Init}'$.

Proof. Similar to the proof of Theorem 7.14. □

We finally need to prove that P can do a $\mathsf{roll}\ \gamma$ transition if and only if the event structure generated by P can do a $\mathsf{roll}\ \gamma$-labelled event followed by reversing all the events corresponding to actions and $\mathsf{start}\ \mathsf{roll}$'s with tags causally dependent on γ and then undoing $\mathsf{roll}\ \gamma$. For this we need Lemmas 7.17 to 7.21.

Lemma 7.17 states that all rolls are in conflict.

Lemma 7.17. *Let* P *be a roll-CCSK process with* $\{\!|\, P \,|\!\} = \langle (E, F, \mapsto, \rhd, \lambda, \mathsf{Act}), \mathsf{Init}, k \rangle$. *Given events* $e, e' \in E$ *such that* $e \neq e'$ *and there exist tags* γ, γ' *such that* $\lambda(e) = \mathsf{roll}\ \gamma$ *and* $\lambda(e') = \mathsf{roll}\ \gamma'$ *we get* $e \rhd e'$.

Proof. It is obvious from the syntax of Roll-CCSK that e and e' come from parallel subprocesses or different branches of a choice, and the result follows from the parallel composition and choice rules of Table 7. □

Lemma 7.18 states that reversal of actions and rollback initiations are only caused by rollbacks.

Lemma 7.18. *Let* P *be a roll-CCSK process. If* $\{\!|\, P \,|\!\} = \langle (E, F, \mapsto, \rhd, \lambda, \mathsf{Act}), \mathsf{Init}, k \rangle$ *and* $e \in E$ *where there does not exist* γ *such that* $\lambda(e) = \mathsf{roll}\ \gamma$. *Then whenever* $X \mapsto \underline{e}$, *either* $X = \{e\}$ *or for all* $e' \in X$, *there does not exist* γ' *such that* $\lambda(e') = \mathsf{roll}\ \gamma'$.

Proof. Obvious in most cases. In parallel composition we use the fact that we never have $e'' \in X'' \mapsto e'''$ where there does not exist γ'' such that $\lambda(e'') = \mathsf{roll}\ \gamma''$. □

Lemma 7.19 states that if an event e' in the initial state of a process prevents another event e in the initial state from reversing, then the key of e must have caused the key of e'.

Lemma 7.19. *Let* P *be a roll-CCSK process. If* $\{\!|\, P \,|\!\} = \langle (E, F, \mapsto, \rhd, \lambda, \mathsf{Act}), \mathsf{Init}, k \rangle$ *and* $e, e' \in \mathsf{Init}$ *then* $e' \rhd \underline{e}$ *if and only if* $k(e) \leq_P k(e')$.

Proof. We prove this by induction on P. It is trivial in all cases except $P = \alpha_{\gamma''}[n].P'$ and $P = P_0 \mid P_1$.

- Suppose $P = \alpha_{\gamma''}[n].P'$. Then if $e' \rhd \underline{e}$ either $e = e_\alpha$ or the result follows from induction and the fact that $e \in \mathsf{Init}$ means there does not exist γ'' such that $\lambda(e) = \mathsf{roll}\ \gamma''$. If $e = e_\alpha$ then $e' \rhd \underline{e}$ unless $\lambda(e') \in \{\mathsf{start}\ \mathsf{roll}\ \gamma', \mathsf{roll}\ \gamma\}$ for some γ.
- Suppose $P = P_0 \mid P_1$. Then there exists $i \in \{0, 1\}$ such that either $\pi_i(e') \rhd_i \underline{\pi_i(e)}$ or $\pi_i(e') = \pi_i(e)$. If $\pi_i(e') \rhd_i \underline{\pi_i(e)}$ then the result follows from induction, and if $\pi_i(e') = \pi_i(e)$ then that contradicts $e, e' \in \mathsf{Init}$. □

We now need a function, $\mathsf{N}(e)$, to give us the key an event e labelled with an initiated rollback $\mathsf{roll}\ \gamma$ is rolling back to in Definition 7.20. We prove that N finds such a key in Lemma 7.21.

Definition 7.20 (N)**.** Let P be a Roll-CCSK process with $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k\rangle$ and $\mathcal{E} = (E, F, \mapsto, \rhd, \lambda, \mathsf{Act})$. We define a partial function N on events such that for an event $e \in E$, $\mathsf{N}(e) = \mathsf{roll}\ n$ if (1) $\lambda(e) = \mathsf{roll}\ \gamma$, (2) $\mathsf{Init} \xrightarrow{e}$, (3) for any key m, we have $m \geq_P n$ if and only if there exists an event $e' \in \mathsf{Init}$ such that $k(e') = m$ and $X \subseteq E$ such that $X \mapsto \underline{e'}$ and $e \in X$. Otherwise $\mathsf{N}(e)$ is undefined.

Lemma 7.21. *Let P be a consistent forwards-reachable roll-CCSK process such that $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k\rangle$ and $\mathcal{E} = (E, F, \mapsto, \rhd, \lambda, \mathsf{Act})$, then:*

1. *Let $e \in E$ be an event such that $\lambda(e) = \mathsf{roll}\ \gamma$ and $\mathsf{Init} \xrightarrow{e}$. Then $\mathsf{N}(e)$ is defined and unique.*
2. *Let $P \xrightarrow{\mathsf{roll}\ n}$. Then there exists a unique $e \in E$ such that $\mathsf{N}(e) = \mathsf{roll}\ n$.*

Proof.

1. By Theorems 7.13 and 7.14 any key m occurs in P if and only if it occurs in $k(\mathsf{Init})$ and given $e, e' \in \mathsf{Init}$, $e \in X \mapsto e'$ if and only if $k(e) <_P k(e')$. The rest follows from Lemma 7.11.
2. Since P is consistent and forwards-reachable, exactly one $\alpha_\gamma[n]$ must occur somewhere in P and by Theorems 7.13 and 7.14, we have $e' \in E$ such that $\lambda(e') = \alpha$ and $k(e') = n$. Then we say that e is the event such that $\lambda(e) = \mathsf{roll}\ \gamma$ and by similar logic to the first part, $\mathsf{N}(e) = \lambda(e) = \mathsf{roll}\ n$. To prove uniqueness, we only need to show that in the subprocess $\alpha_\gamma[n].P'$ of P, no other such e' exists. This follows from the definition of R in $\left\{\!\left|\alpha_\gamma[n].P'\right|\!\right\}$ in Table 7 and the consistency of P. □

With these lemmas and definitions introduced, we will now prove Theorems 7.22 and 7.23. These state that a process P can do $\mathsf{roll}\ \gamma$ if and only if the REBES generated by P can do a $\mathsf{roll}\ \gamma$ event, followed by undoing all the events corresponding to actions P rolled back, and then reverse the $\mathsf{roll}\ \gamma$ event.

Theorem 7.22. *Let P be a roll-CCSK process such that $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k\rangle$. Whenever $P \overset{\mathsf{roll}\ n}{\rightsquigarrow} P'$, there exist e and $e_0, e_1, \ldots e_n$ such that $\mathsf{Init} \xrightarrow{\{e\}} X_0 \xrightarrow{\{\underline{e_0}\}} X_1 \cdots \xrightarrow{\{\underline{e_n}\}} X_{n+1} \xrightarrow{\{\underline{e}\}} X_{\mathsf{done}}$, $\mathsf{N}(e) = \mathsf{roll}\ n$, $\{e_0, e_1, \ldots e_n\} = \{e' \mid n \leq_P k(e')\}$, $\{\!|P'|\!\} = \left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$, and there exists an isomorphism $f : \mathcal{E} \to \mathcal{E}'$ such that $f(X_{\mathsf{done}}) = \mathsf{Init}'$.*

Proof. We prove this through induction on the derivation of $P \overset{\mathsf{roll}\ n}{\rightsquigarrow} P'$ using Lemmas 7.6, 7.10, 7.11, 7.17, and 7.18, Proposition 7.12 and 7.19, Theorem 7.13. The full proof can be seen in Appendix F.11. □

Having shown that each rollback transition in the operational semantics corresponds to a sequence of transitions in the generated event structure, we now show the converse.

Theorem 7.23. *Let P be a consistent roll-CCSK process such that $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k\rangle$. Whenever $\mathsf{Init} \xrightarrow{\{e_r\}} X_0 \xrightarrow{\{\underline{e_0}\}} X_1 \cdots \xrightarrow{\{\underline{e_n}\}} X_{n+1} \xrightarrow{\{\underline{e_r}\}} X_{\mathsf{done}}$, we have a transition $P \overset{\mathsf{roll}\ n}{\rightsquigarrow} P'$, such that $\mathsf{N}(e_r) = \mathsf{roll}\ n$, $\{e_0, e_1, \ldots e_n\} = \{e' \mid n \leq_P k(e')\}$, $\{\!|P'|\!\} = \left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$, and there exists an isomorphism $f : \mathcal{E} \to \mathcal{E}'$ such that $f(X_{\mathsf{done}}) = \mathsf{Init}'$.*

Proof. Follows from Theorem 7.13 and Proposition 7.12 □

We have now proved an operational correspondence between the operational semantics presented in Section 6 and the denotational event structure semantics presented in this section. In particular, we have shown that using non-causal event structures lets us model a control operator.

8. Conclusion

We have defined a reversible variant of bundle event structures and a category using these as objects, and used a causal subcategory of this to describe event structure semantics of uncontrolled CCSK, the first event structure semantics of a reversible calculus that we are aware of. We chose to use CCSK for this purpose because the way CCSK records previous actions preserves the structure of the process, meaning we can use very similar semantics for previous and future actions.

Unlike previous efforts to describe truly concurrent semantics of a reversible process calculus such as the one using rigid families [8], we have generated both the event structure and the initial state directly from the process, rather than needing to first undo previous actions to get the original process and from there the event structure, and then redo the actions to get the initial state. This was made easier by CCSK using static reversibility, since we did not have to combine events generated separately from a memory and a process. Our event structure semantics has shown that the same process at different points in its execution will generate isomorphic event structures with different initial states.

We have also proposed a variant of CCSK called Roll-CCSK, which uses the rollback described in [17] to control its reversibility. We have given two semantics for this calculus, one in which the entire roll is completed in one step, and one in which each action being rolled back takes a step. We have proved an operational correspondence between these semantics. We have defined a reversible variant of extended bundle event structures, which add asymmetric conflict to bundle event structures, and a category using these as objects, and used *non-causal* variants of these to define the event structure semantics of Roll-CCSK. Using event structures with non-causal reversibility allows us to treat rolls as normal events where the process $a_\gamma.\mathsf{roll}\ \gamma$ has the event a_γ cause the event $\mathsf{roll}\ \gamma$, and at the same time a_γ prevents $\mathsf{roll}\ \gamma$ from reversing.

We have proved operational correspondence between the operational and event structure semantics of both CCSK (Theorems 4.18 and 4.19) and Roll-CCSK (Theorems 7.13, 7.14, 7.22, and 7.23).

Another common way to control reversibility is by commit or irreversible actions, introduced for reversible CCS in [10]. These are used to denote a safe state which the process cannot reverse past. As such they function as a dual of the rollback, which ensures the process will reverse when hitting a fail state. Irreversible actions are simple to add to event structure semantics of CCSK by making the corresponding events irreversible, but would be more challenging to add to Roll-CCSK due to potentially having events which are irreversible but also required to reverse in order to finish a roll. We chose to focus on rollback since, as pointed out in [17], rollback gives the programmer more control than commit and more closely models system recovery techniques.

Future work. There exist many other reversible calculi which one might want to define event structure semantics for, most of which deal with previous actions by putting them into separate memories, rather than annotating them and keeping them in the process as CCSK does. This will likely make defining event structure semantics more challenging, particularly if trying to avoid defining the event structure corresponding to the fully reversed process first and then finding the action which have already been performed. Having event structure semantics of multiple calculi would allow us to reason about them and compare them.

Acknowledgements. This work was partially supported by an EPSRC DTP award; and also by the following EPSRC projects: EP/V000462/1, 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] Clément Aubert and Ioana Cristescu. Contextual equivalences in configuration structures and reversibility. *JLAMP*, 86(1):77 – 106, 2017.

[2] Steve Awodey. *Category Theory, Second Edition*. Oxford Logic Guides. Oxford University Press, 2010.

[3] Gérard Boudol and Ilaria Castellani. Permutation of transitions: An event structure semantics for CCS and SCCS. In J. W. de Bakker, W. P. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *LNCS*, pages 411–427, Berlin, Heidelberg, 1989. Springer.

[4] Simon Castellan, Jonathan Hayman, Marc Lasson, and Glynn Winskel. Strategies as concurrent processes. *Electr. Notes Theor. Comput. Sci.*, 308:87–107, 2014.

[5] Ilaria Castellani and Zhang Guo-Qiang. Parallel product of event structures. *Theoretical Computer Science*, 179(1):203 – 215, 1997.

[6] Silvia Crafa, Daniele Varacca, and Nobuko Yoshida. Event Structure Semantics of Parallel Extrusion in the Pi-Calculus. In Lars Birkedal, editor, *FOSSACS*, volume 7213 of *LNCS*, pages 225–239, Berlin, Heidelberg, 2012. Springer.

[7] Ioana Cristescu, Jean Krivine, and Daniele Varacca. A compositional semantics for the reversible pi-calculus. In *IEEE Symposium on Logic in Computer Science*, LICS '13, pages 388–397, Washington, DC, USA, 2013. IEEE Computer Society.

[8] Ioana Cristescu, Jean Krivine, and Daniele Varacca. Rigid families for the reversible π-calculus. In *RC 2016*, volume 9720 of *LNCS*, pages 3–19. Springer, 2016.

[9] Vincent Danos and Jean Krivine. Reversible Communicating Systems. In Philippa Gardner and Nobuko Yoshida, editors, *CONCUR*, volume 3170 of *LNCS*, pages 292–307, Berlin, Heidelberg, 2004. Springer.

[10] Vincent Danos and Jean Krivine. Transactions in RCCS. In Martín Abadi and Luca de Alfaro, editors, *CONCUR*, volume 3170 of *LNCS*, pages 398–412, Berlin, Heidelberg, 2004. Springer.

[11] Vincent Danos and Jean Krivine. Formal Molecular Biology Done in CCS-R. *Electr. Notes Theor. Comput. Sci.*, 180(3):31 – 49, 2007. BioConcur.

[12] Harald Fecher, Mila Majster-Cederbaum, and Jinzhao Wu. Bundle event structures: A revised cpo approach. *Information Processing Letters*, 83(1):7 – 12, 2002.

[13] Elena Giachino, Ivan Lanese, Claudio Antares Mezzina, and Francesco Tiezzi. Causal-consistent rollback in a tuple-based language. *JLAMP*, 88:99–120, 2017.

[14] Eva Graversen, Iain Phillips, and Nobuko Yoshida. Event structure semantics of (controlled) reversible CCS. In Jarkko Kari and Irek Ulidowski, editors, *RC*, volume 11106 of *LNCS*, pages 102–122. Springer, 2018.

[15] Eva Graversen, Iain Phillips, and Nobuko Yoshida. Towards a categorical representation of reversible event structures. *JLAMP*, 104:16 – 59, 2019.

[16] Ivan Lanese, Doriana Medic, and Claudio Antares Mezzina. Static versus dynamic reversibility in CCS. *Acta Informatica*, 2019.

[17] Ivan Lanese, Claudio Antares Mezzina, Alan Schmitt, and Jean-Bernard Stefani. Controlling Reversibility in Higher-Order Pi. In Joost-Pieter Katoen and Barbara König, editors, *CONCUR*, volume 6901 of *LNCS*, pages 297–311, Berlin, Heidelberg, 2011. Springer.

[18] Ivan Lanese, Claudio Antares Mezzina, and Jean-Bernard Stefani. Reversing Higher-Order Pi. In Paul Gastin and François Laroussinie, editors, *CONCUR*, volume 6269 of *LNCS*, pages 478–493, Berlin, Heidelberg, 2010. Springer.

[19] Ivan Lanese, Claudio Antares Mezzina, and Francesco Tiezzi. Causal-Consistent Reversibility. *Bulletin of the EATCS*, 114:17, November 2014.

[20] Ivan Lanese, Naoki Nishida, Adrián Palacios, and Germán Vidal. Cauder: A causal-consistent reversible debugger for erlang. In John P. Gallagher and Martin Sulzmann, editors, *Functional and Logic Programming*, pages 247–263, Cham, 2018. Springer International Publishing.

[21] Romanus Langerak. *Transformations and Semantics for LOTOS*. PhD thesis, Universiteit Twente, 1992.

[22] Doriana Medić and Claudio Antares Mezzina. Static VS Dynamic Reversibility in CCS. In Simon Devitt and Ivan Lanese, editors, *RC 2016*, volume 9720 of *LNCS*, pages 36–51. Springer International Publishing, 2016.

[23] Claudio Antares Mezzina and Vasileios Koutavas. A safety and liveness theory for total reversibility. In *TASE*, pages 1–8. IEEE, 2017.

[24] Mogens Nielsen, Gordon Plotkin, and Glynn Winskel. Petri nets, event structures and domains. In Gilles Kahn, editor, *Semantics of Concurrent Computation*, volume 70 of *LNCS*, pages 266–284, Berlin, Heidelberg, 1979. Springer.

[25] Iain Phillips and Irek Ulidowski. Reversibility and models for concurrency. *Electr. Notes Theor. Comput. Sci.*, 192(1):93–108, 2007.

[26] Iain Phillips and Irek Ulidowski. Reversing algebraic process calculi. *JLAMP*, 73(1-2):70–96, 2007.

[27] Iain Phillips and Irek Ulidowski. Reversibility and asymmetric conflict in event structures. *JLAMP*, 84(6):781 – 805, 2015.

[28] Iain Phillips, Irek Ulidowski, and Shoji Yuen. A Reversible Process Calculus and the Modelling of the ERK Signalling Pathway. In Robert Glück and Tetsuo Yokoyama, editors, *RC*, volume 7581 of *LNCS*, pages 218–232, Berlin, Heidelberg, 2013. Springer.

[29] Iain Phillips, Irek Ulidowski, and Shoji Yuen. Modelling of Bonding with Processes and Events. In Gerhard W. Dueck and D. Michael Miller, editors, *RC*, volume 7948 of *LNCS*, pages 141–154, Berlin, Heidelberg, 2013. Springer.

[30] Frits W Vaandrager. A simple definition for parallel composition of prime event structures. Technical report, Centre for Mathematics and Computer Science, P. O. box 4079, 1009 AB Amsterdam, The Netherlands, 1989. Report CS-R8903.

[31] Glynn Winskel. Event structure semantics for CCS and related languages. In Mogens Nielsen and Erik Meineche Schmidt, editors, *ICALP*, volume 140 of *LNCS*, pages 561–576, Berlin, Heidelberg, 1982. Springer.

A. Relating RBES to other categories of reversible event structures

In this appendix we recall the categories of reversible prime and stable event structures and describe how they relate to the category **RBES** defined in Section 3.2.

Definition A.1 (RPES [27]). A reversible prime event structure (RPES) is a sextuple $\mathcal{E} = (E, F, <, \sharp, \prec, \rhd)$ where E is the set of events, $F \subseteq E$ is the set of reversible events, and

- $< \subseteq E \times (E \cup \underline{F})$ is an irreflexive partial order such that for every $e \in E$, $\{e' \in E \mid e' < e\}$ is finite and conflict-free;
- $\sharp \subseteq E \times E$ is irreflexive and symmetric such that if $e < e'$ then not $e \sharp e'$;
- $\rhd \subseteq E \times \underline{F}$ is the prevention relation;
- $\prec \subseteq E \times \underline{F}$ is the reverse causality relation where for each $e \in F$, $e \prec \underline{e}$ and $\{e' \mid e' \prec \underline{e}\}$ is finite and conflict-free and if $e \prec \underline{e'}$ then not $e \rhd \underline{e'}$;
- $\sharp$ is hereditary with respect to sustained causation $\ll$ and $\ll$ is transitive, where $e \ll e'$ means that $e < e'$ and if $e \in F$ then $e' \rhd \underline{e}$.

Definition A.2 (RPES morphism [15]). Let $\mathcal{E}_0 = (E_0, F_0, <_0, \sharp_0, \prec_0, \rhd_0)$ and $\mathcal{E}_1 = (E_1, F_1, <_1, \sharp_1, \prec_1, \rhd_1)$ be RPESs. A morphism $f : \mathcal{E}_0 \to \mathcal{E}_1$ is a partial function $f : E_0 \to E_1$ such that

- for all $e \in E_0$, if $f(e) \neq \bot$ then $\{e_1 \mid e_1 <_1 f(e)\} \subseteq \{f(e') \mid e' <_0 e\}$;
- for all $e, e' \in E_0$, if $f(e) \neq \bot \neq f(e')$ and $f(e) \sharp_1 f(e')$ then $e \sharp_0 e'$;
- for all $e \in F_0$, if $f(e) \neq \bot$ then $\{e_1 \mid e_1 \prec_1 \underline{f(e)}\} \subseteq \{f(e') \mid e' \prec_0 \underline{e}\}$;
- for all $e \in E_0$ and $e' \in F_0$, if $f(e) \neq \bot \neq f(e')$ and $f(e) \rhd_1 \underline{f(e')}$ then $e \rhd_0 \underline{e'}$;
- for all $e, e' \in E_0$, if $f(e) = f(e') \neq \bot$ and $e \neq e'$ then $e \sharp_0 e'$;
- $f(F_0) \subseteq F_1$.

Definition A.3 (RPES to RBES). The functor $B_r : \mathbf{RPES} \to \mathbf{RBES}$ is defined as:

1. $B_r((E, F, <, \sharp, \prec, \rhd)) = (E, F, \mapsto, \sharp, \rhd)$, where $\{e'\} \mapsto e$ if $e' < e$ and $\{e'\} \mapsto \underline{e}$ if $e' \prec \underline{e}$;
2. $B_r(f) = f$.

Definition A.4 (SRES [15]). A stable reversible event structure (SRES) is a triple $\mathcal{E} = (E, F, \mathsf{Con}, \vdash)$ where E is the set of events, $\mathsf{Con} \subseteq_{\mathrm{fin}} 2^E$ is the consistency relation, which is left-closed, $\vdash \subseteq \mathsf{Con} \times 2^E \times (E \cup \underline{E})$ is the enabling relation, and

1. if $X \otimes Y \vdash e^*$ then $(X \cup \{e\}) \cap Y = \emptyset$;
2. if $X \otimes Y \vdash \underline{e}$ then $e \in X$;
3. if $X \otimes Y \vdash e^*$, $X \subseteq X' \in \mathsf{Con}$, and $X' \cap Y = \emptyset$ then $X' \otimes Y \vdash e^*$;
4. if $X \otimes Y \vdash e^*$, $X' \otimes Y' \vdash e^*$, and $X \cup X' + e^* \in \mathsf{Con}$ then $X \cap X' \otimes Y \cap Y' \vdash e^*$.

Definition A.5 (SRES morphism [15]). Let $(E_0, \mathsf{Con}_0, \vdash_0)$ and $(E_1, \mathsf{Con}_1, \vdash_1)$ be two SRESs. A morphism $f : (E_0, \mathsf{Con}_0, \vdash_0) \to (E_1, \mathsf{Con}_1, \vdash_1)$ is a partial function $f : E_0 \to E_1$ such that:

- for all $e \in E_0$, if $f(e) \neq \bot$ and $X \otimes Y \vdash_0 e^*$ then there exists a $Y_1 \subseteq E_1$ such that for all $e_0 \in E_0$, if $f(e_0) \in Y_1$ then $e_0 \in Y$ and $f(X) \otimes Y_1 \vdash_1 f(e)^*$;
- for any $X_0 \in \mathsf{Con}_0$, $f(X_0) \in \mathsf{Con}_1$;
- for all $e, e' \in E_0$, if $f(e) = f(e') \neq \bot$ and $e \neq e'$ then no $X \in \mathsf{Con}_0$ exists such that $e, e' \in X$.

