Journal of Logical and Algebraic Methods in Programming 121 (2021) 100675

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

On primitives for compensation handling as adaptable processes

Jovana Dedeić [a,*], Jovanka Pantović [a,*], Jorge A. Pérez [b,c]

[a] University of Novi Sad, Serbia
[b] University of Groningen, the Netherlands
[c] CWI, Amsterdam, the Netherlands

A R T I C L E   I N F O

Article history:
Received 12 April 2020
Received in revised form 13 March 2021
Accepted 17 March 2021
Available online 23 March 2021

Keywords:
Concurrency
Process calculi
Relative expressiveness
Compensation handling
Dynamic update

A B S T R A C T

Mechanisms for *compensation handling* and *dynamic update* are increasingly relevant in the specification of reliable communicating systems. Compensations and updates are intuitively similar: both specify how the behavior of a concurrent system changes at runtime in response to an exceptional event. However, calculi for concurrency with compensations and updates are technically quite different.

We compare calculi for concurrency with compensation handling and dynamic update from the standpoint of their *relative expressiveness*. We develop two encodings of a process calculus with compensation handling into a calculus of adaptable processes. These encodings differ in the target language considered: the first considers adaptable processes with *subjective updates* in which, intuitively, a process reconfigures itself; the second considers *objective updates* in which a process is reconfigured by a process in its context.

Our main discovery is that subjective updates are more *efficient* than objective ones in encoding primitives for compensation handling: the first encoding requires less computational steps than the second one to mimic a single computation step in the source language of compensable processes. Our encodings satisfy strong correctness criteria; they shed light on the intricate semantics of compensation handling.

1. Introduction

Many software applications are based on *long-running transactions* (LRTs). Frequently found in service-oriented systems [11], LRTs are computing activities which extend in time and may involve distributed, loosely coupled resources. These features sharply distinguish LRTs from traditional database-like transactions. One particularly delicate aspect of LRTs is managing (partial) failures: mechanisms for detecting failures and bringing the LRT back to a consistent state need to be explicitly programmed. Because ensuring the correctness of such mechanisms is error prone, specialized constructs, such as *exceptions* and *compensations*, have been put forward to offer direct programming support for LRTs. For instance, in Java we find the construct try P catch(e) Q, where Q is in charge of managing exceptions e raised inside P; in WS-BPEL [1] we find advanced mechanisms exploiting fault, termination, and compensation handlers to handle errors. In this paper, our focus is in compensation mechanisms: as their name suggests, they are meant to compensate the fact that an LRT has failed or has been canceled. Upon receiving a failure signal, a compensation mechanism is expected to install and activate alternative behaviors for recovering system consistency. Such a compensation behavior may be different from the LRT's initial behavior.

* Corresponding authors.
E-mail address: radenovicj@uns.ac.rs (J. Dedeić).

https://doi.org/10.1016/j.jlamp.2021.100675
2352-2208/© 2021 Elsevier Inc. All rights reserved.

A variety of calculi for concurrency with constructs for compensation handling has been proposed (see, e.g. [2,5,17,6,11]). Building upon process calculi such as CCS [18], CSP [14], and the π-calculus [19], they capture different forms of error recovery and offer reasoning techniques (e.g., behavioral equivalences) on communicating processes with compensation constructs. The relationships between the different proposals are not clear, and there has been limited work aimed to formally comparing the expressiveness of the proposed mechanisms—relevant studies in this direction include those in [6,4,15,16]. In particular, Lanese et al. [15] develop a formal comparison of different approaches to LRTs in a concurrent and mobile setting. They consider a process language on top of which different primitives for error handling, distilled from the vast literature on the subject, are uniformly considered. This approach naturally leads to clear comparisons.

More in details, Lanese et al. defined a core calculus of compensable processes, which extends the π-calculus with *transactions* $t[P,Q]$ (where processes P and Q represent default and compensation activities, respectively), *protected blocks* $\langle Q\rangle$, and *compensation updates* $\mathtt{inst}\lfloor\lambda X.Q\rfloor.P$, which reconfigure a compensation activity. One key merit of their calculus is that different proposals arise as instances. To this end, compensations may admit *static* or *dynamic* recovery (depending on whether compensation updates are allowed) and the response to failures can be specified via *preserving*, *discarding*, and *aborting* semantics. The process language in [15] thus leads to six distinct calculi with compensation primitives.

Related to compensation handling, but on a somewhat different vein, a process calculus of *adaptable processes* was proposed by Bravetti et al. [3], with the aim of specifying *dynamic update* in communicating systems. Adaptable processes specify forms of dynamic reconfiguration that are triggered by exceptional events, not necessarily catastrophic. A simple example is the reconfiguration of specific units of a robot swarm, which is usually hard to predict and entails modifying the device's behavior; still, it is certainly not a failure. Adaptable processes can be deployed in *locations*, which serve as delimiters for dynamic updates. A process P *located at* l, denoted $l[P]$, can be reconfigured by an *update prefix* $l\{(X).Q\}.R$, where Q denotes an adaptation routine for l, parameterized by variable X.

Using located processes and update prefixes, dynamic update in [3] is realized by the following reduction rule, in which C_1 and C_2 denote contexts of arbitrarily nested locations:

$$C_1\big[l[P]\big] \mid C_2\big[l\{(X).Q\}.R\big] \longrightarrow C_1\big[Q\{^P/_X\}\big] \mid C_2\big[R\big] \tag{1}$$

We call this an ***objective update***: a located process is reconfigured by an update prefix at a different context. Indeed, the update prefix $l\{(X).Q\}$ *moves* from C_2 to C_1, and the reconfigured behavior $Q\{^P/_X\}$ is left in C_1. Notice that X may occur zero or many times in Q; if Q does not contain X then the current behavior P will be erased as a result of the update. This way, dynamic update is a form of process mobility, implemented using *higher-order process communication* as found in languages such as, e.g., the higher-order π-calculus [22], the Kell calculus [23], and Homer [13].