Definition A.6 (FCRBES to SRES). The functor $E_b : \mathbf{FCRBES} \to \mathbf{SRES}$ is defined as:

1. $E_b((E, F, \mapsto, \sharp, \rhd)) = (E, F, \mathsf{Con}, \vdash)$ where
   (a) Con is the set of finite conflict-free subsets of E;
   (b) For $e \in E$, $X \otimes \emptyset \vdash e$ if $X \cup \{e\} \in \mathsf{Con}$ and for all $X' \subseteq E$ such that $X' \mapsto e$, $X' \cap X \neq \emptyset$;

(c) For $e \in F$, $X \otimes Y \vdash \underline{e}$ if $X \in \mathsf{Con}$, for all $X' \subseteq E$ such that $X' \mapsto \underline{e}$, $X' \cap X \neq \emptyset$, $Y = \{e' \mid e' \rhd \underline{e}\}$, and $Y \cap X = \emptyset$;

2. $E_b(f) = f$.

Proposition A.7. *Given a FCRBES $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd)$, we have that $E_b(\mathcal{E})$ is an SRES.*

Proof. We prove the conditions of Definition A.4.

1. If $X \otimes Y \vdash e$ then $Y = \emptyset$ and therefore $(X \cup \{e\}) \cap Y = \emptyset$. If $X \otimes Y \vdash \underline{e}$ then $Y \cap X = \emptyset$ and $e \in X$, and therefore $(X \cup \{e\}) \cap Y = \emptyset$.
2. If $X \otimes Y \vdash \underline{e}$ then $\{e\} \mapsto \underline{e}$ and therefore $\{e\} \cap X \neq \emptyset$ and $e \in X$.
3. If $X \otimes Y \vdash e^*$, $X \subseteq X' \in \mathsf{Con}$, and $X' \cap Y = \emptyset$ then for all $X'' \mapsto e^*$, $X \cap X' \neq \emptyset$, and therefore $X' \otimes Y \vdash e^*$.
4. If $X \otimes Y \vdash e^*$, $X' \otimes Y' \vdash e^*$, and $X \cup X' + e^* \in \mathsf{Con}$ then, $Y = Y'$, $X \cup X'$ is conflict-free, meaning there must exist an $X'' \subseteq (X \cap X')$ such that for all $X''' \mapsto e^*$, we have $X''' \cap X'' \neq \emptyset$. This means $X \cap X' \otimes Y \cap Y' \vdash e^*$. □

B. Proofs from Section 3

B.1. Proof of Proposition 3.10

Proof. We first show that π_0 and π_1 are morphisms:

1. If $\pi_i(e) \sharp_i \pi_i(e')$, then obviously $e \sharp e'$.
2. If $\pi_i(e) = \pi_i(e')$ and $e \neq e'$, then $\pi_{1-i}(e) \neq \pi_{1-i}(e')$, and therefore $e \sharp e'$.
3. If $X_i \mapsto \pi_i(e)$, then $\{e' \in E \mid \pi_i(e') \in X_i\} \mapsto e$. Clearly $\pi_i(\{e' \in E \mid \pi_i(e') \in X_i\}) = X_i$, and for all $e' \in \{e' \in E \mid \pi_i(e') \in X_i\})$, $\pi_i(e') \neq \bot$.
4. If X is a configuration of $\mathcal{E}_0 \times \mathcal{E}_1$, then we show that $\pi_i(X)$ satisfies the requirements of a configuration of $\mathcal{E}_i$:
   (a) As shown above $\pi_i(e) \sharp_i \pi_i(e') \Rightarrow e \sharp e'$, which means that X conflict-free implies $\pi_i(X)$ conflict-free.
   (b) If there exists a sequence $e_1, \ldots, e_n$ such that $X = \{e_1, \ldots, e_n\}$ and for all j, $1 \leq j \leq n$, if $Y \mapsto e_{j+1}$ then $\{e_1, \ldots, e_j\} \cap Y \neq \emptyset$, then $\pi_i(X) = \{\pi_i(e_1), \ldots, \pi_i(e_n)\}$ and if $\pi_i(e_{j+1}) \neq \bot$, then whenever $Y_i \mapsto \pi_i(e_{j+1})$, $\{e' \in E \mid \pi_i(e') \in Y_i\} \mapsto e_{j+1}$, meaning $\{e' \in E \mid \pi_i(e') \in Y_i\} \cap \{e_1, \ldots, e_j\} \neq \emptyset$. Therefore, we must get $Y_i \cap \{\pi_i(e_1), \ldots, \pi_i(e_j)\} \neq \pi_i(\emptyset) = \emptyset$.

We then show that for any BES, $\mathcal{E}_2 = (E_2, \mapsto_2, \sharp_2)$, if there exist morphisms $f_0 : \mathcal{E}_2 \to \mathcal{E}_0$ and $f_1 : \mathcal{E}_2 \to \mathcal{E}_1$, then there exists a unique morphism $f : \mathcal{E}_2 \to \mathcal{E}$, such that $\pi_0 \circ f = f_0$ and $\pi_1 \circ f = f_1$.

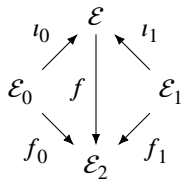
Clearly $f(e) = (f_0(e), f_1(e))$ is the only partial function for which this commutes, so we prove it to be a morphism:

1. If $f(e) \sharp f(e')$ then there exists $i \in \{0, 1\}$ such that either $\pi_i(f(e)) \sharp_i \pi_i(f(e'))$, in which case clearly $f_i(e) \sharp_i f_i(e')_i$, and therefore $e \sharp_2 e'$, or $\pi_i(f(e)) = \pi_i(f(e')) \neq \bot$ and $\pi_{1-i}(f(e)) \neq \pi_{1-i}(f(e'))$, in which case $f_i(e) = f_i(e') \neq \bot$, and $e \neq e'$, meaning $e \sharp_2 e'$.

2. If $f(e) = f(e') \neq \bot$ then $f_0(e) = f_0(e') \neq \bot$ or $f_1(e) = f_1(e') \neq \bot$, meaning if $e \neq e'$ then $e \sharp_2 e'$.
3. For $X \subseteq E$, if $X \mapsto f(e)$, then there exists $i \in \{0, 1\}$ and $X_i \subseteq E_i$ such that $X_i \mapsto \pi_i(e)$ and $X = \{e' \in E \mid \pi_i(e') \in X_i\}$. And since $X_i \mapsto f_i(e)$, there exists $X_2 \subseteq E_2$ such that $X_2 \mapsto_2 e$, $f_i(X_2) \subseteq X_i$, and if $e' \in X_2$ then $f_i(e') \neq \bot$. Clearly $f(X_2) \subseteq X$.
4. If X is a configuration of $\mathcal{E}_2$, then we show that $f(X)$ satisfies the requirements of a configuration of $\mathcal{E}_0 \times \mathcal{E}_1$:
   (a) As shown above, if $f(e) \sharp f(e')$, then $e \sharp_2 e'$, meaning if X is conflict-free, then $f(X)$ is conflict-free.
   (b) If there exists a sequence $e_1, \ldots, e_n$ such that $X = \{e_1, \ldots, e_n\}$ and for all j, $1 \leq j \leq n$, if $Y \mapsto e_{j+1}$ then $\{e_1, \ldots, e_j\} \cap Y \neq \emptyset$, then $f(X) = \{f(e_1), \ldots, f(e_n)\}$ and if $f(e_{j+1}) \neq \bot$, then whenever $Y' \mapsto f(e_{j+1})$, $\{e' \in E \mid f(e') \in Y'\} \mapsto e_{j+1}$, meaning $\{e' \in E \mid f(e') \in Y'\} \cap \{e_1, \ldots, e_j\} \neq \emptyset$. Therefore, we must get $Y' \cap \{f(e_1), \ldots, f(e_j)\} \neq f(\emptyset) = \emptyset$. □

B.2. Proof of Proposition 3.13

Proof. Obviously $\mathcal{E}$ is a BES, and ι_0 and ι_1 are morphisms, so we simply need to prove that if there exists a BES $\mathcal{E}_2 = (E_2, \mapsto_2, \sharp_2)$ and morphisms $f_0 : \mathcal{E}_0 \to \mathcal{E}_2$ and $f_1 : \mathcal{E}_1 \to \mathcal{E}_2$, then there exists a unique BES morphism $f : \mathcal{E} \to \mathcal{E}_2$ such that the following commutes:



Clearly the only partial function for which this could hold is $f(j, e) = f_j(e)$, so we prove it to be a morphism:

- If $f(e) \sharp_2 f(e')$ then $e = (j, e_j)$, $e' = (j', e_{j'})$ $f_j(e_j) = f(e)$, $f_{j'}(e_{j'}) = f(e')$, and either $j \neq j'$ or $j = j'$. If $j \neq j'$, then obviously $e \sharp e'$. If $j = j'$, then $f_j(e_j) \sharp_2 f_j(e_{j'})$, meaning $e_j \sharp_j e_{j'}$, and therefore $e \sharp e'$.
- If $f(e) = f(e') \neq \bot$ then $e = (j, e_j)$ and $e' = (j', e_{j'})$ and $f_j(e_j) = f(e) = f(e') = f_{j'}(e_{j'})$.

  If $j = j'$ then $e \neq e'$ means that $e_j \neq e_{j'}$, which means that $e_j \sharp_j e_{j'}$ and therefore $e \sharp e'$.

  If $j \neq j'$, then by definition $e \sharp e'$.
- If $X_2 \mapsto f(e)$, then $e = (j, e_j)$, and there exists X_j such that $X_j \mapsto_j e_j$, $f_j(X_j) \subseteq X_2$, and if $e'_j \in X_j$ then $f_j(e'_j) \neq \bot$. This means $\{(j, e'_j) \mid e'_j \in X_j\} \mapsto e$, $f(\{(j, e'_j) \mid e'_j \in X_j\}) \subseteq X_2$, and if $e' \in \{(j, e'_j) \mid e'_j \in X_j\}$ then $e' \neq \bot$.

The diagram obviously commutes. □

B.3. Proof of Proposition 3.30

Definition B.1 (CS morphism). Let $C_0 = (E_0, F_0, \mathsf{C}_0, \to_0)$ and $C_1 = (E_1, F_1, \mathsf{C}_1, \to_1)$ be configuration systems. A CS morphism is a partial function $f : E_0 \to E_1$ such that

1. for any $X \in \mathsf{C}_0$, $f(X) \in \mathsf{C}_1$;
2. for any $X, Y \in \mathsf{C}_0$, $A \subseteq E_0$, and $B \subseteq F_0$, if $X \xrightarrow{A \cup \underline{B}}_0 Y$ and $f(A \cup B) \neq \emptyset$ then $f(X) \xrightarrow{f(A) \cup f(\underline{B})}_1 f(Y)$;
3. for all $e_0, e_0' \in E_0$, if $f(e_0) = f(e_0') \neq \bot$ and $e_0 \neq e_0'$ then there exists no $X \in \mathsf{C}_0$ such that $e_0, e_0' \in X$.

Proof. We first show that if $\mathcal{E} = (E, F, \mapsto, \sharp, \rhd)$ is an RBES, then $C_{br}(\mathcal{E}) = (E, F, \mathsf{C}, \to)$ is a CS, as defined in Definition 3.28, meaning that for $X, Y \in \mathsf{C}$, $A \subseteq E$, and $B \subseteq F$, if $X \xrightarrow{A \cup \underline{B}} Y$, then:

1. $A \cap X = \emptyset$, $B \subseteq X \cap F$, and $Y = (X \setminus B) \cup A$ by definition.
2. For all $A' \subseteq A$ and $B' \subseteq B$, $(X \setminus B') \cup A = Z \in \mathsf{C}$ because $Z \subseteq X \cup A \in \mathsf{C}$. Moreover $X \xrightarrow{A' \cup \underline{B'}} Z$ and $Z \xrightarrow{(A \setminus A') \cup \underline{(B \setminus B')}} Y$ obviously fulfil the conditions for transitions.

We then show that if $f : \mathcal{E}_0 \to \mathcal{E}_1$ is an RBES morphism then $f : C_{br}(\mathcal{E}_0) \to C_{br}(\mathcal{E}_1)$ is a CS morphism satisfying the conditions of Definition B.1:

1. Suppose $X \in \mathsf{C}_0$. Then X is conflict-free. Since, by definition of an RBES morphism, $f(e_0) \sharp_1 f(e_0') \Rightarrow e_0 \sharp_0 e_0'$, this implies that $f(X)$ is conflict-free, and therefore $f(X) \in \mathsf{C}_1$.
2. Suppose $X \xrightarrow{A \cup \underline{B}}_0 Y$. Then $f(X) \xrightarrow{f(A) \cup f(\underline{B})}_1 f(Y)$ because:
   (a) $f(X), f(Y) \in \mathsf{C}_1$ since $X, Y \in \mathsf{C}_0$, as implied by item 1.
   (b) $f(Y) = (f(X) \setminus f(B)) \cup f(A)$ since $Y = (X \setminus B) \cup A$.
   (c) $f(A) \cap f(X) = \emptyset$ since $A \cap X = \emptyset$, and by definition of an RBES morphism, for all $e_0, e_0' \in E_0$, if $f(e_0) = f(e_0') \neq \bot$ and $e_0 \neq e_0'$ then $e_0 \sharp_0 e_0'$, implying $e_0, e_0' \notin (X \cup A) = Y \in \mathsf{C}_0$, since Y is conflict-free.
   (d) $f(B) \subseteq f(X)$ since $B \subseteq$.
   (e) $f(X) \cup f(A)$ is conflict-free because $X \cup A$ is conflict-free and $f(e_0) \sharp_1 f(e_0') \Rightarrow e_0 \sharp_0 e_0'$.
   (f) For all $e \in B$, if $e' \rhd \underline{e}$ then $e' \notin X \cup A$.
   (g) For all $e_1 \in f(B)$, if $e_1' \rhd_1 \underline{e_1}$ then $e_1' \notin f(X) \cup f(A)$ because for any $e_0 \in B$, $e_0' \in E_0$ such that $f(e_0) = e_1$ and $f(e_0') = e_1'$ we have $e_0' \rhd \underline{e_0}$ and therefore $e_0' \notin X \cup A$.
   (h) For all $e_1 \in f(A)$ and $X_1 \subseteq E_1$, if $X_1 \mapsto_1 e_1$ then $X_1 \cap (f(X) \setminus f(B)) \neq \emptyset$ because then there exists e_0 and X_0 such that $f(e_0) = e_1$, $X_0 \mapsto_0 e_0$, $f(X_0) \subseteq X_1$, and if $e_0' \in X_0$ then $f(e_0') \neq \bot$. Therefore, $X_0 \cap (X \setminus B) \neq \emptyset$, and $f(X_0) \cap (f(X) \setminus f(B)) \neq \emptyset$.
   (i) For all $e_1 \in f(B)$ and $X_1 \subseteq E_1$, if $X_1 \mapsto_1 \underline{e_1}$ then $X_1 \cap (f(X)1 \setminus (f(B) \setminus \{e_1\})) \neq \emptyset$ for similar reasons to the previous condition.

(j) For every $a \in f(A)$, $\{e \in E_1 \mid e <_1 f(a)\} \subseteq f(X)$ because for every $a_0 \in A$, $\{e \in E_0 \mid e <_0 a_0\} \subseteq X$, and by definition of a PES morphism, $\{e_1 \mid e_1 <_1 f(a_0)\} \subseteq \{f(e) \mid e <_0 a_0\}$.

3. Suppose $f(e_0) = f(e_0') \neq \bot$ and $e_0 \neq e_0'$. By definition of an RBES morphism, $e_0 \sharp_0 e_0'$, and since all $X \in \mathsf{C}_0$ are conflict-free, there exists no $X \in \mathsf{C}_0$ such that $e_0, e_0' \in X$.

Then to prove it is a functor we simply need to show that:

$C_{br}(f : \mathcal{E}_1 \to \mathcal{E}_2) = C_{br}(f) : C_{br}(\mathcal{E}_1) \to C_{br}(\mathcal{E}_2)$, which is obvious since $C_{br}(f) = f$, $C_{br}(E_0) = E_0$, and $C_{br}(E_1) = E_1$.

$C_{br}(1_{\mathcal{E}}) = 1_{C_{br}(\mathcal{E})}$ since the identity function for all RBES and CS objects is $f(e) = e$.

$C_{br}(f \circ f') = C_{br}(f) \circ C_{br}(f')$ since $C_{br}(f) = f$ and $C_{br}(f') = f'$. □

C. Proofs from Section 4

C.1. Proof of Theorem 4.18

Proof. We say that $\mathcal{E}' = (E', F', \mapsto', \sharp', \rhd', \lambda', \mathsf{Act}')$ and the inverse of f is $g : \mathcal{E}' \to \mathcal{E}$. We prove the theorem by inductions on $P \xrightarrow{\mu[m]} P'$ using the forwards semantics seen in Table 1 and describing how $\mathcal{E}$ and $\mathcal{E}'$ are constructed:

- Suppose $P = \alpha.Q$, $P' = \alpha[m].Q$, $\mu = \alpha$, and $\mathsf{std}(Q)$. Then there exist $\mathcal{E}_Q$ and e_α such that:
  - $\{\!|Q|\!\} = \langle \mathcal{E}_Q, \mathsf{Init}, k \rangle$,
  - $\mathcal{E}_Q = (E_Q, F_Q, \mapsto_Q, \sharp_Q, \rhd_Q, \lambda_Q, \mathsf{Act}_Q)$,
  - $e_\alpha \notin E_Q$,
  - $E = E_Q \cup \{e_\alpha\}$,
  - $F = F_Q \cup \{e_\alpha\}$,
  - $X \mapsto e^*$ if $X \mapsto_Q e^*$ or $X = \{e_\alpha\}$ and $e^* = e \in E_Q$,
  - $\sharp = \sharp_Q$,
  - $\rhd = \rhd_Q \cup (E_Q \times \{\underline{e_\alpha}\})$,
  - $\mathsf{Act} = \mathsf{Act}_Q \cup \{\alpha\}$,
  - for all $e \in E$, $\lambda(e) = \begin{cases} \lambda_Q(e) & \text{if } e \in E_Q \\ \alpha & \text{if } e = e_\alpha \end{cases}$,
  - and if $\mathsf{Init} \neq \emptyset$ then for all $e \in E$, $\{e\} \mapsto e$ and for all $e \in F$, $e \rhd \underline{e}$.

  There also exist $\mathcal{E}_Q'$ and e_α' such that:
  - $\{\!|Q'|\!\} = \langle \mathcal{E}_Q', \mathsf{Init}_Q', k_Q' \rangle$,
  - $\mathcal{E}'$ is constructed similarly to $\mathcal{E}$,

- $\mathsf{Init}' = \mathsf{Init}'_Q \cup \{e'_\alpha\}$,
- and $k'(e) = \begin{cases} k'_Q(e) & \text{If } e \in \mathsf{Init}'_Q \\ m & \text{If } e = e'_\alpha \end{cases}$.

As $\mathcal{E}_Q$ and $\mathcal{E}'_Q$ have been generated by the same process, we have isomorphisms $f_Q : \mathcal{E}_Q \to \mathcal{E}'_Q$ and $g_Q : \mathcal{E}'_Q \to \mathcal{E}_Q$. We say that $f = f_Q \cup \{(e_\alpha, e'_\alpha)\}$ and $g = g_Q \cup \{(e'_\alpha, e_\alpha)\}$. These are obviously isomorphisms.

Since Init is conflict-free and $\sharp = \sharp_Q$, $X = \mathsf{Init} \cup \{e_\alpha\}$ is conflict-free, and therefore a configuration of $C_{br}(\mathcal{E})$. And since no $X' \subseteq E$ exists such that $X' \mapsto e_\alpha$, we get $\mathsf{Init} \xrightarrow{\{e_\alpha\}} \{e_\alpha\}$, and clearly $\lambda(e_\alpha) = \alpha$ and $k(f(e)) = m$.

- Suppose that $P = \alpha[n].Q$, $P' = \alpha[n].Q'$, $Q \xrightarrow{\mu[m]} Q'$, and $m \neq n$. Then there exist $\mathcal{E}_Q$, Init_Q, k_Q, e_α, $\mathcal{E}'_Q$, Init'_Q, k'_Q, and e'_α similar to the previous case.

  By induction, we get isomorphisms $f_Q : \mathcal{E}_Q \to \mathcal{E}'_Q$ and $g_Q : \mathcal{E}'_Q \to \mathcal{E}_Q$ and a transition $\mathsf{Init}_Q \xrightarrow{\{e\}} X_Q$ in $C_{br}(\mathcal{E}_Q)$ such that $\lambda_Q(e) = \mu$, $k'_Q(f_Q(e)) = m$, and $f_Q(X_Q) = \mathsf{Init}'_Q$.

  We define $f = f_Q \cup \{(e_\alpha, e'_\alpha)\}$ and $g = g_Q \cup \{(e'_\alpha, e_\alpha)\}$. Since Init_Q and X_Q are conflict-free in $\mathcal{E}_Q$, $\mathsf{Init}_Q \cup \{e_\alpha\} = \mathsf{Init}$ and $X_Q \cup \{e_\alpha\} = X$ are configurations of $C_{rb}(\mathcal{E})$, and clearly $\mathsf{Init} \xrightarrow{\{e\}} X$.

- Suppose $P = Q \mid R$, $P' = Q' \mid R$, $Q \xrightarrow{\mu[m]} Q'$, and $\mathsf{fsh}[m](R)$. Then there exist $\mathcal{E}_Q$ and $\mathcal{E}_R$ such that:
  - $\{\!|Q|\!\} = \langle \mathcal{E}_Q, \mathsf{Init}_Q, k_Q \rangle$,
  - $\mathcal{E}_Q = (E_Q, F_Q, \mapsto_Q, \sharp_Q, \rhd_Q, \lambda_Q, \mathsf{Act}_Q)$,
  - $\{\!|R|\!\} = \langle \mathcal{E}_R, \mathsf{Init}_R, k_R \rangle$,
  - $\mathcal{E}_R = (E_R, F_R, \mapsto_R, \sharp_R, \rhd_R, \lambda_R, \mathsf{Act}_R)$,
  - $E = \{(e, *) \mid e \in E_Q\} \cup \{(*, e) \mid e \in E_R\} \cup \{(e, e') \mid e \in E_Q, e' \in E_R, \lambda_Q(e) = \overline{\lambda_R(e')}\}$,
  - $F = \{(e, *) \mid e \in F_Q\} \cup \{(*, e) \mid e \in F_R\} \cup \{(e, e') \mid e \in F_Q, e' \in F_R, \lambda_0(e) = \overline{\lambda_1(e')}\}$,
  - there exist π_Q and π_R such that $\pi_i(e_Q, e_R) = e_i$ for $i \in \{Q, R\}$,
  - $X \mapsto e^*$ if there exists $i \in \{Q, R\}$ and $X_i \subseteq E_i$ such that $X_i \mapsto_i \pi_i(e)$ and $X = \{e' \in E \mid \pi_i(e') \in X_i\}$,
  - $e \sharp e'$ if there exists $i \in \{Q, R\}$ such that $\pi_i(e) \sharp_i \pi_i(e')$ or $\pi_i(e) = \pi_i(e') \neq \bot$,
  - $e \rhd \underline{e'}$ if there exists $i \in \{Q, R\}$ such that $\pi_i(e) \rhd_i \underline{\pi_i(e')}$,
  - $\mathsf{Act} = \mathsf{Act}_Q \cup \mathsf{Act}_R \cup \{\tau\}$,

- $\lambda(e) = \begin{cases} \tau & \text{if } e = (e_Q, e_R) \\ \lambda_Q(e_Q) & \text{if } e = (e_Q, *) \\ \lambda_1(e_R) & \text{if } e = (*, e_R) \end{cases}$,
- $\mathsf{Init} = \{(e_Q, e_R) \mid e_Q \in \mathsf{Init}_Q, e_R \in \mathsf{Init}_R, k_Q(e_Q) = k_R(e_R)\} \cup \{(e_Q, *) \mid e_Q \in \mathsf{Init}_Q \text{ and } \nexists e_R \in \mathsf{Init}_R . \lambda_Q(e_Q) = \overline{\lambda_R(e_R)} \text{ and } k_Q(e_Q) = k_R(e_R)\} \cup \{(*, e_R) \mid e_R \in \mathsf{Init}_R \text{ and } \nexists e_Q \in \mathsf{Init}_Q . \lambda_Q(e_Q) = \overline{\lambda_R(e_R)} \text{ and } k_Q(e_Q) = k_R(e_R)\}$,
- and $k(e) = \begin{cases} k_Q(e_Q) & \text{If } e = (e_Q, *) \\ k_R(e_R) & \text{If } e = (*, e_R) \\ k_Q(e_Q) & \text{If } e = (e_Q, e_R) \end{cases}$.