An alternative to objective update is ***subjective update***, in which process reconfiguration flows in the opposite direction: it is the located process the one to move to a (remote) context enclosing an update prefix, as expressed by the following reduction rule:

$$C_1\big[l[P] \mid R_1\big] \mid C_2\big[l\{(X).Q\}.R\big] \longrightarrow C_1\big[\mathbf{0} \mid R_1\big] \mid C_2\big[Q\{^P/_X\} \mid R\big] \tag{2}$$

As objective update, subjective update relies on process mobility; however, the direction of movement is different: above, P moves from C_1 to C_2, and the reconfigured behavior $Q\{^P/_X\}$ is left in C_2, not in C_1. Thus, in a subjective update the located process "reconfigures itself", which makes for a more autonomous semantics for adaptation than objective updates.[1]

Example 1.1. We contrast objective and subjective update by means of an example, adapted from [3]. Consider an *interrupt* operator that starts executing process P but may abandon its execution to execute Q instead; once Q emits a termination signal t_Q, the operator returns to execute what is left of P. Using adaptable processes, this kind of behavior can be expressed as follows:

$$Sys = l_1\big[l[P] \mid R_1\big] \mid l_2\big[l\{(X).Q \mid t_Q.X\}.R_2\big]$$

where l, l_1, and l_2 are different locations and name t_Q is only known to Q. If P evolves into P' right before being interrupted, under a semantics with *objective update* we have

$$\begin{aligned} Sys &\longrightarrow^* l_1\big[l[P'] \mid R_1\big] \mid l_2\big[l\{(X).Q \mid t_Q.X\}.R_2\big] \\ &\longrightarrow l_1\big[Q \mid t_Q.P' \mid R_1\big] \mid l_2\big[R_2\big] \\ &\longrightarrow^* l_1\big[P' \mid R_1\big] \mid l_2\big[R_2\big] \end{aligned}$$

This way, P and its derivative P' reside at location l_1. Notice that executing Sys under a semantics with subjective update would yield a different behavior, because P' (and Q) would be wrongly moved to l_2:

[1] We use adjectives 'subjective' and 'objective' for updates following the distinction between subjective and objective *mobility*, as in calculi such as Ambients [7] and Seal [8]. As explained in [8], Ambients use subjective mobility (an agent moves itself), while Seal uses objective mobility (an agent is moved by its context).

$$\begin{aligned} Sys &\longrightarrow^* l_1\big[l[P'] \mid R_1\big] \mid l_2\big[l\{(X).Q \mid t_Q.X\}.R_2\big] \\ &\longrightarrow l_1\big[R_1\big] \mid l_2\big[Q \mid t_Q.P' \mid R_2\big] \\ &\longrightarrow^* l_1\big[R_1\big] \mid l_2\big[P' \mid R_2\big] \end{aligned}$$

This shows that to achieve the intended interrupt behavior in a subjective setting, Sys should be modified in order to eventually bring process P' back to l_1. The following variation of Sys achieves this:

$$Sys' = l_1\big[l[P] \mid l'\{(X).X\}.R_1\big] \mid l_2\big[l\{(X).l'[Q \mid t_Q.X]\}.R_2\big]$$

where we use l' as an auxiliary location that "pulls back" P' from l_2 into l_1.

The aim of this paper is to compare process calculi with compensation handling (as formalized in [15]) and with dynamic update (as formalized in [3]), from the point of view of *relative expressiveness*. There are good reasons for focusing on compensation handling as in [15] and on dynamic update as in [3]. On the one hand, the calculus of compensable processes in [15] is expressive enough to capture several different languages proposed in the literature; the analyses of expressiveness in [15] are exhaustive and bring uniformity to the study of formal models for LTRs. Because of its expressiveness, this calculus provides an appropriate starting point for further investigations. On the other hand, as we have seen, the calculus of adaptable processes in [3] is a simple process model of dynamic adaptation, based on a few process constructs and endowed with a clean operational (reduction) semantics, which supports both objective and subjective updates. (In contrast, as we will see, the calculus of compensable processes relies on an intricate Labeled Transition System.) As such, adaptable processes provide a flexible framework to elucidate the underpinnings of compensation handling, from a fresh perspective.

Contributions. In this paper, we present the following contributions:

1. We develop two *translations* of a core calculus with compensation handling with discarding semantics [15] into adaptable processes [3]: while the first translation relies on a calculus with objective updates, the second exploits subjective updates.
2. We establish that the two translations are *valid encodings* [12], i.e., they satisfy structural properties (compositionality and name invariance) and semantic properties (operational correspondence, divergence reflection, and success sensitiveness) that bear witness to their robustness.
3. We exploit the correctness properties of our two encodings to clearly distinguish between subjective and objective updates in calculi for concurrency. We introduce an encodability criterion called *efficiency*, which allows us to formally state that subjective updates are better suited to encode compensation handling than objective updates, because they induce tighter operational correspondences.

Points (1) and (3) deserve further explanations. Concerning (3), our encoding into adaptable processes with objective updates reveals a limitation: in representing the collection of protected blocks scattered within nested transactions, objective updates leave behind processes in the "wrong" location. The situation is reminiscent of the differences shown in Example 1.1 for the interrupt behavior. To remedy this, the encoding uses additional synchronizations to move processes to the right locations. This reflects prominently in the *cost* of the encoding, i.e., the number of target computation steps required to mimic a source computation step (this number is spelled out precisely by our operational correspondence results). The encoding into the calculus with subjective updates does not require these additional synchronizations, and so it is more efficient than the encoding that uses objective update.