We also have $\left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$ constructed similarly from some $\left\langle \mathcal{E}'_Q, \mathsf{Init}'_Q, k'_Q \right\rangle$ and $\left\langle \mathcal{E}'_R, \mathsf{Init}'_R, k'_R \right\rangle$ such that $\{\!|Q'|\!\} = \left\langle \mathcal{E}'_Q, \mathsf{Init}'_Q, k'_Q \right\rangle$ and $\{\!|R|\!\} = \left\langle \mathcal{E}'_R, \mathsf{Init}'_R, k'_R \right\rangle$. We clearly have isomorphisms $f_Q : \mathcal{E}_Q \to \mathcal{E}_{Q'}$, $g_Q : \mathcal{E}'_Q \to \mathcal{E}_Q$, $f_R : \mathcal{E}_R \to \mathcal{E}'_R$, and $g_R : \mathcal{E}'_R \to \mathcal{E}_R$ and a transition $\mathsf{Init}_Q \xrightarrow{\{e_Q\}} X_Q$ of $C_{rb}(\mathcal{E}_Q)$ such that $\lambda_Q(e) = \mu$, $k'_Q(f_Q(e_Q)) = m$, and $f_Q(X_Q) = \mathsf{Init}'_Q$.

Since Init is conflict-free and X_Q is conflict-free in $\mathcal{E}_Q$, clearly $\mathsf{Init} \cup \{(e_Q, *)\} = X$ is conflict-free, and $\mathsf{Init} \xrightarrow{(e_Q, *)} X$.

We define our isomorphisms as

$$f(e) = \begin{cases} (f_Q(e'), *) & \text{if } e = (e', *) \\ (*, f_R(e')) & \text{if } e = (*, e') \\ (f_Q(e'), f_R(e'')) & \text{if } e = (e', e'') \end{cases}$$

and

$$g(e) = \begin{cases} (g_Q(e'), *) & \text{if } e = (e', *) \\ (*, g_R(e')) & \text{if } e = (*, e') \\ (g_Q(e'), g_R(e'')) & \text{if } e = (e', e'') \end{cases}$$

And, since $\mathsf{fsh}[m](R)$, $f(X) = \mathsf{Init}'$. The rest of the proof is straightforward.

- Suppose $P = Q \mid R$, $P' = Q' \mid R'$, $Q \xrightarrow{\alpha[m]} Q'$, $R \xrightarrow{\overline{\alpha}[m]} R'$, and $\mu = \tau$. Then we have $\langle \mathcal{E}, \mathsf{Init}, k \rangle$ constructed from $\{\!|Q|\!\} = \left\langle \mathcal{E}_Q, \mathsf{Init}_Q, k_Q \right\rangle$ and $\{\!|R|\!\} = \langle \mathcal{E}_R, \mathsf{Init}_R, k_R \rangle$ and $\left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$ constructed from $\{\!|Q'|\!\} = \left\langle \mathcal{E}'_Q, \mathsf{Init}'_Q, k'_Q \right\rangle$ and $\{\!|R'|\!\} = \left\langle \mathcal{E}'_R, \mathsf{Init}'_R, k'_R \right\rangle$ as in the previous case.

  By induction, we have isomorphisms $f_Q : \mathcal{E}_Q \to \mathcal{E}'_Q$, $g_Q : \mathcal{E}'_Q \to \mathcal{E}_Q$, $f_R : \mathcal{E}_R \to \mathcal{E}_{R'}$, and $g_R : \mathcal{E}_{R'} \to \mathcal{E}_R$ and transitions $\mathsf{Init}_Q \xrightarrow{\{e_Q\}} X_Q$ of $C_{rb}(\mathcal{E}_Q)$ such

that $\lambda_Q(e) = \alpha$, $k'_Q(f_Q(e_Q)) = m$, and $f_Q(X_Q) = \mathsf{Init}'_Q$, and $\mathsf{Init}_R \xrightarrow{\{e_R\}} X_R$ of $C_{rb}(\mathcal{E}_R)$ such that $\lambda_R(e) = \overline{\alpha}$, $k_{R'}(f_R(e_R)) = m$, and $f_R(X_R) = \mathsf{Init}_{R'}$.

We define our isomorphisms as

$$f(e) = \begin{cases} (f_Q(e'), *) & \text{if } e = (e', *) \\ (*, f_R(e')) & \text{if } e = (*, e') \\ (f_Q(e'), f_R(e'')) & \text{if } e = (e', e'') \end{cases}$$

and

$$g(e) = \begin{cases} (g_Q(e'), *) & \text{if } e = (e', *) \\ (*, g_R(e')) & \text{if } e = (*, e') \\ (g_Q(e'), g_R(e'')) & \text{if } e = (e', e'') \end{cases}$$

We know that Init, X_Q, and X_R are conflict-free, so the only way $\mathsf{Init} \cup \{(e_Q, e_R)\}$ has conflict is if Init_Q or Init_R an event with the key m, which we know from Lemma 5.2 of [26] is not possible. The rest of the proof is straightforward.

- Suppose $P = Q + R$, $P' = Q' + R$, $Q \xrightarrow{\mu[m]} Q'$, and $\mathsf{std}(R)$. Then there exist $\mathcal{E}_Q$ and $\mathcal{E}_R$ such that:
  - $\{\!|Q|\!\} = \langle \mathcal{E}_Q, \mathsf{Init}_Q, k_Q \rangle$,
  - $\mathcal{E}_Q = (E_Q, F_Q, \mapsto_Q, \sharp_Q, \rhd_Q, \lambda_Q, \mathsf{Act}_Q)$,
  - $\{\!|R|\!\} = \langle \mathcal{E}_R, \mathsf{Init}_R, k_R \rangle$,
  - $\mathcal{E}_R = (E_R, F_R, \mapsto_R, \sharp_R, \rhd_R, \lambda_R, \mathsf{Act}_R)$,
  - $E = E_Q \cup E_R$,
  - $F = F_Q \cup F_R$,
  - $X \mapsto e^*$ if there exists $i \in \{Q, R\}$ such that $X \mapsto_i e^*$,
  - $\sharp = \sharp_Q \cup \sharp_R \cup (E_Q \times E_R) \cup (E_R \times E_Q)$,
  - $\rhd = \rhd_Q \cup \rhd_R \cup (E_Q \times \underline{F_R}) \cup (E_R \times \underline{F_Q})$,
  - $\mathsf{Act} = \mathsf{Act}_Q \cup \mathsf{Act}_R$,
  - for all $e \in E$, $i \in \{Q, R\}$, $\lambda(e) = \lambda_i(e)$ if $e \in E_i$,
  - $\mathsf{Init} = \mathsf{Init}_Q \cup \mathsf{Init}_R$,
  - for $i \in \{Q, R\}$, $k(e) = k_i(e)$ if $e \in \mathsf{Init}_i$,
  - if $\mathsf{Init}_Q \neq \emptyset$ and $\mathsf{Init}_Q \neq \emptyset$ then for all $e \in E$, $\{e\} \mapsto e$ and for all $e \in F$, $e \rhd \underline{e}$.

  We also have $\langle \mathcal{E}', \mathsf{Init}', k' \rangle$ constructed similarly from some $\left\langle \mathcal{E}'_Q, \mathsf{Init}'_Q, k'_Q \right\rangle$ and $\langle \mathcal{E}'_R, \mathsf{Init}'_R, k'_R \rangle$ such that $\{\!|Q'|\!\} = \left\langle \mathcal{E}'_Q, \mathsf{Init}'_Q, k'_Q \right\rangle$ and $\{\!|R|\!\} = \langle \mathcal{E}'_R, \mathsf{Init}'_R, k'_R \rangle$.

We clearly have isomorphisms $f_Q : \mathcal{E}_Q \to \mathcal{E}'_Q$, $g_Q : \mathcal{E}'_Q \to \mathcal{E}_Q$, $f_R : \mathcal{E}_R \to \mathcal{E}'_R$, and $g_R : \mathcal{E}'_R \to \mathcal{E}_R$ and a transition $\mathsf{Init}_Q \xrightarrow{\{e\}} X$ of $C_{rb}(\mathcal{E}_Q)$ such that $\lambda_Q(e) = \mu$, $k'_Q(f_Q(e)) = m$, and $f(X) = \mathsf{Init}'_Q$. We define our isomorphisms

$$f(e) = \begin{cases} f_Q(e) & \text{if } e \in E_Q \\ f_R(e) & \text{if } e \in E_R \end{cases}$$

and

$$g(e) = \begin{cases} g_Q(e) & \text{if } e \in E'_Q \\ g_R(e) & \text{if } e \in E'_R \end{cases}$$

Since $\mathsf{std}(R)$, $\mathsf{Init}_R = \emptyset$, and therefore $\mathsf{Init} = \mathsf{Init}_Q$, which is conflict-free end therefore a configuration. Obviously $\mathsf{Init} \xrightarrow{\{e\}} X$ in $C_{rb}(\mathcal{E})$, and the rest follows.

- Suppose $P = Q \setminus A$, $P' = Q' \setminus A$, $Q \xrightarrow{\mu[m]} Q'$, and $\mu \notin A \cup \overline{A}$. Then there exists $\mathcal{E}_Q$ such that:
  - $\{\!|Q|\!\} = \langle \mathcal{E}_Q, \mathsf{Init}_Q, k_Q \rangle$
  - $\mathcal{E} = \mathcal{E}_Q \upharpoonright \{e \mid \lambda_Q(e) \notin A \cup \overline{A}\}$
  - $\mathsf{Init} = \mathsf{Init}_Q \cap \{e \mid \lambda_Q(e) \notin A \cup \overline{A}\}$
  - $k = k_Q \upharpoonright \{e \mid \lambda_Q(e) \notin A \cup \overline{A}\}$

  And there exists $\mathcal{E}'_Q$ such that:
  - $\{\!|Q'|\!\} = \langle \mathcal{E}'_Q, \mathsf{Init}'_Q, k'_Q \rangle$
  - $\mathcal{E}' = \mathcal{E}'_Q \upharpoonright \{e \mid \lambda'_Q(e) \notin A \cup \overline{A}\}$
  - $\mathsf{Init}' = \mathsf{Init}'_Q \cap \{e \mid \lambda'_Q(e) \notin A \cup \overline{A}\}$
  - $k' = k'_Q \upharpoonright \{e \mid \lambda'_Q(e) \notin A \cup \overline{A}\}$

  By inductions we have isomorphisms $f_Q : \mathcal{E}_Q \to \mathcal{E}'_Q$ and $g : \mathcal{E}'_Q \to \mathcal{E}_Q$ and a transition $\mathsf{Init}_Q \xrightarrow{\{e_Q\}} X_Q$ or $C_{rb}(\mathcal{E}_Q)$ such that $\lambda_Q(e) = \mu$, $k'_Q(f_Q(e_Q)) = m$, and $f_Q(X_Q) = \mathsf{Init}'_Q$. We define our isomorphisms as $f_Q \upharpoonright \{e \mid \lambda'_Q(e) \notin A \cup \overline{A}\}$ and $g_Q \upharpoonright \{e \mid \lambda'_Q(e) \notin A \cup \overline{A}\}$. And since $\lambda(e) \notin A \cup \overline{A}$, the rest of the proof is straightforward.

- Suppose $P = Q[f']$, $P' = Q'[f']$, $Q \xrightarrow{\nu[m]} Q'$, and $f'(\nu) = \mu$. Then there exist λ_Q and Act_Q such that:
  - $\{\!|Q|\!\} = \langle (E, F, \mapsto, \sharp, \rhd, \lambda_Q, \mathsf{Act}_Q), \mathsf{Init}, k \rangle$,
  - $\mathsf{Act} = f'(\mathsf{Act}_Q)$

– and $\lambda = f' \circ \lambda_Q$.

And there exist λ'_Q and Act'_Q such that

– $\{\!|Q'|\!\} = \left\langle (E', F', \mapsto', \sharp', \rhd', \lambda'_Q, \mathsf{Act}'_Q), \mathsf{Init}', k' \right\rangle$,
– $\mathsf{Act}' = f'(\mathsf{Act}'_Q)$
– and $\lambda = f' \circ \lambda'_Q$.

By induction, we get isomorphisms $f_Q : \mathcal{E}_Q \to \mathcal{E}'_Q$ and $g_Q : \mathcal{E}'_Q \to \mathcal{E}_Q$ and a transition $\mathsf{Init} \xrightarrow{\{e\}} X$ in $C_{rb}(\mathcal{E}_Q)$ such that $\lambda_Q(e) = \nu$, $k'(f'(e)) = m$, and $f(X) = \mathsf{Init}'$. We define our isomorphisms $f = f_Q$ and $g = g_Q$, and the rest of the proof is straightforward.

- Suppose $P \equiv Q$, $P' \equiv Q'$, and $Q \xrightarrow{\mu[m]} Q'$. Then the result follows from induction and Proposition 4.15. □

C.2. Proof of Theorem 4.19

Proof. We say that $\mathcal{E}' = (E', F', \mapsto', \sharp', \rhd', \lambda', \mathsf{Act}')$ and the inverse of f is $g : \mathcal{E}' \to \mathcal{E}$, and prove the theorem by induction on P.

- Suppose $P = 0$. Then $E = \emptyset$, and obviously no transitions exist in $C_{br}(\mathcal{E})$.
- Suppose $P = \alpha.Q$. Then $\{e_\alpha\} \mapsto e'$ for all $e' \in E \setminus \{e_\alpha\}$, meaning by definition $e = e_\alpha$. In addition, since P is reachable, clearly $\mathsf{std}(P)$ meaning $\mathsf{Init} = \emptyset$. This means we get $P \xrightarrow{\alpha[m]} \alpha[m].Q$ for some fresh m, and the isomorphisms are similar to this case in the proof of Theorem 4.18.
- Suppose $P = \alpha[n].Q$ and $\{\!|Q|\!\} = \left\langle \mathcal{E}_Q, \mathsf{Init}_Q, k_Q \right\rangle$. Then $e_\alpha \in \mathsf{Init}$, and clearly $\mathsf{Init}_Q \xrightarrow{e} X_Q$, meaning there exists a key m and a transition $Q \xrightarrow{\lambda(e)[m]} Q'$, such that $\{\!|Q'|\!\} = \left\langle \mathcal{E}'_Q, \mathsf{Init}'_Q, k'_Q \right\rangle$ and there exist isomorphisms $f_Q : \mathcal{E}_Q \to \mathcal{E}'_Q$ and $g_Q : \mathcal{E}'_Q \to \mathcal{E}_Q$ such that $k'_Q(f_Q(e)) = m$ and $f_Q(X_Q) = \mathsf{Init}'_Q$. If $m \neq n$, then $P \xrightarrow{\lambda(e)[m]} \alpha[m].P'_Q$. Otherwise, we can chose a fresh m and still get a transition. We define our isomorphisms as $f = f_Q \cup \{(e_\alpha, e'_\alpha)\}$ and $g = g_Q \cup \{(e'_\alpha, e_\alpha)\}$ and the rest of the proof is straightforward.
- Suppose $P = P_0 + P_1$, $\{\!|P_0|\!\} = \langle \mathcal{E}_0, \mathsf{Init}_0, k_0 \rangle$, $C_{br}(\mathcal{E}_0) = (E_0, F_0, \mathsf{C}_0, \to_0)$, $\{\!|P_1|\!\} = \langle \mathcal{E}_1, \mathsf{Init}_1, k_1 \rangle$, and $C_{br}(\mathcal{E}_1) = (E_1, F_1, \mathsf{C}_1, \to_1)$. Then either $\mathsf{Init}_0 \xrightarrow{e}_0 X_0$ and $\mathsf{Init}_1 = \emptyset$, or $\mathsf{Init}_1 \xrightarrow{e}_1 X_1$ and $\mathsf{Init}_0 = \emptyset$.

  If $\mathsf{Init}_0 \xrightarrow{e}_0 X_0$, then there exists a key m and a transition $P_0 \xrightarrow{\lambda_0(e)[m]} P'_0$, such that $\left\{\!\!\left|P'_0\right|\!\!\right\} = \left\langle \mathcal{E}'_0, \mathsf{Init}'_0, k'_0 \right\rangle$ and there exist isomorphisms $f_0 : \mathcal{E}_0 \to \mathcal{E}'_0$ and $g_0 : \mathcal{E}'_0 \to \mathcal{E}_0$ such that $k'_0(f_0(e)) = m$ and $f_0(X_0) = \mathsf{Init}'_0$. Then, since $\mathsf{Init}_1 = \emptyset$

means $\mathsf{std}(P_1)$, $P \xrightarrow{\lambda(e)[m]} P_0' + P_1$, and the isomorphisms are similar to this case in the proof of Theorem 4.18.

If $\mathsf{Init}_1 \xrightarrow{e}_1 X_1$, then the proof is similar.

- Suppose $P = P_0 \mid P_1$ and we have $\{\!|P_0|\!\} = \langle \mathcal{E}_0, \mathsf{Init}_0, k_0 \rangle$, $\mathcal{E}_0 = (E_0, F_0, \mapsto_0, \sharp_0, \rhd_0, \lambda_0, \mathsf{Act}_0)$, $C_{br}(\mathcal{E}_0) = (E_0, F_0, \mathsf{C}_0, \to_0)$, $\{\!|P_1|\!\} = \langle \mathcal{E}_1, \mathsf{Init}_1, k_1 \rangle$, $\mathcal{E}_1 = (E_1, F_1, \mapsto_1, \sharp_1, \rhd_1, \lambda_1, \mathsf{Act}_1)$, and $C_{br}(\mathcal{E}_1) = (E_1, F_1, \mathsf{C}_1, \to_1)$. Then either $e = (e_0, *)$, $e = (*, e_1)$, or $e = (e_0, e_1)$.

  If $e = (e_0, *)$, then whenever $X_0' \mapsto_0 e_0$, we get $\{e' \in E \mid \pi_0(e') \in X_0'\} \mapsto e$. And whenever $\pi_0(e') \sharp_0 \pi_0(e)$, we get $e' \sharp e$. This means Init_0 is conflict-free, $\pi_0(X)$ is conflict-free, and $\mathsf{Init}_0 \xrightarrow{e_0}_0 \pi_0(X)$. There therefore exists a key m and a transition $P_0 \xrightarrow{\lambda_0(e_0)[m]} P_0'$, such that $\{\!|P_0'|\!\} = \langle \mathcal{E}_0', \mathsf{Init}_0', k_0' \rangle$ and there exist isomorphisms $f_0 : \mathcal{E}_0 \to \mathcal{E}_0'$ and $g_0 : \mathcal{E}_0' \to \mathcal{E}_0$ such that $k_0'(f_0(e_0)) = m$ and $f_0(\pi_0(X)) = \mathsf{Init}_0'$.

  We chose an m, which is fresh for P_1, and we get $P \xrightarrow{\lambda_0(e_0)[m]} P_0' \mid P_1$. We define our isomorphisms

$$f(e') = \begin{cases} (f_0(e_0'), *) & \text{if } e' = (e_0', *) \\ (*, e_1') & \text{if } e' = (*, e_1') \\ (f_0(e_0'), e_1') & \text{if } e' = (e_0', e_1') \end{cases}$$

  and

$$g(e') = \begin{cases} (g_0(e_0'), *) & \text{if } e' = (e_0', *) \\ (*, e_1') & \text{if } e' = (*, e_1') \\ (g_0(e_0'), e_1') & \text{if } e' = (e_0', e_1') \end{cases}$$

  Since $\mathcal{E}' = \mathcal{E}_0' \times \mathcal{E}_1$, these are isomorphisms, and the rest of the case is straightforward.

  If $e = (e_0, *)$, the argument is similar.

  If $e = (e_0, e_1)$, then for $i \in \{0, 1\}$, whenever $X_i' \mapsto_i e_i$, we get $\{e' \in E \mid \pi_i(e') \in X_i'\} \mapsto e$. And whenever $\pi_i(e') \sharp_0 \pi_i(e)$, we get $e' \sharp e$. This means Init_i is conflict-free, $\pi_i(X)$ is conflict-free, and $\mathsf{Init}_i \xrightarrow{e_0}_i \pi_i(X)$. There therefore exists a key m_i and a transition $P_i \xrightarrow{\lambda_i(e_i)[m_i]} P_i'$, such that $\{\!|P_i'|\!\} = \langle \mathcal{E}_i', \mathsf{Init}_i', k_i' \rangle$ and there exist isomorphisms $f_i : \mathcal{E}_0 \to \mathcal{E}_i'$ and $g_i : \mathcal{E}_i' \to \mathcal{E}_i$ such that $k_i'(f_i(e_i)) = m_i$ and $f_i(\pi_i(X)) = \mathsf{Init}_0'$.

  We say that $m_0 = m_1$ is a fresh m, and then since $\lambda_0(e_0) = \overline{\lambda_1(e_1)}$ and $\lambda(e) = \tau$, we get $P \xrightarrow{\lambda(e)[m]} P_0' \mid P_1'$. We define our isomorphisms

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

and

$$g(e') = \begin{cases} (g_0(e'_0), *) & \text{if } e' = (e'_0, *) \\ (*, g_1(e'_1)) & \text{if } e' = (*, e'_1) \\ (g_0(e'_0), g_1(e'_1)) & \text{if } e' = (e'_0, e'_1) \end{cases}$$

Since $\mathcal{E}' = \mathcal{E}'_0 \times \mathcal{E}'_1$, these are isomorphisms, and the rest of the case is straightforward.

- Suppose $P = Q \setminus A$, $[\![Q]\!] = \langle \mathcal{E}_Q, \mathsf{Init}, k \rangle$, and $C_{br}(\mathcal{E}_Q) = (E_Q, F_Q, \mathsf{C}_Q, \rightarrow_Q)$. Then $\lambda(e) \notin A \cup \overline{A}$ and $\mathsf{Init} \xrightarrow{e}_Q X$, meaning there exists a key n and a transition $Q \xrightarrow{\lambda_Q(e)} Q'$ such that $[\![Q']\!] = \left\langle \mathcal{E}'_Q, \mathsf{Init}'_Q, k'_Q \right\rangle$, and there exist isomorphisms $f_Q : \mathcal{E}_Q \to \mathcal{E}'_Q$ and $g_Q : \mathcal{E}'_Q \to \mathcal{E}_Q$ such that $f_Q \circ k'_Q = k_Q \cup \{(e, n)\}$ and $f_Q(X) = \mathsf{Init}'_Q$.

  This means $P \xrightarrow{\lambda_Q(e)} Q' \setminus A$ and the morphisms $f \upharpoonright E$ and $g \upharpoonright \{e' \in E'_Q \mid \lambda'_Q(e') \notin A \cup \overline{A}\}$ clearly fulfil the remaining conditions.

- Suppose $P = Q[f]$, $[\![Q]\!] = \langle \mathcal{E}_Q, \mathsf{Init}, k \rangle$, and $C_{br}(\mathcal{E}_Q) = (E_Q, F_Q, \mathsf{C}_Q, \rightarrow_Q)$. Clearly $\mathsf{Init} \xrightarrow{e}_Q X$, and $f(\lambda_Q(e)) = \lambda(e)$, and the proof is straightforward. □

D. Extended bundle event structures

In this appendix we define the categorical concepts for EBESs, which we defined for BESs in Section 3.1.

We first define a finitely caused subcategory (Definition D.1), though in this case the asymmetric conflict still cannot be modelled by general event structures.

Definition D.1 (Finitely Caused EBES). A finitely caused EBES (FCEBES) is an EBES $\mathcal{E} = (E, \mapsto, \rhd)$ where for any $e \in E$, $\{X \subseteq E \mid X \mapsto e\}$ is finite.

EBES configurations are sets which have an order in which the events could have happened.

Definition D.2 (EBES configuration [21]). Given an EBES $\mathcal{E} = (E, \mapsto, \rhd)$, a configuration of $\mathcal{E}$ is a set $X \subseteq E$ such that there exists a sequence $e_0, \ldots, e_n$ such that:

1. $\{e_0, \ldots, e_n\} = X$;
2. if $e_i \lhd e_j$ then $i < j$;
3. if $X \mapsto e_i$ then $X \cap \{e_0, \ldots, e_{i-1}\} \neq \emptyset$.

A category of EBESs has not, to our knowledge, been defined, so we define an EBES morphism in Definition D.3. This morphism is similar to the BES morphism defined previously, and asymmetric conflict is treated much the same way as symmetric.

Definition D.3 (EBES morphism). Given EBESs $\mathcal{E}_0 = (E_0, \mapsto_0, \rhd_0)$ and $\mathcal{E}_1 = (E_1, \mapsto_1, \rhd_1)$, an EBES-morphism from $\mathcal{E}_0$ to $\mathcal{E}_1$ is a partial function $f : E_0 \to E_1$ such that and for all $e, e' \in E_0$:

1. if $f(e) \lhd_1 f(e')$ then $e \lhd_0 e'$;
2. if $f(e) = f(e') \neq \bot$ and $e \neq e'$ then $e \lhd_0 e'$;
3. for $X_1 \subseteq E_1$ if $X_1 \mapsto_1 f(e)$ then there exists $X_0 \subseteq E_0$ such that $X_0 \mapsto_0 e$, $f(X_0) \subseteq X_1$, and if $e' \in X_0$ then $f(e') \neq \bot$;
4. for any $X_0 \subseteq E_0$, if X_0 is a configuration of $\mathcal{E}_0$, then $f(X_0)$ is a configuration of $\mathcal{E}_1$.

Proposition D.4. *Given EBESs $\mathcal{E}_0 = (E_0, \mapsto_0, \rhd_0)$ and $\mathcal{E}_1 = (E_1, \mapsto_1, \rhd_1)$ and EBES morphism $f : E_0 \to E_1$, if $X \subseteq E_0$ is a configuration of $\mathcal{E}_0$, then $f(X)$ is a configuration of $\mathcal{E}_1$.*

Proposition D.5. ***EBES*** *consisting of EBESs and EBES morphisms is a category.*

Proof. Composition of partial functions is associative and $f(e) = e$ works as an identity arrow, and so we need only show that the morphisms are composable.

If $\mathcal{E}_0 = (E_0, \mapsto_0, \rhd_0)$, $\mathcal{E}_1 = (E_1, \mapsto_1, \rhd_1)$, and $\mathcal{E}_2 = (E_2, \mapsto_2, \rhd_2)$ are EBESs and $f : E_0 \to E_1$ and $g : E_1 \to E_2$ are morphisms, we show that $f \circ g : E_0 \to E_2$ is also a morphism:

1. If $g(f(e)) \lhd_2 g(f(e'))$ then $f(e) \lhd_1 f(e')$, and therefore $e \lhd_0 e'$.
2. If $g(f(e)) = g(f(e'))$ and $e \neq e'$, then either $f(e) = f(e')$, in which case $e \lhd_0 e'$, or $f(e) \neq f(e')$, in which case $f(e) \lhd_1 f(e')$, and therefore $e \lhd_0 e'$.
3. If $X_2 \mapsto_2 g(f(e))$ then there exist $X_1 \subseteq E_1$ and $X_0 \subseteq E_0$ such that $X_1 \mapsto_1 f(e)$, $X_0 \mapsto_0 e$, $g(X_1) \subseteq X_2$, $f(X_0) \subseteq X_1$ and if $e_1 \in X_1$ then $g(e_1) \neq \bot$ and if $e_0 \in X_0$ then $f(e_0) \neq \bot$. This means that $g(f(X_0)) \subseteq X_2$, and if $e_0 \in X_0$ then $g(f(e_0)) \neq \bot$.
4. If X_0 is a configuration of $\mathcal{E}_0$ then $f(X_0)$ is a configuration of $\mathcal{E}_1$, and therefore $g(f(X_0))$ is a configurations of $\mathcal{E}_2$. □

We also construct a product of EBESs in Definition D.6. This definition is very similar to the products in Section 3.

Definition D.6 (EBES product). Given EBESs $\mathcal{E}_0 = (E_0, \mapsto_0, \rhd_0)$ and $\mathcal{E}_1 = (E_1, \mapsto_1, \rhd_1)$, we construct $\mathcal{E}_0 \times \mathcal{E}_1 = (E, \mapsto, \rhd)$ with projections π_0, π_1 where:

1. $E = E_0 \times_* E_1 = \{(e, *) \mid e \in E_0\} \cup \{(*, e) \mid e \in E_1\} \cup \{(e, e') \mid e \in E_0 \text{ and } e' \in E_1\}$;
2. for $(e_0, e_1) \in E$, $\pi_i(e_0, e_1) = e_i$;
3. for any $e \in E$, $X \subseteq E$, $X \mapsto e$ iff there exists $i \in \{0, 1\}$ and $X_i \subseteq E_i$ such that $X_i \mapsto \pi_i(e)$ and $X = \{e' \in E \mid \pi_i(e') \in X_i\}$;

4. for any $e, e' \in E$, $e \lhd e'$ iff there exists $i \in \{0,1\}$ such that $\pi_i(e) \lhd_i \pi_i(e')$, or $\pi_i(e) = \pi_i(e') \neq \bot$ and $\pi_{1-i}(e) \neq \pi_{1-i}(e')$.

Proposition D.7. *Given EBESs $\mathcal{E}_0 = (E_0, \mapsto_0, \rhd_0)$ and $\mathcal{E}_1 = (E_1, \mapsto_1, \rhd_1)$, we have that $\mathcal{E}_0 \times \mathcal{E}_1 = (E, \mapsto, \rhd)$ is their product.*

Proof. We first show that π_0 and π_1 are morphisms:

1. If $\pi_i(e) \lhd_i \pi_i(e')$, then obviously $e \lhd e'$.
2. If $\pi_i(e) = \pi_i(e')$ and $e \neq e'$, then $\pi_{1-i}(e) \neq \pi_{1-i}(e')$, and therefore $e \lhd e'$.
3. If $X_i \mapsto \pi_i(e)$, then $\{e' \in E \mid \pi_i(e') \in X_i\} \mapsto e$. Clearly $\pi_i(\{e' \in E \mid \pi_i(e') \in X_i\}) = X_i$, and for all $e' \in \{e' \in E \mid \pi_i(e') \in X_i\})$, $\pi_i(e') \neq \bot$.
4. If X is a configuration of $\mathcal{E}_0 \times \mathcal{E}_1$, then we show that $\pi_i(X)$ satisfies the requirements of a configuration of $\mathcal{E}_i$. We show that if the requirements of Definition D.2 hold for $e_0, \dots e_n$, then they hold for $\pi_i(e_0), \dots, \pi_i(e_n)$:
   (a) Obviously $\{\pi_i(e_0), \dots, \pi_i(e_n)\} = \pi_i(X)$.
   (b) If $\pi_i(e_j) \lhd_i \pi_i(e_{j'})$, then as shown above, $e_j \lhd e_{j'}$, meaning $j < j'$.
   (c) Whenever $Y_i \mapsto \pi_i(e_{j+1})$, we know $\{e' \in E \mid \pi_i(e') \in Y_i\} \mapsto e_{j+1}$, meaning $\{e' \in E \mid \pi_i(e') \in Y_i\} \cap \{e_1, \dots, e_j\} \neq \emptyset$. Therefore, we must get $Y_i \cap \{\pi_i(e_1), \dots, \pi_i(e_j)\} \neq \pi_i(\emptyset) = \emptyset$.

We then show that for any EBES, $\mathcal{E}_2 = (E_2, \mapsto_2, \rhd_2)$, if there exist morphisms $f_0 : \mathcal{E}_2 \to \mathcal{E}_0$ and $f_1 : \mathcal{E}_2 \to \mathcal{E}_1$, then there exists a unique morphism $f : \mathcal{E}_2 \to \mathcal{E}$, such that $\pi_0 \circ f = f_0$ and $\pi_1 \circ f = f_1$.

Clearly $f(e) = (f_0(e), f_1(e))$ is the only partial function for which this commutes, meaningthat the morphisms clearly commute as described above; we prove it to be a morphism:

1. If $f(e) \lhd f(e')$ then there exists $i \in \{0,1\}$ such that either $\pi_i(f(e)) \lhd_i \pi_i(f(e'))$, in which case clearly $f_i(e) \lhd_i f_i(e')_i$, and therefore $e \lhd_2 e'$, or $\pi_i(f(e)) = \pi_i(f(e')) \neq \bot$ and $\pi_{1-i}(f(e)) \neq \pi_{1-i}(f(e'))$, in which case $f_i(e) = f_i(e') \neq \bot$, and $e \neq e'$, meaning $e \lhd_2 e'$.
2. If $f(e) = f(e') \neq \bot$ then $f_0(e) = f_0(e') \neq \bot$ or $f_1(e) = f_1(e') \neq \bot$, meaning if $e \neq e'$ then $e \lhd_2 e'$.
3. For $X \subseteq E$, if $X \mapsto f(e)$, then there exists $i \in \{0,1\}$ and $X_i \subseteq E_i$ such that $X_i \mapsto \pi_i(e)$ and $X = \{e' \in E \mid \pi_i(e') \in X_i\}$. And since $X_i \mapsto f_i(e)$, there exists $X_2 \subseteq E_2$ such that $X_2 \mapsto_2 e$, $f_i(X_2) \subseteq X_i$, and if $e' \in X_2$ then $f_i(e') \neq \bot$. Clearly $f(X_2) \subseteq X$.
4. If X is a configuration of $\mathcal{E}_2$, then we show that $f(X)$ satisfies the requirements of a configuration of $\mathcal{E}_0 \times \mathcal{E}_1$. We show that if the requirements of Definition D.2 hold for $e_0, \dots e_n$, then they hold for $f(e_0), \dots, f(e_n)$:
   (a) Obviously $\{f(e_0), \dots, f(e_n)\} = f(X)$.
   (b) If $f(e_j) \lhd f(e_{j'})$, then as shown above, $e_j \lhd_2 e_{j'}$, meaning $j < j'$.
   (c) Whenever $Y \mapsto f(e_{j+1})$, we know $\{e' \in E \mid f(e') \in Y\} \mapsto e_{j+1}$, meaning $\{e' \in E \mid f(e') \in Y\} \cap \{e_1, \dots, e_j\} \neq \emptyset$. Therefore, we must get $Y \cap \{f(e_1), \dots, f(e_j)\} \neq f(\emptyset) = \emptyset$. □

Proposition D.8. *Given FCEBESs $\mathcal{E}_0 = (E_0, \mapsto_0, \rhd_0)$ and $\mathcal{E}_1 = (E_1, \mapsto_1, \rhd_1)$, we have that $\mathcal{E}_0 \times \mathcal{E}_1 = (E, \mapsto, \rhd)$ is an FCEBES.*

Proof. Similar to the proof of Proposition 3.11. □

We also construct a coproduct of EBESs in Definition D.9.

Definition D.9 (EBES coproduct). Given EBESs $\mathcal{E}_0 = (E_0, \mapsto_0, \rhd_0)$ and $\mathcal{E}_1 = (E_1, \mapsto_1, \rhd_1)$, we construct $\mathcal{E}_0 + \mathcal{E}_1 = (E, \mapsto, \rhd)$ with injections ι_0, ι_1 where:

- $E = \{(0, e) \mid e \in E_0\} \cup \{(1, e) \mid e \in E_1\}$;
- for $e \in E_j$, $\iota_j(e) = (j, e)$ for $j \in \{0, 1\}$;
- $X \mapsto (j, e)$ iff for all $(j', e') \in X$, $j = j'$ and $\iota_j(X) \mapsto_j e$;
- $(j, e) \lhd (j', e')$ iff $j \neq j'$ or $e \lhd_j e'$.

Proposition D.10. *If $\mathcal{E}_0$ and $\mathcal{E}_1$ are EBESs, then $\mathcal{E}_0 + \mathcal{E}_1$ is their coproduct.*

Proof. Similar to that for BES coproduct (Proposition 3.13). □

Proposition D.11. *Given FCEBESs $\mathcal{E}_0 = (E_0, \mapsto_0, \rhd_0)$ and $\mathcal{E}_1 = (E_1, \mapsto_1, \rhd_1)$, we have that $\mathcal{E}_0 + \mathcal{E}_1 = (E, \mapsto, \rhd)$ is an FCEBES.*

Proof. Similar to Proposition 3.27. □

E. Relating REBES to other categories of reversible event structures

In this appendix we recall the category of reversible asymmetric event structures and describe how it and stable bundle event structures (Definition A.4) relate to the category **REBES** defined in Section 5.

Definition E.1 (RAES [27]). A reversible asymmetric event structure (RAES) is a 4-tuple $\mathcal{E} = (E, F, \prec, \lhd)$ where E is the set of events and

1. $F \subseteq E$ is the set of reversible events;
2. $\lhd \subseteq (E \cup \underline{F}) \times E$ is the irreflexive precedence relation;
3. $\prec \subseteq E \times (E \cup \underline{F})$ is the causation relation, which is irreflexive and well-founded, such that for all $\alpha \in E \cup \underline{F}$, $\{e \in E \mid e \prec \alpha\}$ is finite and has no $\lhd$-cycles;
4. for all $e \in F$, $e \prec \underline{e}$;
5. for all $e \in E$ and $\alpha \in E \cup \underline{F}$ if $e \prec \alpha$ then not $e \rhd \alpha$;
6. $e \lessdot e'$ implies $e \lhd e'$, where $e \lessdot e'$ means that $e \prec e'$ and if $e \in F$ then $e' \rhd \underline{e}$;
7. $\lessdot$ is transitive;
8. if $e \sharp e'$ and $e \lessdot e''$ then $e'' \sharp e'$, where $\sharp = \lhd \cap \rhd$.

Definition E.2 (RAES morphism [15]). Given RAESs $\mathcal{E}_0 = (E_0, F_0, \prec_0, \lhd_0)$ and $\mathcal{E}_1 = (E_1, F_1, \prec_1, \lhd_1)$, an RAES morphism $f : \mathcal{E}_0 \to \mathcal{E}_1$ is a partial function $f : E_0 \to E_1$ such that

1. for all $e^* \in E_0 \cup \underline{F_0}$, if $f(e) \neq \bot$ then $\{e_1 \mid e_1 \prec_1 f(e^*)\} \subseteq \{f(e') \mid e' \prec_0 e^*\}$;

2. for all $e \in E_0$ and $e'^* \in E_0 \cup \underline{F_0}$, if $f(e) \neq \bot \neq f(e'^*)$ and $f(e'^*) \lhd_1 f(e)$ then $e'^* \lhd_0 e$;
3. for all $e, e' \in E_0$, if $f(e) = f(e') \neq \bot$ and $e \neq e'$ then $e \sharp_0 e'$;
4. $f(F_0) \subseteq F_1$.

Definition E.3 (RAES to REBES). The functor $B_{ar} : \mathbf{RAES} \to \mathbf{REBES}$ is defined as:

1. $B_{ar}((E, F, \prec, \lhd)) = (E, F, \mapsto, \rhd)$, where $\{e'\} \mapsto e^*$ if $e' \prec e$;
2. $B_{ar}(f) = f$.

Definition E.4 (FCREBES to SRES). The functor $E_{er} : \mathbf{FCREBES} \to \mathbf{RES}$ is defined as:

1. $E_{er}((E, F, \mapsto, \rhd)) = (E, F, \mathsf{Con}, \vdash)$ where
   (a) Con is the set of finite subsets of E with no $\lhd$-cycles;
   (b) For $e^* \in E \cup \underline{F}$, $X \otimes Y \vdash^*$ if $X \in \mathsf{Con}$, for all $X' \subseteq E$ such that $X' \mapsto e^*$, $X' \cap X \neq \emptyset$, $Y = \{e' \mid e' \rhd e^*\}$, and $Y \cap X = \emptyset$;
2. $E_{er}(f) = f$.

Proposition E.5. *Given an FCREBES $\mathcal{E} = (E, F, \mapsto, \rhd)$, we have that $E_{er}((E, F, \prec, \sharp, \rhd))$ is an SRES.*

Proof. If $X \otimes Y \vdash e^*$, $X' \otimes Y' \vdash e^*$, and $X \cup X' + e^* \in \mathsf{Con}$ then $Y = Y'$ and $X \cup X'$ has no $\lhd$-cycles, meaning there must exist an $X'' \subseteq (X \cap X')$ such that for all $X''' \mapsto e^*$, there exists $e' \in X'' \cap X'''$. This implies $X \cap X' \otimes Y \cap Y' \vdash e^*$. □

F. Proofs from Section 7

F.1. Proof of Proposition 7.4

Lemma F.1. *Given consistent processes P_0 and P_1 such that $\{\!|P_0|\!\} = \langle \mathcal{E}_0, \mathsf{Init}_0, k_0 \rangle$, $\{\!|P_1|\!\} = \langle \mathcal{E}_1, \mathsf{Init}_1, k_1 \rangle$, and $\mathcal{E}_0 \leq \mathcal{E}_1$, and there exists $A\langle \tilde{\alpha}, \tilde{\gamma} \rangle$ in P_0 such that $A(\tilde{\beta}, \tilde{\delta}) = P_A$ and $P_1 = P_0\{^{A\langle \tilde{\alpha}, \tilde{\gamma} \rangle}/_{P_A\{^{\tilde{\alpha}, \tilde{\gamma}}/_{\tilde{\beta}, \tilde{\delta}}\}}\}$, and an action α_γ such that $\left\{\!\left|\alpha_\gamma.P_0\right|\!\right\} = \left\langle \mathcal{E}'_0, \mathsf{Init}'_0, k'_0 \right\rangle$ and $\left\{\!\left|\alpha_\gamma.P_1\right|\!\right\} = \left\langle \mathcal{E}'_1, \mathsf{Init}'_1, k'_1 \right\rangle$, we get $\mathcal{E}'_0 \leq \mathcal{E}'_1$.*

Proof. Obviously $E'_0 \subseteq E'_1$ and $F'_0 = F'_1 \cap E'_0$.

We then prove that $X \mapsto'_0 e^*$ if and only if $X' \mapsto'_1 e^*$, $X = X' \cap E'_0$, and $e^* \in E'_0 \cup \underline{F'_0}$:

- If $X \mapsto'_0 e$ then either $X \mapsto_0 e$, or $X = \{e_\alpha\}$, $e \in E_0$, and $\lambda_0(e) \neq \mathsf{roll}\ \gamma'$.
  - If $X \mapsto_0 e$ then there exists some $X_1 \subseteq E_1$ such that $X_1 \cap E_0 = X_0$ and $X_1 \mapsto_1 e$, meaning $X_1 \mapsto'_1 e$ and $X_1 \cap E'_0 = X_0$.
  - If $X = \{e_\alpha\}$, $e \in E_0$, and $\lambda_0(e) \neq \mathsf{roll}\ \gamma'$ then $e \in E_1$ and $\lambda_1(e) = \lambda_0(e) \neq \mathsf{roll}\ \gamma'$, meaning $\{e_\alpha\} \mapsto'_1 e$.

- If $X \mapsto_1' e$ and $e \in E_0'$ then either $X \mapsto_1 e$, or $X = \{e_\alpha\}$, $e \in E_1$, and $\lambda_1(e) \neq \mathsf{roll}\ \gamma'$.
  - If $X \mapsto_1 e$ and $e \in E_0'$ then $X \cap E_0 = X \cap E_0' \mapsto_0 e$, meaning $X \cap E_0' \mapsto_0 e$.
  - If $e \in E_0'$, $X = \{e_\alpha\}$, $e \in E_1$, and $\lambda_1(e) \neq \mathsf{roll}\ \gamma'$, then $\lambda_0(e) = \lambda_1(e) \neq \mathsf{roll}\ \gamma$, meaning $\{e_\alpha\} \mapsto_0' e$.
- If $X \mapsto_0' \underline{e}$ then either $X = \{e\}$, or $e = e_\alpha$ and $X = \{e' \in E_0 \mid \{e' \mid \lambda_0(e') = \mathsf{roll}\ \gamma$ and $e' \in X' \mapsto e'' \Rightarrow \lambda(e'') = \mathsf{start\ roll}\ \gamma\}\}$, or $\lambda_0(e) \in \{\mathsf{roll}\ \gamma' \mid \gamma'$ is a tag $\} \cup \{\mathsf{start\ roll}\ \gamma' \mid \nexists \beta, n.\beta_{\gamma'}$ or $\beta_{\gamma'}[n]$ occurs in $\alpha_\gamma.P_0\}$ and $X \mapsto_0 \underline{e}$, or $\lambda_0(e) \notin \{\mathsf{roll}\ \gamma' \mid \gamma'$ is a tag $\} \cup \{\mathsf{start\ roll}\ \gamma' \mid \nexists \beta, n.\beta_{\gamma'}$ or $\beta_{\gamma'}[n]$ occurs in $\alpha_\gamma.P_0\}$, $\{e\} \neq X' \mapsto_0 \underline{e}$, and $X = X' \cup \{e' \mid \lambda_0(e') = \mathsf{roll}\ \gamma\}$.
  - If $X = \{e\}$ then obviously $X \mapsto_1' e$.
  - If $e = e_\alpha$ and $X = \{e' \in E_0 \mid \lambda_0(e') = \mathsf{roll}\ \gamma\}$ then $X = \{e' \in E_1 \mid \lambda_1(e') = \mathsf{roll}\ \gamma\} \cap E_0$ and $\{e' \in E_1 \mid \lambda_1(e') = \mathsf{roll}\ \gamma\} \mapsto_1' e$.
  - If $\lambda_0(e) \in \{\mathsf{roll}\ \gamma' \mid \gamma'$ is a tag $\} \cup \{\mathsf{start\ roll}\ \gamma' \mid \nexists \beta, n.\beta_{\gamma'}$ or $\beta_{\gamma'}[n]$ occurs in $\alpha_\gamma.P_0\}$ and $X \mapsto_0 \underline{e}$ then $\lambda_1(e) \in \{\mathsf{roll}\ \gamma' \mid \gamma'$ is a tag$\} \cup \{\mathsf{start\ roll}\ \gamma' \mid \nexists \beta, n.\beta_{\gamma'}$ or $\beta_{\gamma'}[n]$ occurs in $\alpha_\gamma.P_1\}$ and there exists X_1 such that $X_1 \cap E_0 = X$ and $X_1 \mapsto_1 \underline{e}$, meaning $X_1 \mapsto_1' \underline{e}$.
  - If $\lambda_0(e) \notin \{\mathsf{roll}\ \gamma' \mid \gamma'$ is a tag $\} \cup \{\mathsf{start\ roll}\ \gamma' \mid \nexists \beta, n.\beta_{\gamma'}$ or $\beta_{\gamma'}[n]$ occurs in $\alpha_\gamma.P_0\}$, $\{e\} \neq X' \mapsto_0 \underline{e}$, and $X = X' \cup \{e' \mid \lambda_0(e') = \mathsf{roll}\ \gamma\}$ then there exists $X_1' \subseteq E_1$ such that $X_1' \cap E_0 = X'$ and $X_1' \mapsto_1 e$. This means $X_1' \cup \{e' \in E_1 \mid \lambda_1(e') = \mathsf{roll}\ \gamma\} \mapsto_1' e$, and clearly $\{e' \in E_1 \mid \lambda_1(e') = \mathsf{roll}\ \gamma\} \cap E_0 = \{e' \in E_0 \mid \lambda_0(e') = \mathsf{roll}\ \gamma\}$, meaning $(X_1' \cup \{e' \in E_1 \mid \lambda_1(e') = \mathsf{roll}\ \gamma\}) \cap E_0' = X$.
- If $X \mapsto_1' \underline{e}$ and $e \in E_0'$ then either $X = \{e\}$, or $e = e_\alpha$ and $X = \{e' \in E_1 \mid \lambda_1(e') = \mathsf{roll}\ \gamma\}$, or $\lambda_1(e) \in \{\mathsf{roll}\ \gamma' \mid \gamma'$ is a tag $\} \cup \{\mathsf{start\ roll}\ \gamma' \mid \nexists \beta, n.\beta_{\gamma'}$ or $\beta_{\gamma'}[n]$ occurs in $\alpha_\gamma.P_1\}$ and $X \mapsto_1 \underline{e}$, or $\lambda_1(e) \notin \{\mathsf{roll}\ \gamma' \mid \gamma'$ is a tag $\} \cup \{\mathsf{start\ roll}\ \gamma' \mid \nexists \beta, n.\beta_{\gamma'}$ or $\beta_{\gamma'}[n]$ occurs in $\alpha_\gamma.P_1\}$, $\{e\} \neq X' \mapsto_1 \underline{e}$, and $X = X' \cup \{e' \mid \lambda_1(e') = \mathsf{roll}\ \gamma\}$.
  - If $X = \{e\}$ then obviously $X \mapsto_0' e$.
  - If $e = e_\alpha$ and $X = \{e' \in E_1 \mid \lambda_1(e') = \mathsf{roll}\ \gamma\}$ then $\{e' \in E_0 \mid \lambda_0(e') = \mathsf{roll}\ \gamma\} \mapsto_0' \underline{e}$ and obviously $X \cap E_0' = \{e' \in E_0 \mid \lambda_0(e') = \mathsf{roll}\ \gamma\}$.
  - If $\lambda_1(e) \in \{\mathsf{roll}\ \gamma' \mid \gamma'$ is a tag$\} \cup \{\mathsf{start\ roll}\ \gamma' \mid \nexists \beta, n.\beta_{\gamma'}$ or $\beta_{\gamma'}[n]$ occurs in $\alpha_\gamma.P_1\}$ and $X \mapsto_1 \underline{e}$ then $\lambda_0(e) \in \{\mathsf{roll}\ \gamma' \mid \gamma'$ is a tag$\} \cup \{\mathsf{start\ roll}\ \gamma' \mid \nexists \beta, n.\beta_{\gamma'}$ or $\beta_{\gamma'}[n]$ occurs in $\alpha_\gamma.P_0\}$, and $X \cap E_0 \mapsto_0 \underline{e}$, meaning $X \cap E_0 \mapsto_0' \underline{e}$.
  - If $\lambda_1(e) \notin \{\mathsf{roll}\ \gamma' \mid \gamma'$ is a tag$\} \cup \{\mathsf{start\ roll}\ \gamma' \mid \nexists \beta, n.\beta_{\gamma'}$ or $\beta_{\gamma'}[n]$ occurs in $\alpha_\gamma.P_1\}$, $\{e\} \neq X' \mapsto_1 \underline{e}$, and $X = X' \cup \{e' \mid \lambda_1(e') = \mathsf{roll}\ \gamma\}$ then $X' \cap E_0 \mapsto_0 \underline{e}$, meaning $(X' \cap E_0) \cup \{e' \mid \lambda_0(e') = \mathsf{roll}\ \gamma\} \mapsto_0' \underline{e}$, and obviously $X \cap E_0 = (X' \cap E_0) \cup \{e' \mid \lambda_0(e') = \mathsf{roll}\ \gamma\}$.