Concerning (1), while we focus on compensable processes with discarding semantics, we also consider the cases in which the source calculus uses preserving semantics, aborting semantics, and dynamic compensations. § 8 discusses how our encoding can account for these three variations. In all cases, the target language uses subjective updates, which, as just discussed, are more efficient than objective update.

Outline. The paper is organized as follows. In § 2 we informally present the syntax and semantics of the source and target calculi; § 3 gives a formal introduction. § 4 introduces the correctness criteria for encodings. Then, we present our two encodings and prove them correct: § 5 considers a target calculus with subjective update and § 6 considers the case with objective update. § 7 compares the two encodings by formalizing the efficiency claim. § 8 discusses additional encodings, involving a source calculus with preserving and aborting semantics, and with dynamic update. § 9 discusses related works and § 10 collects some concluding remarks. The appendices collect omitted proofs for our technical results.

Origin of the results. This paper distills, improves, and collects preliminary results from our papers [9] and [10]. While in [9] we studied encodings into adaptable processes with objective updates, in [10] we studied encodings into adaptable processes with subjective updates, and compared them against those in [9]. A main difference between [9,10] and the current paper is that here we concentrate on a specific source calculus, namely the calculus in [15] with *static recovery* and *discarding semantics*. Indeed, the developments in [9,10] consider also source calculi with *dynamic recovery* and/or *preserving* and *aborting semantics*. The calculus with static recovery and discarding semantics arguably defines the simplest setting for both encodings, one in which the key differences between compensable and adaptable processes can be more sharply presented. Also, this focus allows us to have a concise presentation. As we discuss in §8, the (efficient) encoding in §5 extends to source calculi with the semantics we considered in [9] and [10].

2. Compensable and adaptable processes, by example

We give an intuitive account of the core calculus with primitives for *compensation handling* (as presented by Lanese et al. [15,16]) and the calculus of *adaptable processes* (introduced by Bravetti et al. [3]). In both cases, we illustrate their most salient features by means of a simple example.

2.1. Compensable processes

The process language with compensations that we consider is based on the calculus in [16] (which is, in turn, a variant of the language in [15]). The languages in [16,15] were introduced as extensions of the π-calculus with primitives for *static* and *dynamic recovery*. We consider a variant with static recovery and without name mobility; this allows us to focus on the fundamental aspects of compensations. The languages in [16,15] feature two distinguishing constructs:

1. *Transactions* $t[P,Q]$, where t is a name and P, Q are processes;
2. *Protected blocks* $\langle Q \rangle$, where Q is a process.

A transaction $t[P,Q]$ consists of a *default activity* P and a *compensation activity* Q. Transactions can be nested: process P in $t[P,Q]$ may contain other transactions. Also, they can be canceled: process $t[P,Q]$ behaves as P until an *error notification* (failure signal) arrives along name t. Error notifications are output messages coming from inside or outside the transaction; to illustrate this, consider the following transitions:

$$t[P,Q] \mid \bar{t}.R \xrightarrow{\tau} Q \mid R \qquad\qquad t[\bar{t}.P_1 \mid P_2, Q] \xrightarrow{\tau} Q \tag{3}$$

The left (resp. right) transition shows how t can be canceled by an external (resp. internal) signal. Failure discards the default behavior; the compensation activity is executed instead. In both cases, the default activity is discarded entirely. This may not be desirable in all cases; after a compensation is enabled, we may like to preserve (some of) the behavior in the default activity. To this end, one can use *protected blocks*: processes Q and $\langle Q \rangle$ have the same behavior, but $\langle Q \rangle$ is not affected by failure signals. This way, the transition

$$t_2[P_2, Q_2] \mid \overline{t_2} \xrightarrow{\tau} \langle Q_2 \rangle,$$

says that the compensation behavior Q_2 will be immune to failures. Consider now process

$$P = t_1\big[t_2[P_2, Q_2] \mid \overline{t_2}.R_1, Q_1\big],$$

in which transaction t_2 occurs nested inside t_1 and P_2 does not contain protected blocks. The labeled transition system (LTS) in [16,15] refines (3) by providing ways to (partially) preserve behavior after a compensation step. This is realized by the *extraction function* on processes, denoted $\mathsf{extr}(\cdot)$. For process P, the semantics in [16,15] decree:

$$t_1\big[t_2[P_2, Q_2] \mid \overline{t_2}.R_1, Q_1\big] \xrightarrow{\tau} t_1\big[\langle Q_2 \rangle \mid \mathsf{extr}(P_2) \mid R_1, Q_1\big].$$

There are different choices for this extraction function: in the *discarding* semantics that we consider here, only top-level protected blocks are preserved (cf. Fig. 1); hence, in the example above, $\mathsf{extr}(P_2) = \mathbf{0}$. The languages in [16,15] include extraction functions for *preserving* and *aborting* semantics that would preserve also (top-level) transactions in P_2. To further illustrate the extraction function, consider the process:

$$P' = t\big[t_1[P_1, Q_1] \mid t_2[\langle P_2 \rangle, Q_2] \mid \langle P_3 \rangle, Q_5\big]. \tag{4}$$

We would have $\bar{t} \mid P' \xrightarrow{\tau} \langle P_3 \rangle \mid \langle Q_5 \rangle$. Thus, the discarding semantics only concerns the compensation activity for transaction t and the protected block $\langle P_3 \rangle$; the protected block $\langle P_2 \rangle$, nested inside t_2, is discarded.