We then prove that $e \rhd'_0 e'^*$ if and only if $e \rhd'_1 e'^*$, $e \in E'_0$, and $e' \in E'_0 \cap \underline{F'_0}$.

- If $e \rhd'_0 e'^*$ then either $\lambda_0(e) \in \{\mathsf{roll}\ \gamma' \mid \gamma' \text{ is a tag}\} \cup \{\mathsf{start\ roll}\ \gamma' \mid \nexists \beta, n.\beta_{\gamma'}$ or $\beta_{\gamma'}[n]$ occurs in $\alpha_\gamma.P_0\}$ and $e'^* = \underline{e_\alpha}$, or $e = e_\alpha$, $e'^* = \underline{e'}$ and $\lambda_0(e') = \mathsf{roll}\ \gamma$, or $\lambda_0(e) = \mathsf{roll}\ \gamma$ and $e'^* = e_\alpha$, or $\lambda_0(e) = \mathsf{roll}\ \gamma$, $e'^* = e'$, and $\lambda_0(e') \notin \{\mathsf{roll}\ \gamma' \mid \gamma'$ is a tag $\} \cup \{\mathsf{start\ roll}\ \gamma' \mid \nexists \beta, n.\beta_{\gamma'}$ or $\beta_{\gamma'}[n]$ occurs in $\alpha_\gamma.P_0\}$.
  - If $\lambda_0(e) \in \{\mathsf{roll}\ \gamma' \mid \gamma'$ is a tag $\} \cup \{\mathsf{start\ roll}\ \gamma' \mid \nexists \beta, n.\beta_{\gamma'}$ or $\beta_{\gamma'}[n]$ occurs in $\alpha_\gamma.P_0\}$ and $e'^* = \underline{e_\alpha}$, then $\lambda_1(e) \in \{\mathsf{roll}\ \gamma' \mid \gamma'$ is a tag $\} \cup \{\mathsf{start\ roll}\ \gamma' \mid \nexists \beta, n.\beta_{\gamma'}$ or $\beta_{\gamma'}[n]$ occurs in $\alpha_\gamma.P_1\}$, and therefore $e \rhd'_1 \underline{e_\alpha}$.
  - If $e = e_\alpha$, $e'^* = \underline{e'}$ and $\lambda_0(e') = \mathsf{roll}\ \gamma$ then $\lambda_1(e') = \mathsf{roll}\ \gamma$, and therefore $e_\alpha \rhd'_1 e'$.
  - If $\lambda_0(e) = \mathsf{roll}\ \gamma$ and $e'^* = e_\alpha$, then $\lambda_1(e) = \mathsf{roll}\ \gamma$, and therefore $e \rhd'_1 e_\alpha$.
  - If $\lambda_0(e) = \mathsf{roll}\ \gamma$, $\lambda_0(e') \notin \{\mathsf{roll}\ \gamma' \mid \gamma'$ is a tag $\} \cup \{\mathsf{start\ roll}\ \gamma' \mid \nexists \beta, n.\beta_{\gamma'}$ or $\beta_{\gamma'}[n]$ occurs in $\alpha_\gamma.P_0\}$, and $e'^* = e'$, then $\lambda_1(e) = \mathsf{roll}\ \gamma$ and $\lambda_1(e') \notin \{\mathsf{roll}\ \gamma' \mid \gamma'$ is a tag $\} \cup \{\mathsf{start\ roll}\ \gamma' \mid \nexists \beta, n.\beta_{\gamma'}$ or $\beta_{\gamma'}[n]$ occurs in $\alpha_\gamma.P_0\}$, meaning $e \rhd'_1 e'$.
- If $e \rhd'_1 e'^*$ for $e, e' \in E_0$ then the argument is similar.

Obviously $\lambda'_0 = \lambda'_1 \restriction_{E'_0}$ and $\mathsf{Act} = \mathsf{ran}(\lambda'_0)$. □

Lemma F.2. *Given consistent processes P_0 and P_1 such that $\{\!|P_0|\!\} = \langle \mathcal{E}_0, \mathsf{Init}_0, k_0 \rangle$, $\{\!|P_1|\!\} = \langle \mathcal{E}_1, \mathsf{Init}_1, k_1 \rangle$, and $\mathcal{E}_0 \leq \mathcal{E}_1$, and an action α_γ such that $\{\!|\alpha_\gamma[m].P_0|\!\} = \langle \mathcal{E}'_0, \mathsf{Init}'_0, k'_0 \rangle$ and $\{\!|\alpha_\gamma[m].P_1|\!\} = \langle \mathcal{E}'_1, \mathsf{Init}'_1, k'_1 \rangle$, we get $\mathcal{E}'_0 \leq \mathcal{E}'_1$.*

Proof. Follows from Lemma F.1 and the definitions of $\{\!|\alpha_\gamma[m].P_0|\!\}$ and $\{\!|\alpha_\gamma[m].P_1|\!\}$. □

Lemma F.3. *Given consistent processes $P_0 \mid P_2$, $P_1 \mid P_2$ with $\{\!|P_0|\!\} = \langle \mathcal{E}_0, \mathsf{Init}_0, k_0 \rangle$, $\{\!|P_1|\!\} = \langle \mathcal{E}_1, \mathsf{Init}_1, k_1 \rangle$, $\{\!|P_0 \mid P_2|\!\} = \langle \mathcal{E}'_0, \mathsf{Init}'_0, k'_0 \rangle$, $\{\!|P_1 \mid P_2|\!\} = \langle \mathcal{E}'_1, \mathsf{Init}'_1, k'_1 \rangle$, and $\mathcal{E}_0 \leq \mathcal{E}_1$, we get $\mathcal{E}'_0 \leq \mathcal{E}'_1$.*

Proof. We first prove that $E'_0 \subseteq E'_1$. For all $e \in E'_0$ we know either $e \in E_0 \times_* E_2$ and $\lambda'_0(e) \in \{\mathsf{roll}\ \gamma, \mathsf{start\ roll}\ \gamma\}$, or $e = (X, e')$ for some $e' \in E_0 \times_* E_2$ and $X \in \mathsf{causes}(e')$. If $e \in E_0 \times_* E_2$ and $\lambda'_0(e) \in \{\mathsf{roll}\ \gamma, \mathsf{start\ roll}\ \gamma\}$ then obviously $e \in E'_1$. If $e = (X, e')$ then if $e' = (e_0, *)$ then for each $(e'_0, e'_2) \in X$ there exists X_0 such that $e'_0 \in X_0$ and $X_0 \mapsto_0 e_0$. This means there exists $X_1 \subseteq E_1$ such that $X_1 \mapsto_1 e_0$ and $X_0 = X_1 \cap E_0$. In addition, for any $X'_1 \subseteq E_1$ such that $X'_1 \mapsto_1 e_0$, we have $X'_1 \cap E_0 \mapsto_0 e_0$, and therefore $(X'_1 \times E_2) \cap X \neq \emptyset$. We therefore get $e \in E'_1$. If $e' = (*, e_2)$ then obviously e_2's causes are the same in $\mathcal{E}'_1$ and therefore $e' \in E'_1$. If $e' = (e_0, e_2)$ then the argument is a combination of the first two cases.

Obviously $F'_0 = E'_0 \cap F'_1$.

We then prove that $X \mapsto'_0 e^*$ if and only if $X' \mapsto'_1 e^*$, $X = X' \cap E'_0$, and $e^* \in E'_0 \cup \underline{F'_0}$:

- If $X \mapsto'_0 e$ then either $e = (X', e')$, $X = \{(X'', e'') \mid X'' \subseteq X'\}$, and $e'' \in X'$, or $e = (e_0, e_2)$ and there exists X' such that $X' \mapsto_{0\times 2} e$ and $X = \{e' \mid (\pi_0(e'), \pi_2(e')) \in X'\}$.
  - If $e = (X', e')$, $X = \{(X'', e'') \mid X'' \subseteq X'\}$, and $e'' \in X'$ then clearly $X \mapsto'_1 e$.
  - If $e = (e_0, e_2)$ and there exists X' such that $X' \mapsto_{0\times 2} e$ and $X = \{e' \mid (\pi_0(e'), \pi_2(e')) \in X'\}$ then $X' \mapsto_{1\times 2} e$, and obviously $\{e' \mid (\pi_0(e'), \pi_2(e')) \in X'\} = \{e' \mid (\pi_1(e'), \pi_2(e')) \in X'\} \cap E'_0$.
- If $X \mapsto'_1 e$ and $e \in E'_0$ then either $e = (X', e')$, $X = \{(X'', e'') \mid X'' \subseteq X'\}$, and $e'' \in X'$, or $e = (e_1, e_2)$ and there exists X' such that $X' \mapsto_{1\times 2} e$ and $X = \{e' \mid (\pi_1(e'), \pi_2(e')) \in X'\}$.
  - If $e = (X', e')$, $X = \{(X'', e'') \mid X'' \subseteq X'\}$, and $e'' \in X'$ then since $e \in E_0$, for all $e''' \in X'$, $\pi_1(e''') \in E_0$, meaning $X \subseteq E'_0$, and $\mapsto'_0 e$.
  - If $e = (e_1, e_2)$ and there exists X' such that $X' \mapsto_{1\times 2} e$ and $X = \{e' \mid (\pi_1(e'), \pi_2(e')) \in X'\}$ then $e_1 \in E_0 \cup \{*\}$, and $X' \cap (E_0 \times_* E_2) \mapsto_{0\times 2} e$, meaning $X \cap E'_0 = \{e' \mid (\pi_0(e'), \pi_2(e')) \in X' \cap (E_0 \times_* E_2)\} \mapsto e$.
- If $X \mapsto'_0 \underline{e}$ then $X = \bigcup \left\{ X'' \;\middle|\; \begin{array}{l} \exists i \in \{0,2\}, X_i \in E_i . X_i \mapsto \underline{\pi_i(e)} \\ \text{or } \exists e_\times \in X' . X_i \mapsto \underline{\pi_i(e_\times)} \\ \text{and } e' \in X'' \text{ iff } \pi_i(e') \in X_i \end{array} \right\}$ and $e = (X', (e_0, e_2))$, or $e = (e_0, e_1)$ and there exists X' such that $X' \mapsto_{\times_0} e$ and $X = \{e' \mid (\pi_0(e'), \pi_1(e')) \in X'\}$, or $X = \{e\}$.
  - If $X = \{e\}$ then obviously $X \mapsto'_1 \underline{e}$.
  - If $e = (X', (e_0, e_2))$ and $X = \left\{ X'' \;\middle|\; \begin{array}{l} \exists i \in \{0,2\}, X_i \in E_i . X_i \mapsto_i \underline{\pi_i(e)} \\ \text{or } \exists e_\times \in X' . X_i \mapsto_i \underline{\pi_i(e_\times)} \\ \text{and } e' \in X'' \text{ iff } \pi_i(e') \in X_i \end{array} \right\}$ then (1) for each X_0 such that $X_0 \mapsto_0 \underline{e_0}$, there exists X_1 such that $X_1 \mapsto_1 \underline{e_0}$ and $X_1 \cap E_0 = X_0$, and (2) for each X_0 such that there exists $(e'_0, e'_2) \in X'$, such that $X_0 \mapsto'_0 \underline{e_0}$, there exists X_1 such that $X_1 \mapsto_1 \underline{e'_0}$ and $X_1 \cap E_0 = X_0$, meaning
$$\bigcup \left\{ X'' \;\middle|\; \begin{array}{l} \exists i \in \{1,2\}, X_i \in E_i . X_i \mapsto_i \underline{\pi_i(e)} \\ \text{or } \exists e_\times \in X' . X_i \mapsto_i \underline{\pi_i(e_\times)} \\ \text{and } e' \in X'' \text{ iff } \pi_i(e') \in X_i \end{array} \right\} \cap E_0 = X$$
  - If $e = (e_0, e_1)$ and there exists X' such that $X' \mapsto_{0\times 2} e$ and $X = \{e' \mid (\pi_0(e'), \pi_1(e')) \in X'\}$, then by Lemma 4.10, there exists X'' such that $X' = X'' \cap (E_0 \times_* E_2)$ and $X'' \mapsto_{1\times 2} e$, meaning $\{e' \mid (\pi_1(e'), \pi_1(e')) \in X''\} \mapsto'_1 e$.

- If $X \mapsto'_1 \underline{e}$ and $e \in E'_0$ then $X = \{e\}$, or $e = (X', (e_1, e_2))$ and $X = \bigcup\{X'' \mid \exists i \in \{0,1\}.\pi_i(X'') \mapsto \underline{e_i}$ or $\exists e' \in X'.\pi_i(X'') \mapsto_i \pi_i(\underline{e'})\}$. If $X = \{e\}$ then obviously $X \mapsto'_0 \underline{e}$.

  If $e = (X', (e_1, e_2))$ and $X = \bigcup \left\{ X'' \;\middle|\; \begin{array}{l} \exists i \in \{1,2\}, X_i \in E_i.X_i \mapsto_i \underline{\pi_i(e)} \\ \text{or } \exists e_\times \in X'.X_i \mapsto_i \underline{\pi_i(e_\times)} \\ \text{and } e' \in X'' \text{ iff } \pi_i(e') \in X_i \end{array} \right\}$ then for each X_1 such that $X_1 \mapsto \underline{e_1}$, we know $X_1 \cap E_0 \mapsto \underline{e_1}$, and for each X_1 such that there exists $(e'_1, e'_2) \in X'$ such that $X_1 \mapsto'_1 \underline{e_1}$, since $e \in E'_0$, $e'_1 \in E_0$, meaning $X_1 \cap E_0 \mapsto \underline{e_1}$. Therefore $X \cap E'_0 \mapsto'_0 e_1$.

We then prove that $e \rhd'_0 e'^*$ if and only if $e \rhd'_1 e'^*$, $e \in E'_0$, and $e' \in E'_0 \cap \underline{F'_0}$.

If $e \rhd'_0 e'^*$ then there exists $i \in \{0,2\}$ such that either (1) $\pi_i(e) \rhd_i \pi_i(e')^*$, or (2) $\pi_i(e) = \pi_i(e') \neq \bot$, or (3) $e'^* = e'$, $e \neq e'$, and $e \in X \mapsto \underline{e'}$, or (4) and there exist γ, γ' such that $\lambda(e) = \mathsf{roll}\ \gamma$ and $\lambda(e') = \mathsf{roll}\ \gamma'$. In all these cases it is clear that the same conditions will apply in $\mathcal{E}'_1$.

Similar logic applies if $e \rhd'_1 e'^*$ and $e, e' \in E'_0$.

Obviously $\lambda'_0 = \lambda'_1 \restriction_{E'_0}$ and $\mathsf{Act} = \mathsf{ran}(\lambda'_0)$. □

F.2. Proof of Lemma 7.6

Proof. We prove this by structural induction on P:

- Suppose $P = 0$. Then there are no events and the lemma is trivially true.
- Suppose $P = \mathsf{roll}\ \gamma$. Then $e \in X \mapsto e'$ means $e = e_s$ and $e' = e_r$, and obviously $\lambda(e') = \mathsf{roll}\ \gamma$.
- Suppose $P = \mathsf{rolling}\ \gamma$. Then the argument is the same as the previous case.
- Suppose $P = \alpha_{\gamma'}.P'$. Then $[\![P']\!] = \langle \mathcal{E}', \mathsf{Init}, k \rangle$ and either $X \mapsto' e'$ or $X = \{e_\alpha\}$. If $X \mapsto' e'$ then $e' \neq e_\alpha$ and by induction if $e' \not\rhd' \underline{e}$ then $\lambda(e') \in \{\mathsf{roll}\ \gamma, \mathsf{start\ roll}\ \gamma\}$. If $X = \{e_\alpha\}$ then, $e' \rhd e_\alpha$ unless $\lambda(e') \in \{\mathsf{roll}\ \gamma, \mathsf{start\ roll}\ \gamma\}$.
- Suppose $P = \alpha_{\gamma'}.[m]P'$. Then the argument is the same as the previous case.
- Suppose $P = P_0 + P_1$. Then $e = (i, e_i)$, $e' = (i, e'_i)$, $e_i \mapsto_i e'_i$, and $e'_i \not\rhd_i \underline{e_i}$, meaning $\lambda_i(e'_i) \in \{\mathsf{roll}\ \gamma, \mathsf{start\ roll}\ \gamma\}$, and therefore $\lambda(e') \in \{\mathsf{roll}\ \gamma, \mathsf{start\ roll}\ \gamma\}$.
- Suppose $P = P_0 \mid P_1$. Then if $e' = (Y', e'')$, $e = (Y, e''')$ and $e''' \in Y'$, meaning there exists $i \in \{0,1\}$ such that $\pi_i(e) \in X_i \mapsto_i \pi_i(e')$. By induction we get that, if $e'_i \not\rhd_i \underline{e_i}$, then there exists γ such that $\lambda_i(e'_i) \in \{\mathsf{roll}\ \gamma, \mathsf{start\ roll}\ \gamma\}$, meaning $e'_{1-i} =* \underline{\ }$ and $\lambda(e') \in \{\mathsf{roll}\ \gamma, \mathsf{start\ roll}\ \gamma\}$, and if $e'_i \rhd_i \underline{e_i}$ then $e' \rhd \underline{e}$. If $e' \in E_\times$ then $\lambda(e') \in \{\mathsf{roll}\ \gamma, \mathsf{start\ roll}\ \gamma\}$
- Suppose $P = P' \setminus A$. Then the result follows from induction.
- Suppose $P = A\langle \tilde{a}, \tilde{\gamma} \rangle$ and $A(\tilde{b}, \tilde{\delta}) = P_A$. Then the result holds if it holds for P_A. □

F.3. Proof of Lemma 7.7

Proof. We prove this by structural induction on P:

- Suppose $P = 0$. Then there are no events and the lemma is trivially true.
- Suppose $P = \mathsf{roll}\ \gamma$. Then no X, X', e, e' exist such that $X \mapsto e \in X' \mapsto e'$.
- Suppose $P = \mathsf{rolling}\ \gamma$. Then the argument is the same as the previous case.
- Suppose $P = \alpha_{\gamma'}.P'$. Then $[\![P']\!] = \langle \mathcal{E}', \mathsf{Init}, k \rangle$, $X' \mapsto' e'$, and either $X = \{e_\alpha\}$, or $X \mapsto' e$. If $X = \{e_\alpha\}$, then $X \mapsto e'$ whenever $\lambda(e') \neq \mathsf{roll}\ \gamma'$. If $X \mapsto' e$ and $X' \mapsto' e'$ then by induction, $X \mapsto e'$.
- Suppose $P = \alpha_{\gamma'}.[m]P'$. Then the argument is the same as the previous case.
- Suppose $P = P_0 + P_1$. Then there exists an $i \in \{0, 1\}$ such that $e = (i, e_i)$, $e' = (i, e_i')$, $\{e_i'' \mid (i, e_i'') \in X'\} \mapsto_i e_i'$, and $\{e_i'' \mid (i, e_i'') \in X\} \mapsto_i e_i$, meaning by induction $\{e_i'' \mid (i, e_i'') \in X\} \mapsto_i e_i'$, and therefore $X \mapsto e'$.
- Suppose $P = P_0 \mid P_1$. Then $e' = (Y', (e_0', e_1'))$ or there exists a γ such that $\lambda(e') = \mathsf{roll}\ \gamma$. If $e' = (Y', (e_0', e_1'))$ then $e = (Y, (e_0, e_1))$ and $(e_0, e_1) \in Y'$, meaning there exists $i \in \{0, 1\}$ such that $e_i \in X_i \mapsto_i e_i'$. Similarly, $X = \{(Y'', e'') \mid (Y'', e'') \in E\}$ for some $e'' \in Y$. Since $Y \in \mathsf{cause}(e)$ and $e \in Y' \in \mathsf{cause}(e')$, there exists $Y'' \in \mathsf{cause}(e')$ such that $Y \subseteq Y''$. This means $X \mapsto e$.
- Suppose $P = P' \setminus A$. Then the result follows from induction.
- Suppose $P = A\,\langle \tilde{a}, \tilde{\gamma} \rangle$ and $A(\tilde{b}, \tilde{\delta}) = P_A$. Then the result holds if it holds for P_A. □

F.4. Proof of Lemma 7.8

Proof. We prove this by structural induction on P:

- Suppose $P = 0$. Then $E = \emptyset$ and the case is trivial.
- Suppose $P = \mathsf{roll}\ \gamma$. Then $e' = e_s$ and $e = e_r$.
- Suppose $P = \mathsf{rolling}\ \gamma$. Then $e' = e_s$ and $e = e_r$.
- Suppose $P = \alpha_\gamma.P'$. Then by induction, if $X \mapsto_{P'} e$, then there exists an e' such that $X = \{e'\}$. If $X \not\mapsto_{P'} e$, then $X = \{e_\alpha\}$.
- Suppose $P = \alpha_\gamma[m].P'$. Then by induction, if $X \mapsto_{P'} e$, there exists an e' such that $X = \{e'\}$. If $X \not\mapsto_{P'} e$, then $X = \{e_\alpha\}$.
- Suppose $P = P_0 + P_1$. Then, by induction if $e = (i, e_i)$ and $X_i \mapsto_i e_i$, then $X_i = e_i'$, and $e' = (i, e_i')$.