With these intuitions in place, we illustrate compensable processes by means of an example that we will use throughout the paper:

Example 2.1 *(Hotel booking scenario).* Consider a simple hotel booking scenario in which a hotel and a client interact to book and pay a room, and to exchange an invoice. This scenario may be represented using compensable processes as follows (below we omit trailing $\mathbf{0}$s):

$$\begin{aligned}
Reservation &\overset{def}{=} Hotel \mid Client \\
Client &\overset{def}{=} \overline{book}.\overline{pay}.(invoice + \overline{t}.refund) \\
Hotel &\overset{def}{=} t[book.pay.\overline{invoice}, \overline{refund}]
\end{aligned}$$

Here we represent the hotel's behavior as a transaction t that allows clients to book a room and pay for it. If the client is satisfied with the reservation, then the hotel will send her an invoice. Otherwise, the client may cancel the transaction; in that case, hotel offers the client a refund. Suppose that the client decides to cancel his reservation; as we will see, there are four transition steps for process *Reservation*:

$$\begin{aligned}
Reservation &\xrightarrow{\tau} t[pay.\overline{invoice}, \overline{refund}] \mid \overline{pay}.(invoice + \overline{t}.refund) \\
&\xrightarrow{\tau} t[\overline{invoice}, \overline{refund}] \mid invoice + \overline{t}.refund \xrightarrow{\tau} \langle \overline{refund} \rangle \mid refund \xrightarrow{\tau} \langle \mathbf{0} \rangle.
\end{aligned}$$

2.2. Adaptable processes

The calculus of adaptable processes was introduced as a variant of Milner's CCS [18] (without restriction and relabeling), extended with the following two constructs, aimed at representing the dynamic reconfiguration (or update) of communicating processes:

1. A *located process*, denoted $l[P]$, represents a process P which resides in a location called l. Locations can be arbitrarily *nested*, which allows to organize process descriptions into meaningful hierarchical structures.
2. *Update prefixes* specify an adaptation mechanism for processes at location l. We write $l\langle\langle (X).Q \rangle\rangle$ and $l\{(X).Q\}$ to denote subjective and objective update prefixes; in both cases, X is a process variable that occurs zero or more times in Q.

This way, in the calculus of adaptable processes the update of a (located) process is given the same status as point-to-point communication. That is, an update prefix for location l can interact with a located process at l to update its current behavior. Depending on the kind of prefix (objective or subjective), this interaction is realized by a reduction rule ((1) or (2), see also below).

We illustrate adaptable processes by revisiting the example above:

Example 2.2. Consider again the hotel booking scenario in Example 2.1, this time expressed using the calculus of adaptable processes (below we omit trailing **0**s):

$$\begin{aligned}
Reservation &\overset{def}{=} Hotel \mid Client \\
Client &\overset{def}{=} \overline{book}.\overline{pay}.(\overline{t}.refund + invoice) \\
Hotel &\overset{def}{=} t[book.pay.\overline{invoice}] \mid t.t\langle\langle (Y).\mathbf{0} \rangle\rangle \mid p_t[\overline{refund}]
\end{aligned}$$

We use CCS processes with the located processes and (subjective) update prefixes. The client's behavior involves sending requests for booking and paying for a room, which are followed by either the reception of an invoice or an output on t signaling the end of the transaction and the request for a refund. The expected behavior of the hotel is located at location t: the hotel allows the client to book a room and pay for it; if the client is satisfied with the reservation, the hotel will send him/her an invoice. The hotel specification includes also (i) a subjective update prefix $t\langle\langle (Y).\mathbf{0} \rangle\rangle$ (in the same way, can be used objective update $t\{(Y).\mathbf{0}\}$), which deletes the location t with its content if the client is not satisfied with the reservation, and (ii) a simple refund procedure located at p_t, which handles the interaction with the client in that scenario.

If the client decides to cancel his reservation, the reduction steps for process *Reservation* would be as follows:

$$\begin{aligned}
Reservation &\longrightarrow t[pay.\overline{invoice}] \mid t.t\langle\langle (Y).\mathbf{0} \rangle\rangle \mid p_t[\overline{refund}] \mid \overline{pay}.(\overline{t}.refund + invoice) \\
&\longrightarrow t[\overline{invoice}] \mid t.t\langle\langle (Y).\mathbf{0} \rangle\rangle \mid p_t[\overline{refund}] \mid \overline{t}.refund + invoice \\
&\longrightarrow t[\overline{invoice}] \mid t\langle\langle (Y).\mathbf{0} \rangle\rangle \mid p_t[\overline{refund}] \mid refund \longrightarrow p_t[\overline{refund}] \mid refund \longrightarrow p_t[\mathbf{0}].
\end{aligned}$$

In this example we could have used objective update $t\{(Y).\mathbf{0}\}$ instead of subjective update $t\langle\langle (Y).\mathbf{0} \rangle\rangle$; with objective update, the behavior of process *Reservation* is quite similar. A detailed derivation and explanation for this scenario will be provided later on, once we have formally defined our translations.

3. The calculi

We now introduce formally compensable processes (§ 3.1) and adaptable processes (§ 3.3). To focus on their essentials, both calculi are defined as extensions of CCS [18] (no name passing is considered). In § 3.2 we identify a class of *well-formed* compensable processes, useful in our developments.

$$\mathsf{extr}(t[P, Q]) = \mathbf{0} \qquad \mathsf{extr}(\langle P\rangle) = \langle P\rangle \qquad \mathsf{extr}(P \mid Q) = \mathsf{extr}(P) \mid \mathsf{extr}(Q)$$
$$\mathsf{extr}((\nu x)P) = (\nu x)\mathsf{extr}(P) \qquad \mathsf{extr}(\mathbf{0}) = \mathsf{extr}(\pi.P) = \mathbf{0} \qquad \mathsf{extr}(!\pi.P) = \mathbf{0}$$

Fig. 1. Extraction function.