- Suppose $P = P_0 \mid P_1$. Then either $e = (Y, e_\times)$ or there exists a γ such that $\lambda(e) = \mathsf{roll}\ \gamma$. If $e = (Y, e_\times)$ then $X = \{(Y'', e'') \mid (Y'', e'') \in E\}$ for some $e'' \in Y$. We therefore need to show that given an event $e'' \in Y$, there exists exactly one $Y'' \in \mathsf{cause}(e'')$ such that $Y'' \subseteq Y$. This follows naturally from items 2 and 3 of Definition 7.1.

- Suppose $P = P' \setminus A$. Then the lemma obviously follows from the definition of ρ (Definition 7.3).

- Suppose $P = A\langle \tilde{a}, \tilde{\gamma} \rangle$ and $A(\tilde{b}, \tilde{\delta}) = P_A$. Then the lemma holds if it holds for P_A. □

F.5. Proof of Lemma 7.9

Proof. We prove this by structural induction on P:

- Suppose $P = 0$. Then $E = \emptyset$ and the case is trivial.

- Suppose $P = \mathsf{roll}\ \gamma$. Then $e' = e_r$ and $e = e_s$.

- Suppose $P = \mathsf{rolling}\ \gamma$. Then $e' = e_r$ and $e = e_s$.

- Suppose $P = \alpha_\gamma.P'$. Then either $e = e_\alpha$ and $X = \{e'' \mid \lambda'_{P'}(e'') = \mathsf{roll}\ \gamma\}$, or $\lambda_{P'}(e) = \mathsf{roll}\ \gamma'$ and $X \mapsto_{P'} \underline{e}$, or $\lambda'_{P'}(e'') \neq \mathsf{roll}\ \gamma'$, $\{e\} \neq X' \mapsto_{P'} e$, and $X = X' \cup \{e'' \mid \lambda'_{P'}(e'') = \mathsf{roll}\ \gamma\}$.

  In either case, it is clear that $e' \rhd \underline{e}$.

- Suppose $P = \alpha_\gamma[m].P'$. Then the argument is similar to the previous case.

- Suppose $P = P_0 + P_1$. Then, by induction if $e = (i, e_i)$ and $e'_i \in X_i \mapsto_i \underline{e_i}$, then $e'_i \rhd \underline{e_i}$, and $e' = (i, e'_i)$, meaning $e' \rhd \underline{e}$.

- Suppose $P = P_0 \mid P_1$. Then the lemma holds by definition.

- Suppose $P = P' \setminus A$. Then the result follows from induction.

- Suppose $P = A\langle \tilde{a}, \tilde{\gamma} \rangle$ and $A(\tilde{b}, \tilde{\delta}) = P_A$. Then the result holds if it holds for P_A. □

F.6. Proof of Lemma 7.10

Proof. We prove this by structural induction on P:

- Suppose $P = 0$. Then $E = \emptyset$ and the case is trivial.

- Suppose $P = \mathsf{roll}\ \gamma$. Then $X = \{e_r\}$ and $e = e_s$.

- Suppose $P = \mathsf{rolling}\ \gamma$. Then $X = \{e_r\}$ and $e = e_s$.

- Suppose $P = \alpha_\gamma.P'$. Then either $e = e_\alpha$ and $X = \{e'' \mid \lambda'_{P'}(e'') = \mathsf{roll}\ \gamma\}$, or $\lambda_{P'}(e) = \mathsf{roll}\ \gamma'$ and $X \mapsto_{P'} \underline{e}$, or $\lambda'_{P'}(e'') \neq \mathsf{roll}\ \gamma'$, $\{e\} \neq X' \mapsto_{P'} e$, and $X = X' \cup \{e'' \mid \lambda'_{P'}(e'') = \mathsf{roll}\ \gamma\}$.

  In either case, it is clear that there exists only one X.

- Suppose $P = \alpha_\gamma[m].P'$. Then the argument is similar to the previous case.

- Suppose $P = P_0 + P_1$. Then, by induction if $e = (i, e_i)$ then there exists at most one X_i such that $e'_i \in X_i \mapsto_i \underline{e_i}$, meaning $\{i\} \times X_i \mapsto e$.

- Suppose $P = P_0 \mid P_1$. Then either $e = (X', e')$, in which case the lemma obviously holds, or there exists X' such that $X' \mapsto_\times e$ and $X = \{e' \mid (\pi_0(e'), \pi_1(e')) \in X'\}$. By induction, since there exists an $i \in \{0, 1\}$ such that $\pi_i(e) = \bot$, there can only exist one such X'.

- Suppose $P = P' \setminus A$. Then the result follows from induction.

- Suppose $P = A\,\langle \tilde{a}, \tilde{\gamma} \rangle$ and $A(\tilde{b}, \tilde{\delta}) = P_A$. Then the result holds if it holds for P_A. □

F.7. Proof of Lemma 7.11

Proof. We prove this by structural induction on P:

- Suppose $P = 0$. Then $E = \emptyset$ and the case is trivial.

- Suppose $P = \mathsf{roll}\ \gamma$. Then there does not exist any e such that $\lambda(e) = \mu$.

- Suppose $P = \mathsf{rolling}\ \gamma$. Then there does not exist any e such that $\lambda(e) = \mu$.

- Suppose $P = \alpha_\gamma.P'$. Then either $X \mapsto_{P'} e$ or $X = \{e_\alpha\}$ and $e \in E_{P'}$.

  If $X \mapsto_{P'} e$ then there exists an $X_{P'}$ such that $X_{P'} \mapsto_{P'} \underline{e}$ and $X' = X_{P'} \cup \{e'' \mid \lambda_{P'}(e'') = \mathsf{roll}\ \gamma\}$. This means there exists $X'_{P'}$ such that $X'_{P'} \mapsto_{P'} \underline{e'}$, $X_{P'} \subseteq X'_{P'}$, and $X'_{P'} \cup \{e'' \mid \lambda_{P'}(e'') = \mathsf{roll}\ \gamma\} = X'' \mapsto \mathit{underlinee}$.

  If $X = \{e_\alpha\}$ and $e \in E_{P'}$ then $X' = \{e'' \mid \lambda_{P'}(e'') = \mathsf{roll}\ \gamma\}$ and there exists an $X_{P'}$ such that $X_{P'} \mapsto_{P'} \underline{e}$ and $X'' = X_{P'} \cup \{e'' \mid \lambda_{P'}(e'') = \mathsf{roll}\ \gamma\}$, meaning clearly $X' \subseteq X''$.

- Suppose $P = \alpha_\gamma[m].P'$. Then the argument is similar to the previous case.

- Suppose $P = P_0 + P_1$. Then, if $e = (i, e_i)$ and $e' = (i, e'_i)$ then there exists X'_i such that $X' = \{i\} \times X'_i$ and $X'_i \mapsto \underline{e'_i}$, meaning there exists $X''_i \supseteq X'_i$ such that $X''_i \mapsto \underline{e_i}$ and therefore $\{i\} \times X''_i = \overline{X''} \mapsto \underline{e}$.

- Suppose $P = P_0 \mid P_1$. Then either $e = (Y, e_\times)$, or there exists X' such that $X' \mapsto_\times e$ and $X = \{e' \mid (\pi_0(e'), \pi_1(e')) \in X'\}$.

  If $e = (Y, e_\times)$ then $e' = (Y', e'_\times)$ and $Y' \cup \{e'_\times\} \subseteq Y$, and

$$X' = \bigcup \left\{ X''' \;\middle|\; \begin{array}{l} \exists i \in \{0,1\}, X_i \in E_i.X_i \mapsto_i \underline{\pi_i(e')} \\ \text{or } \exists e''_\times \in Y'.X_i \mapsto_i \underline{\pi_i(e''_\times)} \\ \text{and } e'' \in X''' \text{ iff } \underline{\pi_i(e'')} \in X_i \end{array} \right\}.$$

We define $X'' = \bigcup \left\{ X''' \;\middle|\; \begin{array}{l} \exists i \in \{0,1\}, X_i \in E_i.X_i \mapsto_i \underline{\pi_i(e)} \\ \text{or } \exists e''_\times \in Y.X_i \mapsto_i \underline{\pi_i(e''_\times)} \\ \text{and } e'' \in X''' \text{ iff } \underline{\pi_i(e'')} \in X_i \end{array} \right\}$ and show that $X' \subseteq X''$. By definition, since $e_\times \in Y'$, whenever $X_i \mapsto \pi_i(e')$ we get $\pi(e'') \in X_i$ iff $e'' \in X''$. And if there exists $e''_\times \in Y'$ such that $X_i \mapsto \pi_i(e''_\times)$ then, since $Y' \subseteq Y$, $e''_\times \in Y$ and therefore $\pi(e'') \in X_i$ iff $e'' \in X''$.

- Suppose $P = P' \setminus A$. Then the lemma obviously follows from induction.
- Suppose $P = A\,\langle \tilde{a}, \tilde{\gamma} \rangle$. Then the lemma holds if it holds for P_A. □

F.8. Proof of Proposition 7.12

Proof. We say that $\mathcal{E} = (E, F, \mapsto, \rhd, \lambda, \mathsf{Act})$ and $\mathcal{E}' = (E', F', \mapsto', \rhd', \lambda', \mathsf{Act}')$ and do a case analysis on the structural congruence rules, describing how $\mathcal{E}$ and $\mathcal{E}'$ are constructed and defining isomorphisms for each rule:

$P = Q \mid R$ and $P' = R \mid Q$: Then there exist $\mathcal{E}_Q$ and $\mathcal{E}_R$ such that for $i \in \{Q, R\}$, $[\![P_i]\!] = \langle \mathcal{E}_i, \mathsf{Init}_i, k_i \rangle$ and $\langle \mathcal{E}, \mathsf{Init}, k \rangle$ is composed of them as defined in the event structure semantics.

And there exist $\mathcal{E}'_Q$ and $\mathcal{E}'_R$ such that for $i \in \{Q, R\}$, $[\![P_i]\!] = \left\langle \mathcal{E}'_i, \mathsf{Init}'_i, k'_i \right\rangle$ and $\left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$ is composed of them as defined in the event structure semantics.

And by induction we have isomorphisms $f_Q : \mathcal{E}_Q \to \mathcal{E}'_Q$ and $f_R : \mathcal{E}_R \to \mathcal{E}'_R$ fulfilling the conditions.

We first define a helper function

$$f'(e) = \begin{cases} (f_R(e_R), f_Q(e_Q)) & \text{if } e = (e_Q, e_R) \\ (f_R(e_R), *) & \text{if } e = (*, e_R) \\ (*, f_Q(e_Q)) & \text{if } e = (e_Q, *) \end{cases}$$

and then our isomorphism

$$f(e) = \begin{cases} (\{ f'(e'') \mid e'' \in X \}, f'(e')) & \text{if } e = (X, e') \\ f'(e) & \text{otherwise} \end{cases}$$

Since the definition of parallel composition treats both parts the same way, this clearly fulfils the conditions.

$P = P_0 \mid (P_1 \mid P_2)$ and $P' = (P_0 \mid P_1) \mid P_2$: Then there exist $\mathcal{E}_0$, $\mathcal{E}_1$, $\mathcal{E}_2$, and $\mathcal{E}_{12}$ such that $[\![P_0]\!] = \langle \mathcal{E}_0, \mathsf{Init}_0, k_0 \rangle$, $[\![P_1]\!] = \langle \mathcal{E}_1, \mathsf{Init}_1, k_1 \rangle$, $[\![P_2]\!] = \langle \mathcal{E}_2, \mathsf{Init}_2, k_2 \rangle$, $\langle \mathcal{E}_{12}, \mathsf{Init}_{12}, k_{12} \rangle$ is composed of $\langle \mathcal{E}_1, \mathsf{Init}_1, k_1 \rangle$ and $\langle \mathcal{E}_2, \mathsf{Init}_2, k_2 \rangle$ as described in the parallel rule, and $\langle \mathcal{E}, \mathsf{Init}, k \rangle$ is composed of $\langle \mathcal{E}_0, \mathsf{Init}_0, k_0 \rangle$ and $\langle \mathcal{E}_{12}, \mathsf{Init}_{12}, k_{12} \rangle$ as described in the parallel rule.

Additionally, there exist event structures $\mathcal{E}'_0$, $\mathcal{E}'_1$, $\mathcal{E}'_2$, and $\mathcal{E}_{01}$ generated as follows: $[\![P_0]\!] = \left\langle \mathcal{E}'_0, \mathsf{Init}'_0, k'_0 \right\rangle$, $[\![P_1]\!] = \left\langle \mathcal{E}'_1, \mathsf{Init}'_1, k'_1 \right\rangle$, $[\![P_2]\!] = \left\langle \mathcal{E}'_2, \mathsf{Init}'_2, k'_2 \right\rangle$, $\langle \mathcal{E}_{01}, \mathsf{Init}_{01}, k_{01} \rangle$ is composed of $\langle \mathcal{E}_0, \mathsf{Init}_0, k_0 \rangle$ and $\langle \mathcal{E}_1, \mathsf{Init}_1, k_1 \rangle$ as described in the parallel rule. Then $\left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$ is composed of $\langle \mathcal{E}_{01}, \mathsf{Init}_{01}, k_{01} \rangle$ and $\langle \mathcal{E}_2, \mathsf{Init}_2, k_2 \rangle$ as described in the parallel rule. And there exist isomorphisms $f_0 : \mathcal{E}_0 \to \mathcal{E}'_0$, $f_1 : \mathcal{E}_1 \to \mathcal{E}'_1$, and $f_2 : \mathcal{E}_2 \to \mathcal{E}'_2$ satisfying the conditions of the proposition.

We define a helper function $f_{01}(e_0, e_1) = (f_0(e_0), f_1(e_1))$ if $e_0 \in E_0$ and $e_1 \in E_1$ and define the morphism

$$f(e) = \begin{cases} (f_{01}(e_0, e_1), f_2(e_2)) & \text{if } e = (e_0, (e_1, e_2)) \\ (Y, ((Y', e_{01}), f_2(e_2)) & \text{if } e = (X, (e_0, (X', (e_1, e_2)))), \\ & e_{01} = f_{01}(e_0, e_1) \\ & Y' = \{ f_{01}(e'_0, e'_1) \mid \exists e'_2, X''. \\ & (e'_0, (X'', (e'_1, e'_2))) \in X \text{ and} \\ & e'_0 \in X_0 \in \mathsf{cause}(e_0) \text{ or } e'_1 \in X_1 \in \mathsf{cause}(e_1) \}, \\ & \text{and} \\ & Y = \{ ((f_{01}(Y''), f_{01}(e'_0, e'_1)), e'_2) \in E_{01\times 2} \mid \\ & \exists X''.(e'_0, (X'', (e'_1, e'_2))) \in X \text{ and} \\ & (f_{01}(Y''), f_{01}(e'_0, e'_1)) \in Y' \} \end{cases}$$

We first show that for any $e = (X, (e_0, (X', (e_1, e_2))))$, there exists at most one possible $f(e) \in E'$: Since causes must be conflict-free, there can at most exist one e'_2 and X'' for each e'_0 and e'_1 such that $(e'_0, (X'', (e'_1, e'_2))) \in X$, meaning there can only exist one Y' and Y fulfilling the conditions.

We then show that for any $e = (X, (e_0, (X', (e_1, e_2))))$, there exists $f(e') = (Y, ((Y', (e'_0, e'_1)), e'_2)) \in E'$: By induction, $e'_0 \in E'_0$, $e'_1 \in E'_1$, and $e'_2 \in E'_2$, so we show that $(Y', (e'_0, e'_1)) \in E_{01}$. We know there exists $X_1 \in \mathsf{cause}(e_1)$ such that $X_1 \subseteq \pi_1(X') = \pi_1(\pi_{12}(X))$, and there exists $X_0 \in \mathsf{cause}(e_0)$ such that $X_0 \subseteq \pi_0(X)$. And since for all $e \in Y'$, either $e'_0 \in X_0 \in \mathsf{cause}(e_0)$ or $e'_Y \in X_Y \in \mathsf{cause}(e_1)$, we get that $Y' \in \mathsf{cause}(f_0(e_0), f_1(e_1))$, and therefore we have an event $(Y', (f_0(e_0), f_1(e_1))) \in E_{01}$. And for similar reasons we also get $Y \in \mathsf{cause}((Y', (f_0(e_0), f_1(e_1))), f_2(e_2))$, meaning we have an event $(Y, ((Y', (f_0(e_0), f_1(e_1))), f_2(e_2))) \in E'$.

We then show that for any $e' = (X, ((X', (e'_0, e'_1)), e'_2)) \in E'$, there exists $e = (Y, (e_0, (Y', (e_1, e_2)))) \in E$ such that $f(e) = e'$. By induction, there obviously exist e_0, e_1, e_2 such that $f_0(e_0) = e'_0$, $f_1(e_1) = e'_1$, and $f_2(e_2) = e'_2$. We also know there exist $X_0 \in \mathsf{cause}(e'_0)$, $X_1 \in \mathsf{cause}(e'_1)$, and $X_2 \in \mathsf{cause}(e'_2)$ such that (1) whenever $((X'', (e''_0, e''_1)), e''_2) \in X$, either $e''_0 \in X_0$ or $e''_1 \in X_1$ or $e''_2 \in X_2$, and for each $(e'''_0, e'''_1) \in X''$, there exists $X''' \subseteq X''$ and e'''_2 such that $((X''', (e'''_0, e'''_1)), e'''_2) \in X$; and (2) whenever $e_i \in X_i$, there exists $((X'', (e''_0, e''_1)), e''_2) \in X$ such that $e_i \in \{e''_0, e''_1, e''_2\}$.

For $i \in \{0,1,2\}$, since f_i is an isomorphism, $f_i^{-1}(X_i) \in \mathsf{cause}(e_i)$, meaning if we set

$$Y' = \{(f_1^{-1}(e_1''), f_2^{-1}(e_2'')) \mid ((X'', (e_0'', e_1'')), e_2'') \in X \text{ and } e_1'' \in X_1 \text{ or } e_2'' \in X_2\}$$

and

$$Y = \left\{ (f_0^{-1}(e_0''), (Y'', (f_1^{-1}(e_1''), f_2^{-1}(e_2'')))) \;\middle|\; \begin{array}{l} \exists X''.((X'', (e_0'', e_1'')), e_2'') \in X \text{ and} \\ (Y'', (f_1^{-1}(e_1''), f_2^{-1}(e_2''))) \in Y' \end{array} \right\}$$

we have $e = (Y, (e_0, (Y', (e_1, e_2)))) \in E$ and $f(e) = e'$.

We then show that f is a morphism, meaning for $e, e' \in E$:

- Obviously $\lambda(e) = \lambda'(f(e))$.
- If $f(e) = f(e')$ then one of the following holds: (1) $e = (e_0, (e_1, e_2)) = e'$, or (2) $e = (X, (e_0, (Y, (e_1, e_2))))$ and $e' = (X', (e_0, (Y', (e_1, e_2))))$, and $(e_0'', (Y'', (e_1'', e_2''))) \in X$ if and only if there exists $(e_0'', (Y''', (e_1'', e_2''))) \in X'$. However, since $Y'', Y''' \in \mathsf{cause}(e_1'', e_2'')$, either $Y'' = Y'''$, or there exist $y'' \in Y''$ and $y''' \in Y'''$ such that $y'' \sharp_{12} y'''$. And in addition, there exist $e_0''', e_0'''', Y_{y''}$, and $Y_{y'''}$ such that $(e_0''', (Y_{y''}, y'')) \in X$ and $(e_0'''', (Y_{y'''}, y''')) \in X'$. Since X and x' must be conflict-free, $X = X'$.
- If $X \mapsto f(e)^*$, then either $e^* = (e_0, (e_1, e_2))$, $e^* = (Y, (e_0, (Y', (e_1, e_2))))$, $e^* = \underline{(e_0, (e_1, e_2))}$, or $e^* = \underline{(Y, (e_0, (Y', (e_1, e_2))))}$.
  If $e = (e_0, (e_1, e_2))$ then there exists $i \in 0, 12$ and X_i such that $X_i \mapsto_i \pi_i(e)$, and $X = \{e'' \mid \pi_i(e'') \in X_i\}$, meaning if $i = 0$, then $\{e'' \mid \pi_0(e'') \in X_i\} \mapsto (e_0, e_1)$ and therefore $\{e'' \mid \pi_0(\pi_{01}(e'')) \in X_i\} \mapsto ((e_0, e_1), e_2)$, and obviously $f(\{e'' \mid \pi_0(\pi_{01}(e'')) \in X_i\}) = X$. If $i = 12$ then there exists $j \in 1, 2$ and X_j such that $X_j \mapsto_j \pi_j(e)$, and $X = \{e'' \mid \pi_j(\pi_{12}(e'')) \in X_j\}$, and by similar logic if $j = 1$ then $\{e'' \mid \pi_1(\pi_{01}(e'')) \in X_j\} \mapsto ((e_0, e_1), e_2)$, and $f(\{e'' \mid \pi_1(\pi_{01}(e'')) \in X_j\}) = X$ and if $j = 2$ then $\{e'' \mid \pi_2(e'') \in X_j\} \mapsto ((e_0, e_1), e_2)$, and $f(\{e'' \mid \pi_2(e'') \in X_j\}) = X$.
  If $e = (Y, (e_0, (Y', (e_1, e_2))))$ and $f(e) = (Z, ((Z', (f_0(e_0), f_1(e_1))), f_2(e_2)))$ then there exists $e' = ((Z'', (e_0', e_1')), e_2') \in Z$ such that $X = \{(X', e') \mid X' \subseteq X\}$, and obviously

$P = P' \mid 0$**:** Then there exists $\mathcal{E}$ and $\mathcal{E}'$ such that $\{\!|P'|\!\} = \left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$, $\{\!|P|\!\} = \langle \mathcal{E}, \mathsf{Init}, k \rangle$, and $\mathcal{E}$ is composed of $\mathcal{E}'$ and the empty LREBES, $\mathcal{E}_0$ as described in the parallel composition rule.

We define $f(e) = \begin{cases} e' & \text{if } e = (X, (e', *)) \\ e' & \text{if } e = (e', *) \end{cases}$

And show that $f : \mathcal{E} \to \mathcal{E}'$ is a morphism, meaning for all $e_0, e_1 \in E$:

- Clearly $\lambda(e_0) = \lambda'(f(e_0))$.

- If $f(e_0) = f(e_1)$ then there exists e such that either $e_0 = (e, *) = e_1$ or there exist X_0 and X_1 such that $e_0 = (X_0(e, *))$ and $e_1 = (X_1(e, *))$. However, by Lemma 7.8, we know that whenever $X_0' \mapsto e$, X_0' contains exactly one event, e_0'. Since that event cannot synchronise with anything from E_0, e_0' must be in every possible cause of e, and similarly for the causes of e_0'', meaning e can only have one cause in $\mathcal{E}' || \mathcal{E}_0$, and therefore $e_0 = e_1$.
- For $X' \subseteq E'$, if $X' \mapsto' f(e_0)^*$ then $f(e_0) = e_0'$ and either $e_0 = (e_0', *)$ or $e_0 = (X_0, (e_0', *))$.
  If $e_0^* = (e_0', *)$, then $\{e \mid \exists e' \in X'.e = (X, (e', *)) \text{ or } e = (e', *)\} \mapsto e_0$. Clearly $\{e \mid \exists e' \in X'.e = (X, (e', *)) \text{ or } e = (e', *)\} = \{e \mid f(e) \in X'\}$
  If $e_0^* = (X_0, (e_0', *))$ then by Lemma 7.8 there exists e such that $X' = \{e\}$. Clearly this requires that $(e, *) \in X_0$, which means $\{(X_0', (e, *)) \mid X_0' \subseteq X_0\} \mapsto e_0$, and clearly $f(\{(X_0', (e, *)) \mid X_0' \subseteq X_0\}) = \{e\}$.
  If $e_0^* = \underline{(e_0', *)}$ then $\{e \mid e = (X, (e', *)) \text{ or } e = (', *) \text{ for } e' \in X'\} \mapsto e_0^*$.
  If $e_0^* = \overline{(X_0, (e_0', *))}$ then $\bigcup\{X'' \mid \exists X''' \in E'.X''' \mapsto' \underline{e_0'} \text{ or } \exists (e', *) \in X_0.X'' \mapsto' \underline{e'}, \text{ and } e'' \in X'' \text{ iff } f(e'') \in X'''\} \mapsto e_0^*$, by Lemmas 7.10 and 7.11, we know that for all $e \in X_0$, if $X'' \mapsto \underline{e}$, then $X'' \subseteq X'$, meaning $X' = \bigcup\{X'' \mid \exists X''' \in E'.X''' \mapsto' \underline{e_0'} \text{ or } \exists (e', *) \in X_0.X'' \mapsto' \underline{e'}, \text{ and } e'' \in X'' \text{ iff } f(e'') \in X'''\}$.
- If $f(e_0) \rhd f(e_1)^*$ then by definition, $e_0 \rhd e_1 *$.

We then prove f is bijective: We already showed above, that f is injective, and it is clear that it is also surjective.

In order to show f is an isomorphism, we therefore only need to show that f^{-1} is a morphism, meaning for $e_0', e_1' \in E'$:

- Again, clearly $\lambda(f^{-1}(e_0')) = \lambda'(e_0')$.
- If $f^{-1}(e_0') = f^{-1}(e_1')$ then we already know f is a bijection, so $e_0' = e_1'$.
- For $X \subseteq E$, if $X \mapsto f^{-1}(e_0')^*$ then $f^{-1}(e_0') = e_0$ and either $e_0 = (e_0', *)$ or $e_0 = (X_0, (e_0', *))$.
  If $e_0^* = (e_0', *)$, then $\{e \mid (e, *) \in X \text{ or } \exists X'.(X', (e, *)) \in X\} \mapsto e_0'$.
  If $e_0^* = (X_0, (e_0', *))$ then by Lemma 7.8 we know there exists an e such that $X = \{e\}$. This means there exists X' such that $e = (X', (e', *))$ and $(e', *) \in X_0$, meaning $\{e'\} \mapsto' e_0'$.
  If $e_0^* = \overline{(e_0', *)}$ then either $X = \{e_0\}$, and obviously $\{e_0'\} \mapsto \underline{e_0'}$, or there exists an $\overline{X'}$ such that $X' \mapsto e_0$ and $X = \{e \mid \exists e' \in X'.e = (e', *) \text{ or } e = (X'', (e', *))\}$.
  If $e_0^* = \overline{(X_0, (e_0', *))}$ then either $X = \{e_0\}$, and obviously $\{e_0'\} \mapsto \underline{e_0'}$, or $X = \bigcup\{\overline{X'' \mid f(X'')} \mapsto' \underline{e_0'} \text{ or } \exists (e', *) \in X_0.f(X'') \mapsto' \underline{e'}\}$. Clearly any of these X''s can be used to fulfil the condition.

- If $f^{-1}(e'_0) \rhd f^{-1}(e'_1)^*$ then either (1) $e'_0 \rhd e'^*_1$, (2) $e'_0 = e'_1$ and $f^{-1}(e'_0) \neq f^{-1}(e'_1)^*$, (3) $e'^*_1 = e'_1$, $f^{-1}(e'_0) \neq f^{-1}(e'_1)^*$, and $f^{-1}(e'_0) \in X \mapsto \overline{f^{-1}(e'_1)}$, or (4) $e'^*_1 = e'_1$ and there exist γ_0 and γ_1 such that $\lambda(f^{-1}(e'_0)) = \mathsf{roll}\ \gamma_0$ and $\lambda(f^{-1}(e_1)) = \mathsf{roll}\ \gamma_1$.
  In case 1, the condition is trivially fulfilled. Case 2 will never occur. In case 3, as shown above, $e'_0 \in f(X) \mapsto \overline{e'_1}$, and by Lemma 7.9, this means $e'_0 \rhd e'_1$.
  In case 4, since the e'_0 and e'_1 must both have been caused by a rollback at the end of a subprocess, they were either in parallel or different option in a choice, and in either case clearly $e'_0 \rhd' e'_1$.

And obviously from Lemma 7.7 and the definition of Init, we see that $f(\mathsf{Init}) = \mathsf{Init}'$ and $f \circ k' = k$.

$P = X + Y$ **and** $P' = Y + X$**:** Selection works the same in Roll-CCSK as in CCSK, so this case is the same as in Proposition 4.15.

$P = (X + Y) + Z$ **and** $P' = (X + Y) + Z$**:** Selection works the same in Roll-CCSK as in CCSK, so this case is the same as in Proposition 4.15.

$P = P' + 0$**:** Selection works the same in roll-CCSK as in CCSK, so this case is the same as in Proposition 4.15.

$P = Q \setminus A$**,** $P' = Q' \setminus A$**, and** $Q \equiv Q'$**:** Then we have $\{\!|Q|\!\} = \left\langle \mathcal{E}_Q, \mathsf{Init}_Q, k_Q \right\rangle$, and $\{\!|Q'|\!\} = \left\langle \mathcal{E}_{Q'}, \mathsf{Init}_{Q'}, k_{Q'} \right\rangle$, there exist an isomorphism $f_Q : \mathcal{E}_Q \to \mathcal{E}_{Q'}$ such that $f_Q(\mathsf{Init}_Q) = \mathsf{Init}_{Q'}$ and for all $e \in \mathsf{Init}_Q$, $k_Q(e) = k_{Q'}(f_Q(e))$, and by applying the restriction we get

$$\langle \mathcal{E}, \mathsf{Init}, k \rangle = \left\langle \mathcal{E}_Q \upharpoonright \rho(A \cup \overline{A}), \mathsf{Init}_Q \cap \rho(A \cup \overline{A}), k_Q \upharpoonright \rho(A \cup \overline{A}) \right\rangle$$

and

$$\left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle = \left\langle \mathcal{E}_{Q'} \upharpoonright \rho(A \cup \overline{A}), \mathsf{Init}_{Q'} \cap \rho(A \cup \overline{A}), k_{Q'} \upharpoonright \rho(A \cup \overline{A}) \right\rangle$$

We now show that $e \in \rho(A \cup \overline{A})$ if and only if $f(e) \in \rho(A \cup \overline{A})$.

For any $e \in E_Q$, obviously $\lambda_Q(e) \in A \cup \overline{A}$ iff $\lambda_{Q'}(f(e)) \in A \cup \overline{A}$. We show that for any $X \subseteq E_Q$, $X \in \mathsf{causes}(e)$ if and only if $f(X) \in \mathsf{cause}(f(e))$ by induction in the size of X.

If $X = \emptyset$ then there does not exist $x \subseteq E_Q$ such that $x \mapsto_Q e$, and by definition of an morphism, there cannot exist $x' \subseteq E_{Q'}$ such that $x' \mapsto f(e)$, meaning $\emptyset \in \mathsf{cause}(e)$. And since f is an isomorphism the same argument can be used for f^{-1}.

If X contains n events, and for all events e' and $X' \in \mathsf{cause}(e')$ such that X' contains less that n events, $X' \subseteq \rho(A \cup \overline{A})$ if and only if $f(X') \in \mathsf{cause}(f(e'))$ then whenever $x' \mapsto_{Q'} f(e)$, there exists $x \subseteq E_Q$ such that $x \mapsto_Q e$ and $f(x) \subseteq x'$, meaning there exists e'' such that $x \cap X = \{e''\}$, and $x' \cap f(X) \supseteq \{f(e'')\}$. And

by induction if $X'' \mapsto e'' \in X$ then $X'' \subset X$ and therefore $f(X'') \in \mathsf{causes}(e'')$. And since X is conflict-free, obviously $f(X)$ is conflict-free. And since f is an isomorphism the same argument can be used for f^{-1}.

$P = A\langle \tilde{a}, \tilde{\gamma} \rangle$ **and** $P' = (\nu\ \tilde{\gamma}) P_A \{^{\tilde{a},\tilde{\gamma}}/_{\tilde{b},\tilde{\delta}}\}$ **where** $A\left\langle \tilde{b}, \tilde{\delta} \right\rangle = P_A$**:** Follows from Proposition 7.4. □

F.9. Proof of Theorem 7.13

Proof. We say that the inverse of f is $g : \mathcal{E}' \to \mathcal{E}$ and prove the result by induction on the transition $P \xrightarrow{\mu[m]} P'$ by constructing $\mathcal{E}$, $\mathcal{E}'$, f and g for each case:

- Suppose $P = \alpha_\gamma.Q$, $P' = \alpha_\gamma[m].Q$, $\mu = \alpha$, and $\mathsf{std}(Q)$. Then there exist $\mathcal{E}_Q$ and e_α such that $\{\!|Q|\!\} = \left\langle \mathcal{E}_Q, \mathsf{Init}, k \right\rangle$ and $\langle \mathcal{E}, \mathsf{Init}, k \rangle$ is constructed based on this as described in the prefix rule.

  And there exist $\mathcal{E}_{Q'}$ and e'_α such that $\{\!|Q|\!\} = \left\langle \mathcal{E}_{Q'}, \mathsf{Init}_{Q'}, k_{Q'} \right\rangle$ and $\left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$ is constructed from this a described in the past prefix rule.

  By induction, there must exist isomorphisms $f_Q : \mathcal{E}_Q \to \mathcal{E}_{Q'}$ and $g_Q : \mathcal{E}_{Q'} \to \mathcal{E}_Q$, and we define $f = f_Q \cup \{(e_\alpha, e'_\alpha)\}$ and $g = g_Q \cup \{(e'_\alpha, e_\alpha)\}$, which are clearly isomorphisms.

  Since $\mathsf{std}(Q)$, meaning $\mathsf{Init} = \emptyset$, and since no X exists such that $X \mapsto e_\alpha$, $\mathsf{Init} \xrightarrow{e_\alpha} \{e_\alpha\}$, and the rest of the conditions are obviously satisfied.

- Suppose that $P = \alpha_\gamma[n].Q$, $P' = \alpha_\gamma[n].Q'$, $Q \xrightarrow{\mu[m]} Q'$, and $n \neq m$.

  Then there exist $\mathcal{E}_Q$ and e_α such that $\{\!|Q|\!\} = \left\langle \mathcal{E}_Q, \mathsf{Init}_Q, k_Q \right\rangle$ and $\langle \mathcal{E}, \mathsf{Init}, k \rangle$ is constructed based on this as described in the past prefix rule.

  And there exist $\mathcal{E}_{Q'}$ and e'_α such that $\{\!|'Q|\!\} = \left\langle \mathcal{E}_{Q'}, \mathsf{Init}_{Q'}, k_{Q'} \right\rangle$ and $\left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$ is constructed from this a described in the past prefix rule.

  By induction, we get isomorphisms $f_Q : \mathcal{E}_Q \to \mathcal{E}_{Q'}$ and $g_Q : \mathcal{E}_{Q'} \to \mathcal{E}_Q$ and a transition $\mathsf{Init}_Q \xrightarrow{\{e\}} X_Q$ in $C_{re}(\mathcal{E}_Q)$ such that $\lambda_Q(e) = \mu$, $k_{Q'}(f_Q(e)) = m$, and $f_Q(X_Q) = \mathsf{Init}_{Q'}$.

  We define $f = f_Q \cup \{(e_\alpha, e'_\alpha)\}$ and $g = g_Q \cup \{(e'_\alpha, e_\alpha)\}$. Since Init_Q and X_Q are conflict-free in $\mathcal{E}_Q$, $\mathsf{Init}_Q \cup \{e_\alpha\} = \mathsf{Init}$ and $X_Q \cup \{e_\alpha\} = X$ are configurations of $C_{re}(\mathcal{E})$, and clearly $\mathsf{Init} \xrightarrow{\{e\}} X$.

- Suppose $P = P_0 \mid P_1$, $P' = P'_0 \mid P_1$, $P_0 \xrightarrow{\mu[m]} P'_0$, and $\mathsf{fsh}[m](P_1)$. Then there exist $\mathcal{E}_0$ and $\mathcal{E}_1$ such that for $i \in \{0, 1\}$, $\{\!|P_i|\!\} = \langle \mathcal{E}_i, \mathsf{Init}_i, k_i \rangle$, and $\langle \mathcal{E}, \mathsf{Init}, k \rangle$ is constructed as described in the parallel composition rule.

  And there exist $\mathcal{E}'_0$ and $\mathcal{E}'_1$ such that $\{\!|P'_0|\!\} = \left\langle \mathcal{E}'_0, \mathsf{Init}'_0, k'_0 \right\rangle$, $\{\!|P_1|\!\} = \left\langle \mathcal{E}'_1, \mathsf{Init}'_1, k'_1 \right\rangle$, and $\left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$ is constructed as described in the parallel composition rule.

We have isomorphisms $f_0 : \mathcal{E}_0 \to \mathcal{E}'_0$, $g_0 : \mathcal{E}'_0 \to \mathcal{E}_0$, $f_1 : \mathcal{E}_1 \to \mathcal{E}'_1$, and $g_1 : \mathcal{E}'_1 \to \mathcal{E}_1$, and there exists a transition $\mathsf{Init}_0 \xrightarrow{e_\mu} X$ in $C_{re}(\mathcal{E}_0)$ such that $\lambda_0(e_\mu) = \mu$, $k'_0(f_0(e)) = m$ and $f_0(X) = \mathsf{Init}'_0$.

We define functions

$$f'(e) = \begin{cases} (f_0(e_0), *) & \text{if } e = (e_0, *) \\ (*, f_1(e_1)) & \text{if } e = (*, e_1) \\ (f_0(e_0), f_1(e_1)) & \text{if } e = (e_0, e_1) \end{cases}$$

and

$$g'(e) = \begin{cases} (g_0(e_0), *) & \text{if } e = (e_0, *) \\ (*, g_1(e_1)) & \text{if } e = (*, e_1) \\ (g_0(e_0), g_1(e_1)) & \text{if } e = (e_0, e_1) \end{cases}$$

and our isomorphisms as:

$$f(e) = \begin{cases} (f'(X), f'(e')) & \text{if } e = (X, e') \\ f'(e) & \text{otherwise} \end{cases}$$

and

$$g(e) = \begin{cases} (g'(X), g'(e')) & \text{if } e = (X, e') \\ g'(e) & \text{otherwise} \end{cases}$$

It is clear that $f \circ g = I_{\mathcal{E}}$ and $g \circ f = I_{\mathcal{E}'}$.

We show that $f : \mathcal{E} \to \mathcal{E}'$ is a morphism, meaning for all $e, e' \in E$:

- Obviously $\lambda(e) = \lambda'(f(e))$
- If $f(e) = f(e')$ then since f_0 and f_1 are injective, $e = e'$.
- For $X' \subseteq E'$, if $X' \mapsto' f(e)^*$ then either $e^* = (Y, e_\times$ and there exists $e_{\times'} \in f(Y)$ such that $X' = \{(Y', e_{\times'}) \mid Y' \subseteq f'(Y)\}$, or $e^* \in E_\times$ and there exists X'' such that $X'' \mapsto'_\times f(e)$ and $X' = \{e' \mid (\pi'_0(e'), \pi'_1(e')) \in X''\}$, or $e^* = \underline{(Y, e_\times)}$ and $X' = \{f(e)\}$, or $e^* = \underline{(Y, e_\times)}$ and

$$X' = \bigcup \left\{ X'' \;\middle|\; \begin{array}{l} \exists i \in \{0,1\}, X_i \in E'_i \,.\, X'_i \mapsto'_i \overline{\pi'_i(f(e))} \\ \text{or } \exists e_{\times'} \in f(Y) \,.\, X'_i \mapsto'_i \overline{\pi'_i(e_{\times'})} \\ \text{and } e'' \in X'' \text{ iff } \pi'_i(e'') \in X'_i \end{array} \right\}$$

  or $e^* \in E_\times$ and there exists X'' such that $X'' \mapsto'_\times \underline{f(e)}$ and $X' = \{e' \mid (\pi'_0(e'), \overline{\pi'_1(e')}) \in X''\}$.

  If $e^* = (Y, e_\times$ and there exists $e_{\times'} \in f(Y)$ such that $X' = \{(Y', e_{\times'}) \mid Y' \subseteq f(Y)\}$ then there exists an $e'_\times \in Y$ such that $f(e'_\times) = e_{\times'}$ and clearly $\{(Y', e'_\times) \mid Y' \subseteq Y\} \mapsto (Y, e_\times)$.

  If $e^* \in E_\times$ and there exists X'' such that $X'' \mapsto'_\times f(e)$ and $X' = \{e' \mid (\pi'_0(e'), \pi'_1(e')) \in X''\}$ then by induction and since $||$ is an REBES product,

there exists $X''' \subseteq E_\times$ such that $X''' \mapsto_\times e$, $f(X''') \subseteq X''$, and if $e' \in X'''$ then $f(e') \neq \bot$. This means $\{e' \mid (\pi_0(e'), \pi_1(e')) \in X'''\} \mapsto e$.
If $e^* = \underline{(Y, e_\times)}$ and $X' = \{f(e)\}$, or $e^* = \underline{(Y, e_\times)}$ and

$$X' = \bigcup \left\{ X'' \;\middle|\; \begin{array}{l} \exists i \in \{0,1\}, X'_i \in E'_i . X'_i \mapsto'_i \underline{\pi'_i(f(e))} \\ \text{or } \exists e_{\times'} \in f(Y) . X'_i \mapsto'_i \underline{\pi'_i(e_{\times'})} \\ \text{and } e'' \in X'' \text{ iff } \pi'_i(e'') \in X'_i \end{array} \right\}$$

then by induction, since $g = f^{-1}$ is a morphism for each $X_i \mapsto_i \underline{\pi_i(e)}$, $f(X_i) \subset X'_i \mapsto'_i \underline{\pi'_i(f(e))}$, and for each $X_i \mapsto_i \underline{\pi_i(e'_\times)} \in \underline{Y}$, $f(X_i) \subset X'_i \mapsto'_i \underline{\pi'_i(f(e'_\times))}$ meaning

$$\left\{ X'' \;\middle|\; \begin{array}{l} \exists i \in \{0,1\}, X_i \in E_i . X_i \mapsto_i \underline{\pi_i(e)} \\ \text{or } \exists e'_\times \in Y . X_i \mapsto_i \underline{\pi_i(e'_\times)} \\ \text{and } e'' \in X'' \text{ iff } \pi'_i(e'') \in X_i \end{array} \right\}$$
$$\subseteq \left\{ X'' \;\middle|\; \begin{array}{l} \exists i \in \{0,1\}, X_i \in E'_i . X'_i \mapsto'_i \underline{\pi'_i(f(e))} \\ \text{or } \exists e_{\times'} \in f(Y) . X'_i \mapsto'_i \underline{\pi'_i(e_{\times'})} \\ \text{and } e'' \in X'' \text{ iff } \pi'_i(e'') \in X'_i \end{array} \right\}$$

If $e^* \in \underline{E_\times}$ and there exists X'' such that $X'' \mapsto'_\times \underline{f(e)}$ and $X' = \{e' \mid (\pi'_0(e'), \pi'_1(e')) \in X''\}$ then by induction and because $||$ is an REBES product, there exists $X''' \subseteq E_\times$ such that $f(X''') \subseteq X''$ and $X''' \mapsto_\times \underline{e}$. This means $\{e' \mid (\pi_0(e'), \pi_1(e')) \in X'''\} \mapsto \underline{e}$.

– If $f(e) \rhd' f(e')^*$ then there exists $i \in \{0,1\}$ such that either $\pi'_i(f(e)) \rhd'_i \pi'_i(f(e'))^*$, or $\pi'_i(f(e)) = \pi'_i(f(e')) \neq \bot$, and $f(e) \neq f(e')$, or $f(e')^* = f(e')$, $f(e) \neq f(e')$, and $f(e) \in X \mapsto' \underline{f(e')}$, or $f(e')^* = f(e')$ and there exist γ, γ' such that $\lambda'(f(e)) = \mathsf{roll}\ \gamma$ and $\lambda'(f(e')) = \mathsf{roll}\ \gamma'$.
If $\pi'_i(f(e)) \rhd'_i \pi'_i(f(e'))^*$ then by induction $\pi_i(e) \rhd_i \pi_i(e')^*$, meaning $e^* \rhd e$.
If $\pi'_i(f(e)) = \pi'_i(f(e')) \neq \bot$, and $f(e) \neq f(e')$ then $\pi_i(e) = \pi_i(e') \neq \bot$ and $e \neq e'$, meaning $e \rhd e'^*$.
If $f(e')^* = f(e')$, $f(e) \neq f(e')$, and $f(e) \in X \mapsto' \underline{f(e')}$ then, since, by similar arguments to the previous case, $g(X) \mapsto \underline{f(e')}$, and $e \in g(X)$, $e \rhd e^*$.
If $f(e')^* = f(e')$ and there exist γ, γ' such that $\lambda'(f(e)) = \mathsf{roll}\ \gamma$ and $\lambda'(f(e')) = \mathsf{roll}\ \gamma'$, then $\lambda(e) = \mathsf{roll}\ \gamma$ and $\lambda(e') = \mathsf{roll}\ \gamma'$, meaning $e \rhd e'^*$.

By similar arguments, g is a morphism too.

We now show that there exists an $(Y, (e_\mu, *)) \in E$ such that $\{e' \mid (\pi_0(e'), \pi_1(e')) \in Y\} \subseteq \mathsf{Init}$. Since $\mathsf{Init}_0 \xrightarrow{\{e_\mu\}}$, for every $X_0 \mapsto_0 e_\mu$, $X_0 \cap \mathsf{Init}_0 = X_0 = \{e_0\}$, and if $X'_0 \mapsto_0 e_0$ then by Lemma 7.7, $X'_0 \mapsto_0 e_\mu$, and therefore $X'_0 \cap \mathsf{Init}_0 \neq \emptyset$. Therefore there must exist one $(Y, (e_\mu, *)) \in E$ such that $\{e' \mid (\pi_0(e'), \pi_1(e')) \in Y\} \subseteq \mathsf{Init}$.
We use this $(Y, (e_\mu, *))$ as our e and show that $\mathsf{Init} \xrightarrow{\{e\}}$: Since $Y \subseteq \mathsf{Init}$, for every

$X \mapsto e$, $X \cap \mathsf{Init} \neq \emptyset$. And if $e' \rhd e$ then it must that either $\pi_0(e') \rhd_0 e_\mu$, in which case $\pi_0(e') \notin \mathsf{Init}_0$, and therefore $e' \notin \mathsf{Init}$, or $\pi_0(e') = e_\mu$ and $e \neq e'$, in which case, since $\mathsf{Init}_0 \xrightarrow{\{e_\mu\}}$, $e' \notin \mathsf{Init}_0$, and therefore $e' \notin \mathsf{Init}$, or $e' \in X \mapsto \underline{e}$ and $e' \neq e$, in which case $\pi_0(e') \in X_0 \mapsto e_\mu$ or $\pi_0(e') \in X_0 \mapsto \pi_0(e'')$ for $e'' \in Y$, and by Lemmas 7.9 and 7.11, $\pi_0(e') \rhd e_\mu$, meaning $\pi_0(e') \notin \mathsf{Init}_0$, and $e' \notin I$.