We start by defining some relevant base sets for names.

Definition 3.1 *(Base sets).* We assume the following countable sets of names:

- $\mathcal{N}_t$ is a finite set of *transaction names*, ranged over by $t, t', s, s', \ldots$, also used as *error notification names*;
- $\mathcal{N}_l$ is a set of *location names*, ranged over by $l, l', t, t', s, s', \ldots$, also used as *input names*;
- $\mathcal{N}_s$ is the set that collects all other (input/output) names, ranged over by $a, b, c, \ldots$.

For compensable processes, we shall use the set $\mathcal{N}_c = \mathcal{N}_t \cup \mathcal{N}_s$; for adaptable processes, we shall use the set $\mathcal{N}_a = \mathcal{N}_l \cup \mathcal{N}_s$. Some assumptions on these sets are in order. First, $\mathcal{N}_l \cap \mathcal{N}_s = \emptyset$ and $\mathcal{N}_t \cap \mathcal{N}_s = \emptyset$. Also, $\mathcal{N}_t \subseteq \mathcal{N}_l$: our encoding will map each transaction into a process residing at a location with the same name.

Finally, we shall use $x, y, w, x', y', w', \ldots$ to denote elements of the three sets when there is no need to distinguish them. For adaptable processes, we shall use $X, Y, Z, \ldots$ to denote *process variables*.

3.1. Compensable processes

Syntax. We introduce the calculus of *compensable processes* (with discarding semantics). It considers prefixes π and processes $P, Q, \ldots$ defined as:

$$\pi ::= a \mid \overline{x}$$
$$P, Q ::= \mathbf{0} \mid \pi.P \mid !\pi.P \mid (\nu x)P \mid P \mid Q \mid t[P, Q] \mid \langle Q\rangle$$

Prefixes π include input actions (a), output actions ($\overline{a}$) and error notifications ($\overline{t}$). Processes for inaction ($\mathbf{0}$), action prefix ($\pi.P$), guarded replication ($!\pi.P$), restriction ($(\nu x)P$) and parallel composition ($P \mid Q$) are standard. Protected blocks $\langle Q\rangle$ and transactions $t[P, Q]$ have already been motivated. Name x is bound in $(\nu x)P$. In our encodability results, we shall write $\mathcal{C}$ to denote the calculus of compensable processes.

Operational semantics. Following [15,16], the semantics of compensable processes is given in terms of a Labeled Transition System (LTS). Ranged over by α, α', the set of labels includes a, $\overline{a}$, $\overline{t}$ and τ. As in CCS, a denotes an input action, $\overline{a}$ denotes an output action, $\overline{t}$ denotes an error notification and τ denotes synchronization (internal action). As explained in § 2, this LTS is parametric in an extraction function, which is defined in Fig. 1 and realizes the intended discarding semantics.

Error notifications can be *internal* or *external* to the transaction: if the error notification is generated from the default activity then we call it internal; otherwise, the error notification is external. Fig. 2 gives the rules of the LTS; we comment briefly on each of them:

- Axioms (L-In) and (L-Out) execute input and output prefixes, respectively.
- Rule (L-Rep) deals with guarded replication.
- Rule (L-Par1) allows one parallel component to progress independently.
- Rule (L-Res) is the standard rule for restriction. A transition of process P determines a transition of process $(\nu x)P$, where the side condition provides that the restricted name x does not occur inside α.
- Rule (L-Comm1) defines communication on x.
- Rule (L-Block) specifies that protected blocks are transparent units of behavior.
- Rule (L-Rec-Out) allows an external process to abort a transaction via an output action $\overline{t}$. The resulting process contains two parts: the first is obtained from the default activity of the transaction via the extraction function (cf. Fig. 1); the second corresponds to the compensation activity, executed in a protected block.
- Rule (L-Scope-Out) allows the default activity of a transaction to progress.
- Rule (L-Rec-In) handles failure when the error notification is internal to the transaction.

It is convenient to define structural congruence ($\equiv$) and evaluation contexts also for compensable processes.

Definition 3.2 *(Structural congruence).* Structural congruence is the smallest congruence relation on processes that is generated by the following rules:

$$\text{(L-In)}\ a.P \xrightarrow{a} P \qquad \text{(L-Out)}\ \overline{x}.P \xrightarrow{\overline{x}} P \qquad \text{(L-Rep)}\ \frac{\pi.P \xrightarrow{\alpha} P'}{!\pi.P \xrightarrow{\alpha} P' \mid !\pi.P} \qquad \text{(L-Par1)}\ \frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \qquad \text{(L-Res)}\ \frac{P \xrightarrow{\alpha} P' \quad \alpha \notin \{x,\overline{x}\}}{(\nu x)P \xrightarrow{\alpha} (\nu x)P'} \qquad \text{(L-Comm1)}\ \frac{P \xrightarrow{x} P' \quad Q \xrightarrow{\overline{x}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\text{(L-Block)}\ \frac{P \xrightarrow{\alpha} P'}{\langle P\rangle \xrightarrow{\alpha} \langle P'\rangle} \qquad \text{(L-Rec-Out)}\ \frac{}{t[P,Q] \xrightarrow{t} \mathsf{extr}(P) \mid \langle Q\rangle} \qquad \text{(L-Scope-Out)}\ \frac{P \xrightarrow{\alpha} P' \quad \alpha \notin \{t,\overline{t}\}}{t[P,Q] \xrightarrow{\alpha} t[P',Q]} \qquad \text{(L-Rec-In)}\ \frac{P \xrightarrow{\overline{t}} P'}{t[P,Q] \xrightarrow{\tau} \mathsf{extr}(P') \mid \langle Q\rangle}$$

Fig. 2. LTS for compensable processes. The symmetric counterparts of (L-Par1) and (L-Comm1) have been omitted.

$$\begin{array}{ll}
P \mid Q \equiv Q \mid P & (\nu x)\mathbf{0} \equiv \mathbf{0} \\
P \mid (Q \mid R) \equiv (P \mid Q) \mid R & (\nu x)(\nu y)P \equiv (\nu y)(\nu x)P \\
P \mid \mathbf{0} \equiv P & Q \mid (\nu x)P \equiv (\nu x)(P \mid Q) \text{ if } x \notin \mathtt{fn}(Q) \\
!\pi.P \equiv \pi.P \mid !\pi.P & t[(\nu x)P,Q] \equiv (\nu x)t[P,Q] \text{ if } t \neq x,\ x \notin \mathtt{fn}(Q) \\
P \equiv Q \text{ if } P \equiv_\alpha Q & \langle(\nu x)P\rangle \equiv (\nu x)\langle P\rangle
\end{array}$$

The first column in Definition 3.2 contains standard rules: commutativity, associativity, and neutral element for parallel composition. We rely on usual notions of α-conversion (noted $\equiv_\alpha$). The second column contains garbage collection of useless restrictions, swapping of restrictions, and scope extrusion for parallel composition, transaction scope and protected blocks.

Definition 3.3 *(Evaluation contexts).* The syntax of contexts in compensable processes is given by the following grammar:

$$C[\bullet] ::= [\bullet] \ \mid\ \langle C[\bullet]\rangle \ \mid\ t[C[\bullet],P] \ \mid\ C[\bullet] \mid P \ \mid\ (\nu x)C[\bullet],$$

where P is a compensable process.

We write $C[Q]$ to denote the process obtained by replacing the hole $[\bullet]$ in context $C[\bullet]$ with Q.

The following proposition is key to our operational correspondence statements.

Proposition 3.1. *Let P be a compensable process. If $P \xrightarrow{\tau} P'$ then one of the following holds:*

(a) $P \equiv E[C[\overline{a}.P_1] \mid D[a.P_2]]$ *and* $P' \equiv E[C[P_1] \mid D[P_2]]$,
(b) $P \equiv E[C[t[P_1,Q]] \mid D[\overline{t}.P_2]]$ *and* $P' \equiv E[C[\mathsf{extr}(P_1) \mid \langle Q\rangle] \mid D[P_2]]$,
(c) $P \equiv C[t[D[\overline{t}.P_1],Q]]$ *and* $P' \equiv C[\mathsf{extr}(D[P_1]) \mid \langle Q\rangle]$,

for some contexts C, D, E, processes P_1, P_2, Q and names a, t.

Proof. See §A.1 at page 37. □

Remark 3.2 *(Reductions).* It is convenient to define a reduction semantics for compensable processes. We do so by exploiting the LTS just introduced: we shall write $P \longrightarrow P'$ whenever $P \xrightarrow{\tau} P''$ and $P'' \equiv P'$, for some P''. As customary, we write $\longrightarrow^*$ to denote the reflexive and transitive closure of $\longrightarrow$.

3.2. Well-formed compensable processes

We shall focus on *well-formed* compensable processes: a class of processes that disallows certain non-deterministic interactions that involve nested transaction and error notification names. Concise examples of processes that are not well-formed are the following:

$$P = t_1\big[a \mid t_2[b,\overline{b}],\overline{a}\big] \mid \overline{t_1} \mid \overline{t_2} \ \times \qquad P_1 = t_1[a,b] \mid t_2[\overline{t_1},d] \mid \overline{t_2} \ \times \qquad P_2 = t_1[\overline{t_2},a] \mid t_2[\overline{t_1},b] \ \times \tag{5}$$

Processes P, P_1 and P_2 feature concurrent error notifications (on t_1 and t_2), which induce a form of non-determinism that is hard to capture properly in the (lower level) representation that we shall give in terms of adaptable processes. Indeed, P features an *interference* between the failure of t_1 and t_2; it is hard to imagine patterns where this kind of interfering concurrency may come in handy. From the same reason, we will assume that all transaction names in a well-formed process are different. In contrast, we would like to consider as well-formed the following processes (where $t_1 \neq t_2$):

$$P' = t_1\big[a \mid t_2[b,\bar{b}],\bar{a}\big] \mid \overline{t_2}.\overline{t_1} \ \checkmark \qquad P'' = t_1[a,\bar{a}] \mid t_2[b,\bar{b}] \mid \overline{t_1} \mid \overline{t_2} \ \checkmark \tag{6}$$

In what follows, we formally introduce well-formed compensable processes. We require some notations: (a) sets of pairs $\Gamma, \Delta \subseteq \mathcal{N}_t \times \mathcal{N}_t$; (b) sets $\gamma, \delta \in \mathcal{N}_t$; and (c) boolean $p \in \{\top, \bot\}$. These elements have the following reading:

- Γ is the set of (potential) pairs of parallel failure signals in P;
- Δ is the set of (potential) pairs of nested transaction names in P (with form (parent, child));
- γ is the set of failure signals in P;
- δ is the set of top-level transactions in P;
- p is true if and only if P contains protected blocks.

This way, the well-formedness predicate, denoted $\Gamma; \Delta \vdash_{\gamma;\delta;p} P$, is inductively defined in Fig. 3. We write $\mathcal{P}(P)$ to denote the parameters Γ, Δ, γ, and δ associated to P, i.e., $\mathcal{P}(P) = (\Gamma, \Delta, \gamma, \delta)$.