We therefore have $\mathsf{Init} \xrightarrow{\{e\}} I \cup \{e\}$, and obviously $\lambda(e) = \lambda_0(e_\mu) = \mu$ and $f \circ k' = k \cup \{(e, m)\}$, and since $f_0(\mathsf{Init}_0 \cup \{e_\mu\}) = \mathsf{Init}'_0$ and $f_1(\mathsf{Init}_1) = \mathsf{Init}'_1$, and there only exists one $(Y, (e_\mu, *)) \in E$ such that $\{e' \mid (\pi_0(e'), \pi_1(e')) \in Y\} \subseteq \mathsf{Init}$, $f(\mathsf{Init} \cup \{e\}) = \mathsf{Init}'$.

- Suppose $P = P_0 \mid P_1$, $P' = P'_0 \mid P'_1$, $P_0 \xrightarrow{\alpha[m]} P'_0$, $P_1 \xrightarrow{\overline{\alpha}[m]} P'_1$, and $\mu = \tau$.

  Then the construction of $\langle \mathcal{E}, \mathsf{Init}, k \rangle$ and $\left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$ and the isomorphisms are similar to the previous case. And by induction we have transitions $\mathsf{Init}_0 \xrightarrow{\{e_0\}}$ and $\mathsf{Init}_1 \xrightarrow{\{e_1\}}$ fulfilling the conditions.

  For similar reasons to the previous case there exists exactly one $(Y, (e_0, e_1)$ such that $\{e' \mid (\pi_0(e'), \pi_1(e')) \in Y\} \subseteq \mathsf{Init}$, and we use this $(Y, (e_0, e_1))$ as e, and the rest of the proof follows similarly.

- Suppose $P = P_0 + P_1$, $P' = P'_0 + P_1$, $P_0 \xrightarrow{\mu[m]} P'_0$, and $\mathsf{std}(P_1)$. Then the rule for selection is the same in roll-CCSK as in CCSK, and the case is therefore identical to Theorem 4.18.

- Suppose $P = Q \setminus A$, $P' = Q' \setminus A$, $Q \xrightarrow{\mu[m]} Q'$, and $\mu \notin A \cup \overline{A}$. Then there exist $\mathcal{E}_Q$ and $\mathcal{E}'_Q$ such that $[\![Q]\!] = \langle \mathcal{E}_Q, \mathsf{Init}_Q, k \rangle$, $[\![Q']\!] = \left\langle \mathcal{E}'_Q, \mathsf{Init}'_Q, k'_Q \right\rangle$, and $\mathcal{E}$ and $\mathcal{E}'$ are constructed from $\mathcal{E}_Q$ and $\mathcal{E}'_Q$ as described in the restriction rule, and there exist isomorphisms $f_Q : \mathcal{E}_Q \to \mathcal{E}'_Q$ and $g : \mathcal{E}'_Q \to \mathcal{E}_Q$ and a transition $\mathsf{Init}_Q \xrightarrow{\{e_Q\}}$ where $\lambda_Q(e_Q) = \mu$, $f_Q \circ k'_Q = k_Q \cup \{(e_Q, m)\}$, and $f_Q(\mathsf{Init}_Q \cup \{e_Q\}) = \mathsf{Init}'_Q$.

  Since there exists a standard process P'' such that $P'' \to^* P$, there cannot exist $e' \in \mathsf{Init}$ such that $\lambda(e') \in A \cup \overline{A}$ or for all $x \in \mathsf{cause}(e')$, there exists $e'' \in x$ such that $\lambda(e'') \in A \cup \overline{A}$, meaning $\mathsf{Init} \cap \rho(A \cup \overline{A}) = \mathsf{Init}$, and, since $e_Q \in \rho(A \cup \overline{A})$, $\mathsf{Init} \xrightarrow{e_Q}$.

- Suppose $P = Q[f]$, $P' = Q'[f]$, $Q \xrightarrow{\nu[m]} Q'$, and $f'(\nu) = \mu$. Then the rule for functions is the same in roll-CCSK as in CCSK, and the case is therefore identical to Theorem 4.18.

- Suppose $P \equiv Q$, $P' \equiv Q'$, and $Q \xrightarrow{\mu[m]} Q'$. Then the result follows from induction and Proposition 7.12. □

F.10. Proof of Theorem 7.14

Proof. We say that the inverse of f is $g : \mathcal{E}' \to \mathcal{E}$ and prove this result by induction on P by constructing $\mathcal{E}$, $\mathcal{E}'$, f and g for each case:

- Suppose $P = 0$. Then $E = \emptyset$, and obviously no transitions exist in $C_{br}(\mathcal{E})$.
- Suppose $P = \mathsf{roll}\ \gamma$. Then there does not exist $e \in E$ such that $\lambda(e) = \mu$.
- Suppose $P = \mathsf{rolling}\ \gamma$. Then there does not exist $e \in E$ such that $\lambda(e) = \mu$.
- Suppose $P = \alpha_\gamma.P''$. Then $\{e_\alpha\} \mapsto e'$ for all $e' \in E \setminus \{e_\alpha\}$ such that $\lambda(e) = \mu$, meaning by definition $e = e_\alpha$. In addition, by Lemma 7.6, whenever $e' \in \mathsf{Init}$, $\lambda(e') \in \{\mathsf{roll}\ \gamma', \mathsf{start\ roll}\ \gamma'\}$ meaning $\mathsf{std}(P)$. This means we get $P \xrightarrow{\alpha[m]} \alpha[m].P''$ for some fresh m, and the isomorphisms are similar to this case in the proof of Theorem 4.18.
- Suppose $P = \alpha[n].P''$ and $[\![P'']\!] = \left\langle \mathcal{E}'', \mathsf{Init}'', k'' \right\rangle$. Then $e_\alpha \in \mathsf{Init}$, and clearly $\mathsf{Init}'' \xrightarrow{e} X''$, meaning there exists a key m and a transition $P'' \xrightarrow{\lambda(e)[m]} P'''$, such that $[\![P''']\!] = \left\langle \mathcal{E}''', \mathsf{Init}''', k''' \right\rangle$ and there exist isomorphisms $f'' : \mathcal{E}'' \to \mathcal{E}'''$ and $g'' : \mathcal{E}''' \to \mathcal{E}''$ such that $k'''(f''(e)) = m$ and $f''(X'') = \mathsf{Init}'''$. If $m \neq n$, then $P \xrightarrow{\lambda(e)[m]} \alpha[m].P'''$. Otherwise, we can chose a fresh m and still get a transition. We define our isomorphisms as $f = f'' \cup \{(e_\alpha, e'_\alpha)\}$ and $g = g'' \cup \{(e'_\alpha, e_\alpha)\}$ and the rest of the proof is straightforward.
- Suppose $P = P_0 + P_1$. Then the proof is similar to the same case in CCSK, as the choice semantics is the same.
- Suppose $P = P_0 \mid P_1$, $[\![P_0]\!] = \langle \mathcal{E}_0, \mathsf{Init}_0, k_0 \rangle$, $C_{br}(\mathcal{E}_0) = (E_0, F_0, \mathsf{C}_0, \to_0)$, $[\![P_1]\!] = \langle \mathcal{E}_1, \mathsf{Init}_1, k_1 \rangle$, and $C_{br}(\mathcal{E}_1) = (E_1, F_1, \mathsf{C}_1, \to_1)$. Then either $e = (Y, (e_0, *))$, $e = (Y, (*, e_1))$, or $e = (Y, (e_0, e_1))$.

  If $e = (Y, (e_0, *))$, then whenever $X'_0 \mapsto_0 e_0$, there exists $e' \in Y$ such that $\pi_0(e') \in X_0$ and $\{e'\} \mapsto e$. And whenever $\pi_0(e') \rhd_0 \pi_0(e)$, we get $e' \rhd e$. This means Init_0 is conflict-free, $\pi_0(X)$ is conflict-free, and $\mathsf{Init}_0 \xrightarrow{e_0}_0 \pi_0(X)$. There therefore exists a key m and a transition $P_0 \xrightarrow{\lambda_0(e_0)[m]} P'_0$, such that $\left\{\!\!\left[P'_0 \right]\!\!\right\} = \left\langle \mathcal{E}'_0, \mathsf{Init}'_0, k'_0 \right\rangle$ and there exist isomorphisms $f_0 : \mathcal{E}_0 \to \mathcal{E}'_0$ and $g_0 : \mathcal{E}'_0 \to \mathcal{E}_0$ such that $k'_0(f_0(e_0)) = m$ and $f_0(\pi_0(X)) = \mathsf{Init}'_0$.

  We chose an m, which is fresh for P_1, and we get $P \xrightarrow{\lambda_0(e_0)[m]} P'_0 \mid P_1$. We define our isomorphisms similarly to the corresponding case in Theorem 7.13, and the proof of them being isomorphisms is similar.

  If $e = (Y, (*, e_1))$, the argument is similar.

  If $e = (Y, (e_0, e_1))$, then for $i \in \{0, 1\}$, whenever $X'_i \mapsto_i e_i$, there exists $e' \in Y$ such that $\pi_i(e') \in X'_i$ and $\{e'\} \mapsto e$. And whenever $\pi_i(e') \sharp_i \pi_i(e)$, we get $e' \sharp e$. This means Init_i is conflict-free, $\pi_i(X)$ is conflict-free, and $\mathsf{Init}_i \xrightarrow{e_0}_i$

$\pi_i(X)$. There therefore exists a key m_i and a transition $P_i \xrightarrow{\lambda_i(e_i)[m_i]} P_i'$, such that $\{\!|P_i'|\!\} = \langle \mathcal{E}_i', \mathsf{Init}_i', k_i' \rangle$ and there exist isomorphisms $f_i : \mathcal{E}_0 \to \mathcal{E}_i'$ and $g_i : \mathcal{E}_i' \to \mathcal{E}_i$ such that $k_i'(f_i(e_i)) = m_i$ and $f_i(\pi_i(X)) = \mathsf{Init}_0'$.

We say that $m_0 = m_1$ is a fresh m, and then since $\lambda_0(e_0) = \overline{\lambda_1(e_1)}$ and $\lambda(e) = \tau$, we get $P \xrightarrow{\lambda(e)[m]} P_0' \mid P_1'$. We define our isomorphisms similarly to the corresponding case in Theorem 7.13, and the proof of them being isomorphism is similar to that case. The rest of the case is straightforward.

- Suppose $P = P'' \setminus A$, $\{\!|P''|\!\} = \langle \mathcal{E}'', \mathsf{Init}, k \rangle$, and $C_{br}(\mathcal{E}'') = (E'', F'', \mathsf{C}'', \to'')$. Then $\lambda(e) \notin A \cup \overline{A}$ and there exists at least one $Y \in \mathsf{cause}(e)$ such that if $e' \in Y$ then $\lambda(e') \notin (A \cup \overline{A})$. And since P is reachable, for all $e' \in \mathsf{Init}$, $\lambda(e') \notin (A \cup \overline{A})$. We therefore know $\mathsf{Init}'' = \mathsf{Init} \xrightarrow{e}{}'' X$, meaning there exists a key n and a transition $P'' \xrightarrow{\lambda''(e)} P'''$ such that $\{\!|P'''|\!\} = \langle \mathcal{E}''', \mathsf{Init}''', k''' \rangle$, and there exist isomorphisms $f' : \mathcal{E}'' \to \mathcal{E}'''$ and $f' : \mathcal{E}''' \to \mathcal{E}''$ such that $f' \circ k''' = \cup\{(e, n)\}$ and $f'(X) = \mathsf{Init}'''$.

  This means $P \xrightarrow{\lambda''(e)} P''' \setminus A$ and the morphisms $f \upharpoonright E$ and $g \upharpoonright E''' \cap \rho(A \cup \overline{A})$ clearly fulfil the remaining conditions.

- Suppose $P = P''[f]$, $\{\!|P''|\!\} = \langle \mathcal{E}'', \mathsf{Init}, k \rangle$. Then the case is similar to the corresponding case of Theorem 4.19. □

F.11. Proof of Theorem 7.22

Proof. We prove this through induction on the derivation of $P \overset{\mathsf{roll}\ n}{\rightsquigarrow} P'$ by constructing $\mathcal{E}$, $\mathcal{E}'$, f and g for each case:

(act ROLL): Suppose $P = \alpha_\gamma[n].R$, $R \overset{\mathsf{roll}\ n}{\rightsquigarrow} R'$, and $P' = \alpha_\gamma.R_{\not\downarrow \{n' | n \leq_P n'\}}$, with $\{\!|R|\!\} = \langle \mathcal{E}_R, \mathsf{Init}_R, k_R \rangle$ and $\left\{\!\left| R_{\not\downarrow \{n' | n \leq_P n'\}} \right|\!\right\} = \langle \mathcal{E}_{R'}, \mathsf{Init}_{R'}, k_{R'} \rangle$. Then $\{n' \mid n \leq_P n'\}$ is n and all n's for which $\beta_{\gamma'}[n']$ occurs in P. It is clear from the semantic rules that this means $\mathsf{Init}' = \{e \mid \lambda(e) \in \{\mathsf{start\ roll}\ \gamma' \mid \mathsf{rolling}\ \gamma'$ occurs in P and $\nexists \beta, n.\beta_{\gamma'}$ or $\beta_{\gamma'}[m]$ occurs in $P\}$ and there exists an isomorphism $f : \mathcal{E} \to \mathcal{E}'$.

In addition we have $e, e_r \in E$ such that $\lambda(e) = \alpha$, $k(e) = n$, $\lambda(e_r) = \mathsf{roll}\ \gamma$, $\mathsf{N}(e_r) = \mathsf{roll}\ n$, and $\{e_r\} \mapsto \underline{e}$. By Lemmas 7.6, 7.10, and 7.11. we then get $\mathsf{Init} \xrightarrow{\{e_r\}} X_0 \xrightarrow{\{\underline{e_0}\}} X_1 \cdots \xrightarrow{\{\underline{e_n}\}} X_{n+1} \xrightarrow{\{\underline{e_r}\}} X_{\mathsf{done}}$ fulfilling the conditions.

(par ROLL): Suppose $P = Q \mid R$, $P' = (Q \mid R)_{\not\downarrow \{n' | n \leq_P n'\}}$, $Q \overset{\mathsf{roll}\ \gamma}{\rightsquigarrow} Q'$, and we generate event structures as follows: $\{\!|Q|\!\} = \langle \mathcal{E}_Q, \mathsf{Init}_Q, k_Q \rangle$, $\{\!|R|\!\} = \langle \mathcal{E}_R, \mathsf{Init}_R, k_R \rangle$, $\left\{\!\left| Q_{\not\downarrow \{n' | n \leq_P n'\}} \right|\!\right\} = \left\langle \mathcal{E}_Q', \mathsf{Init}_Q', k_Q' \right\rangle$, $\left\{\!\left| R_{\not\downarrow \{n' | n \leq_P n'\}} \right|\!\right\} = \left\langle \mathcal{E}_R', \mathsf{Init}_R', k_R' \right\rangle$, and we

construct $\langle \mathcal{E}, \mathsf{Init}, k\rangle$ from $\left\langle \mathcal{E}_Q, \mathsf{Init}_Q, k_Q \right\rangle$ and $\langle \mathcal{E}_R, \mathsf{Init}_R, k_R\rangle$ and $\left\langle \mathcal{E}', \mathsf{Init}', k' \right\rangle$ from $\left\langle \mathcal{E}'_Q, \mathsf{Init}'_Q, k'_Q \right\rangle$ and $\left\langle \mathcal{E}'_R, \mathsf{Init}'_R, k'_R \right\rangle$ as described in the semantics.

It is clear from the semantics that there exists an isomorphism $f : \mathcal{E} \to \mathcal{E}'$.

By induction we have $\mathsf{Init}_Q \xrightarrow{\{e_r\}} X_{(0,Q)} \xrightarrow{\{\underline{e_0}\}} X_{(1,Q)} \cdots \xrightarrow{\{\underline{e_m}\}} X_{(m+1,Q)} \xrightarrow{\{\underline{e_r}\}} X_{(done,Q)}$, $\{e_0, e_1, \ldots e_n\} = \{e' \mid n \leq_Q k_Q(e')\}$, and there exists an isomorphism $f_Q : \mathcal{E}_Q \to \mathcal{E}'_Q$ such that $f(X_{(done,Q)}) = \mathsf{Init}'_Q$.

From this we get that $(e_r, *) \in E$, and for each $X \mapsto (e_r, *)$, we have an $X_Q \mapsto e_r$ such that $X = \{e \in E \mid \pi_Q(e) \in X_Q\}$, and therefore $X \cap \mathsf{Init} \neq \emptyset$. Additionally, if $e \rhd (e_r, *)$, then either $\pi_Q(e) \rhd e_r$, or $\lambda(e) = \mathsf{roll}\ \gamma'$, meaning $e \notin \mathsf{Init}$. We therefore get $\mathsf{Init} \xrightarrow{\{(e_r,*)\}}$. Since by Lemma 7.17 $\nexists e' \in I.\lambda(e') = \mathsf{roll}\ \gamma'$, we get that by Lemma 7.18 for e_i, $0 \leq i \leq n$, whenever $X_i \mapsto e_i$, either $X_i = \{e_i\}$, or $e_r \in X_i$, meaning for any $e \in E$ such that $\pi_Q(e) \in \{e_0, e_1, \ldots e_n\}$, whenever $X \mapsto e$, either $X = \{e\}$ or $(e_r, *) \in X$. The rest follows from Lemma 7.11 and Proposition 7.19.

(prop ROLL Key 1): Suppose $P = \beta'_\gamma[m].R$, $P' = \beta'_\gamma[m].R'$, $m' \neq n$, $R \overset{\mathsf{roll}\ n}{\rightsquigarrow} R'$, $\{\!|R|\!\} = \langle \mathcal{E}_R, \mathsf{Init}_R, k_R\rangle$, and $\{\!|R'|\!\} = \langle \mathcal{E}_{R'}, \mathsf{Init}_{R'}, k_{R'}\rangle$. Then by induction $\mathsf{Init}_R \xrightarrow{\{e_r\}} X_0 \xrightarrow{\{\underline{e_0}\}} X_1 \cdots \xrightarrow{\{\underline{e_n}\}} X_{n+1} \xrightarrow{\{\underline{e_r}\}} X_{\mathsf{done}}$, and there exists an isomorphism $f_R : \mathcal{E}_R \to \mathcal{E}_{R'}$ fulfilling the conditions. Then it is clear from the semantics that the result holds using the isomorphism $f = f_R \cup \{(e_\alpha, e'_\alpha)\}$.

(prop ROLL Key 2): Suppose $P = \beta'_\gamma.R$, $P' = \beta'_\gamma.R'$, $\{\!|R|\!\} = \langle \mathcal{E}_R, \mathsf{Init}_R, k_R\rangle$, $\{\!|R'|\!\} = \langle \mathcal{E}_{R'}, \mathsf{Init}_{R'}, k_{R'}\rangle$, and $R \overset{\mathsf{roll}\ n}{\rightsquigarrow} R'$. Then $\mathsf{Init}_R \xrightarrow{\{e_r\}} X_0 \xrightarrow{\{\underline{e_0}\}} X_1 \cdots \xrightarrow{\{\underline{e_n}\}} X_{n+1} \xrightarrow{\{\underline{e_r}\}} X_{\mathsf{done}}$, and there exists an isomorphism $f_R : \mathcal{E}_R \to \mathcal{E}_{R'}$ such that $f_R(X_d) = \mathsf{Init}_{R'}$. Then it is clear from the semantics that the result holds using the isomorphism $f = f_R \cup \{(e_\alpha, e'_\alpha)\}$.

(prop ROLL Key 3): Suppose $P = P_0 + P_1$, $P' = P'_0 + P_1$, $\{\!|P_0|\!\} = \langle \mathcal{E}_0, \mathsf{Init}_0, k_0\rangle$, $\{\!|P_1|\!\} = \langle \mathcal{E}_1, \mathsf{Init}_1, k_1\rangle$ $\left\{\!\left|P'_0\right|\!\right\} = \left\langle \mathcal{E}'_0, \mathsf{Init}'_0, k'_0 \right\rangle$, and $P_0 \overset{\mathsf{roll}\ n}{\rightsquigarrow} P'_0$. Then by induction $\mathsf{Init}_0 \xrightarrow{\{e_{(r,0)}\}} X_{(0,0)} \xrightarrow{\{\underline{e_{(0,0)}}\}} X_{(1,0)} \cdots \xrightarrow{\{\underline{e_{(n,0)}}\}} X_{(n+1,0)} \xrightarrow{\{\underline{e_{(r,0)}}\}} X_{(done,0)}$, and there exists an isomorphism $f_0 : \mathcal{E}_0 \to \mathcal{E}'_0$ fulfilling the conditions. Then, since P is consistent, $\mathsf{std}(P_1)$, and therefore $\{0\}\times\mathsf{Init}_0 \xrightarrow{\{(0,e_{(r,0)})\}} \{0\}\times X_{(0,0)} \xrightarrow{\{\underline{(0,e_{(0,0)})}\}} \{0\} \times X_{(1,0)} \cdots \xrightarrow{\{\underline{(0,e_{(n,0)})}\}} \{0\} \times X_{(n+1,0)} \xrightarrow{\{\underline{(0,e_{(r,0)})}\}} \{0\} \times X_{(done,0)}$, and the rest obviously holds.

(prop ROLL Key 4): Suppose $P = R \setminus A$, $P' = R' \setminus A$, $\{\!|R|\!\} = \langle \mathcal{E}_R, \mathsf{Init}_R, k_R\rangle$, $\{\!|R'|\!\} = \langle \mathcal{E}_{R'}, \mathsf{Init}_{R'}, k_{R'}\rangle$, and $R \overset{\mathsf{roll}\ n}{\rightsquigarrow} R'$. Then by induction $\mathsf{Init}_R \xrightarrow{\{e_r\}}$

$X_0 \xrightarrow{\{\underline{e_0}\}} X_1 \cdots \xrightarrow{\{\underline{e_n}\}} X_{n+1} \xrightarrow{\{\underline{e_r}\}} X_{\mathsf{done}}$, and there exists an isomorphism $f_R : \mathcal{E}_R \to \mathcal{E}_{R'}$ such that $f_R(X_d) = \mathsf{Init}_{R'}$. Then, since P is consistent, if $\alpha_\gamma[n]$ occurs in R, $\alpha \notin A \cup \overline{A}$, and by Theorem 7.13, whenever $e \in \mathsf{Init}_R$, there exists $X \in \mathsf{cause}(e)$ such that $X \subseteq \rho(A \cup \overline{A})$, and the result follows.

(prop ROLL Key 5): Suppose $P = R[f]$, $P' = R'[f]$, $\{\!|R|\!\} = \langle \mathcal{E}_R, \mathsf{Init}_R, k_R \rangle$, $\{\!|R'|\!\} = \langle \mathcal{E}_{R'}, \mathsf{Init}_{R'}, k_{R'} \rangle$, $\gamma' \neq \gamma$, and $R \overset{\mathsf{roll}\ n}{\rightsquigarrow} R'$. Then by induction $\mathsf{Init}_R \xrightarrow{\{\underline{e_r}\}} X_0 \xrightarrow{\{\underline{e_0}\}} X_1 \cdots \xrightarrow{\{\underline{e_n}\}} X_{n+1} \xrightarrow{\{\underline{e_r}\}} X_{\mathsf{done}}$, and there exists an isomorphism $f_R : \mathcal{E}_R \to \mathcal{E}_{R'}$ fulfilling the conditions, and the result follows.

(prop ROLL Key 6): Suppose $P \equiv Q$, $Q \overset{\mathsf{roll}\ n}{\rightsquigarrow} Q'$, and $Q' \equiv P$. Then the result follows from Proposition 7.12. □