We briefly comment on the rules in Fig. 3:

- Rule (W-NIL) states that the inactive process has neither parallel failure signal nor nested transactions; it also does not contain protected blocks.
- Rules (W-OUT1), (W-OUT2), and (W-IN) enforce that protected blocks or transactions do not appear behind prefixes (i.e., $p = \bot$, $\delta = \emptyset$). Rule (W-OUT1) says that if the name of the prefix is the failure signal then it will be collected by γ. Rule (W-OUT2) says that if the name of the prefix is not the failure signal then the set of the failure signals will be as in the process that appears after the prefix. For example, by (W-NIL) and two successive applications of (W-OUT1), we can infer $\emptyset; \emptyset \vdash_{\{t_1,t_2\};\emptyset;\bot} \overline{t_2}.\overline{t_1}$.
- Rule (W-RES) says that if P satisfies the predicate for some parameters, then $(\nu x)P$ satisfies the predicate with the same parameters.
- Rule (W-BLOCK) specifies that if P satisfies the predicate for some parameters, then $\langle P \rangle$ satisfies the predicate with the same Γ, Δ, γ and δ. The fifth parameter for $\langle P \rangle$ specifies that it contains protected blocks ($p = \top$ in the conclusion). This way, for example, we have $\emptyset; \emptyset \vdash_{\{t_1,t_2\};\emptyset;\top} \langle \overline{t_2}.\overline{t_1} \rangle$.

Rules (W-REP), (W-TRANS), and (W-PAR) rely on the following auxiliary notations. First, given sets γ_1, γ_2, δ and a name t, we introduce the following sets:

$$\gamma_1 \times \gamma_2 = \{(t', t'') : t' \in \gamma_1 \wedge t'' \in \gamma_2\} \qquad \{t\} \times \delta = \{(t, t') : t' \in \delta\}. \tag{7}$$

Also, we write Γ^s and Δ^t to denote the symmetric closure of Γ and the transitive closure of Δ, respectively. We will use, respectively the following functions f_t and f for conditions in Rules (W-TRANS) and (W-PAR):

$$f_t(\mathcal{P}(P), \mathcal{P}(Q)) = (\Gamma_1 \cup \Gamma_2 \cup (\gamma_1 \times \gamma_2), \Delta_1 \cup \Delta_2 \cup (\{t\} \times (\delta_1 \cup \delta_2 \cup \gamma_1 \cup \gamma_2))) \tag{8}$$

$$f(\mathcal{P}(P), \mathcal{P}(Q)) = (\Gamma_1 \cup \Gamma_2 \cup (\gamma_1 \times \gamma_2), \Delta_1 \cup \Delta_2) \tag{9}$$

where $\mathcal{P}(P) = (\Gamma_1, \Delta_1, \gamma_1, \delta_1)$ and $\mathcal{P}(Q) = (\Gamma_2, \Delta_2, \gamma_2, \delta_2)$.
We may now discuss Rules (W-REP), (W-TRANS), and (W-PAR):

- Rule (W-REP) says that the set of pairs of parallel failure signals in $!\pi.P$ is $\gamma \times \gamma$, where γ is the set of failure signals in $\pi.P$. This is directly related to the transition rule (L-REP) in Fig. 2. All other parameters of the predicate satisfied by $!\pi.P$ are the same as for $\pi.P$.
For example, we can derive $\{t_1, t_2\} \times \{t_1, t_2\}; \emptyset \vdash_{\{t_1,t_2\};\emptyset;\bot} \ !\overline{t_2}.\overline{t_1}$.
- Rule (W-TRANS) specifies the well-formed conditions for $t[P, Q]$. First, $\delta = \{t\}$. The set of pairs of parallel failure signals is the union of the respective sets for P and Q and the set whose elements are pairs of failure signals; in the pair, one element belongs to the set of failure signals of P and the second element is from the set of failure signals of Q. This extension with $\gamma_1 \times \gamma_2$ is necessary for $t[P, Q]$, because P may contain protected blocks which will be composed in parallel with $\langle Q \rangle$ in case of an error. The set of pairs of nested transactions is obtained from those for P and Q, also considering further pairs as specified by $\{t\} \times (\delta_1 \cup \delta_2 \cup \gamma_1 \cup \gamma_2)$ (cf. (7)). The rule also enforces that the sets of parallel failure signals and nested transaction names in the parallel composition are disjoint (i.e., $(\Gamma_1 \cup \Gamma_2 \cup (\gamma_1 \times \gamma_2))^s \cap (\Delta_1 \cup \Delta_2 \cup (\{t\} \times (\delta_1 \cup \delta_2 \cup \gamma_1 \cup \gamma_2)))^t = \emptyset$). For example, we can derive $\emptyset; \{(t_1, t_2)\} \vdash_{\emptyset;\{t_1\};\bot} t_1[a \mid t_2[b,\bar{b}],\bar{a}]$.
- Rule (W-PAR) specifies the cases in which $P \mid Q$ satisfies the predicate provided that P and Q individually satisfy it. The set of pairs of parallel failure signals is obtained as in Rule (W-TRANS). The set of pairs of nested transactions is obtained as the union of sets of pairs of nested transactions for P and Q. Also, it must hold that $(\Gamma_1 \cup \Gamma_2 \cup (\gamma_1 \times \gamma_2))^s \cap (\Delta_1 \cup \Delta_2)^t = \emptyset$. For example, for P' and P'' in (6) we have

$$\emptyset; \{(t_1, t_2)\} \vdash_{\{t_1,t_2\};\{t_1\};\bot} t_1\big[a \mid t_2[b,\bar{b}],\bar{a}\big] \mid \overline{t_2}.\overline{t_1} \text{ and } \{(t_1, t_2)\}; \emptyset \vdash_{\{t_1,t_2\};\{t_1,t_2\};\bot} t_1\big[a,\bar{a}\big] \mid t_2[b,\bar{b}] \mid \overline{t_1} \mid \overline{t_2}.$$

$$\text{(W-NIL)}\quad \emptyset;\emptyset \vdash_{\emptyset;\emptyset;\bot} \mathbf{0} \qquad \text{(W-OUT1)}\ \frac{\Gamma;\emptyset \vdash_{\gamma;\emptyset;\bot} P}{\Gamma;\emptyset \vdash_{\gamma\cup\{t\};\emptyset;\bot} \bar{t}.P} \qquad \text{(W-OUT2)}\ \frac{\Gamma;\emptyset \vdash_{\gamma;\emptyset;\bot} P}{\Gamma;\emptyset \vdash_{\gamma;\emptyset;\bot} \bar{a}.P} \qquad \text{(W-IN)}\ \frac{\Gamma;\emptyset \vdash_{\gamma;\emptyset;\bot} P}{\Gamma;\emptyset \vdash_{\gamma;\emptyset;\bot} a.P}$$

$$\text{(W-RES)}\ \frac{\Gamma;\Delta \vdash_{\gamma;\delta;p} P}{\Gamma,\Delta \vdash_{\gamma;\delta;p} (\nu x)P} \qquad \text{(W-BLOCK)}\ \frac{\Gamma;\Delta \vdash_{\gamma;\delta;p} P}{\Gamma;\Delta \vdash_{\gamma;\delta;\top} \langle P\rangle} \qquad \text{(W-REP)}\ \frac{\Gamma;\emptyset \vdash_{\gamma;\emptyset;\bot} \pi.P}{\gamma\times\gamma;\emptyset \vdash_{\gamma;\emptyset;\bot} !\pi.P}$$

$$\text{(W-TRANS)}\ \frac{\Gamma_1;\Delta_1 \vdash_{\gamma_1;\delta_1;p_1} P \quad \Gamma_2;\Delta_2 \vdash_{\gamma_2;\delta_2;p_2} Q \quad f_t(\mathcal{P}(P),\mathcal{P}(Q)) = (\Gamma,\Delta) \quad \Gamma^s\cap\Delta^t=\emptyset}{\Gamma,\Delta \vdash_{\gamma_1\cup\gamma_2;\{t\};p_1\vee p_2} t[P,Q]}$$

$$\text{(W-PAR)}\ \frac{\Gamma_1;\Delta_1 \vdash_{\gamma_1;\delta_1;p_1} P \qquad \Gamma_2;\Delta_2 \vdash_{\gamma_2;\delta_2;p_2} Q \quad f(\mathcal{P}(P),\mathcal{P}(Q)) = (\Gamma,\Delta) \quad \Gamma^s\cap\Delta^t=\emptyset}{\Gamma,\Delta \vdash_{\gamma_1\cup\gamma_2;\delta_1\cup\delta_2;p_1\vee p_2} P\mid Q}$$

Fig. 3. Auxiliary relation for well-formed compensable processes.

One should notice that processes from (5) do not satisfy the predicate, since their sets of pairs of parallel failure signals and nested transaction names are not disjoint: they are both equal to $\{(t_1,t_2)\}$.

We then have the following definition:

Definition 3.4 *(Well-formedness).* A compensable process P is *well-formed* if

(i) transaction names in P are mutually different, and
(ii) $\Gamma;\Delta \vdash_{\gamma;\delta;p} P$ holds for some $\Gamma,\Delta,\gamma,\delta,p$.

The following theorem captures the main properties of well-formed processes: they do not contain subterms with protected blocks or transactions behind prefixes; also, they do not contain potential parallel failure signals for nested transaction names. Since the former is required to hold also for compensations within transactions, we extend evaluation contexts (Definition 3.3) as follows:

$$C^{wf}[\bullet] ::= [\bullet] \mid \langle C^{wf}[\bullet]\rangle \mid t[C^{wf}[\bullet],P] \mid C^{wf}[\bullet]\mid P \mid (\nu x)C^{wf}[\bullet] \mid t[P,C^{wf}[\bullet]]. \tag{10}$$

Theorem 3.3. *Let $\Gamma;\Delta \vdash_{\gamma;\delta;p} P$, for some $\Gamma,\Delta,\gamma,\delta$ and p. Then the following holds:*

(i) *if $P\equiv C^{wf}[\pi.P_1]$ then $\Gamma';\emptyset \vdash_{\gamma';\emptyset;\bot} P_1$, for some Γ' and γ', and*
(ii) $\Gamma^s\cap\Delta^t=\emptyset$.

Proof. (i) By induction on the structure of $C^{wf}[\bullet]$. (ii) By case analysis. □

We may now state a soundness result, which ensures that well-formedness is preserved by transitions.

Theorem 3.4. *If $\Gamma;\Delta \vdash_{\gamma;\delta;p} P$ and $P \xrightarrow{\alpha} P'$ then there are $\Gamma'\subseteq\Gamma$ and $\Delta'\subseteq\Delta$ such that $\Gamma';\Delta' \vdash_{\gamma';\delta';p'} P'$.*

Proof. By induction on the depth of the derivation $P \xrightarrow{\alpha} P'$. See § A.2 at page 39. □

The following is immediate from Definition 3.4 and Theorem 3.4:

Corollary 3.5. *If P is a well-formed compensable process and $P \longrightarrow^* P'$ then P' is well formed.*

(R-IN-OUT)
$$E\Big[C\big[\overline{x}.P\big] \mid D\big[x.Q\big]\Big] \longrightarrow E\Big[C\big[P\big] \mid D\big[Q\big]\Big]$$

(R-OB-UPD)
$$E\Big[C\big[l[P]\big] \mid D\big[l\{(X).Q\}.R\big]\Big] \longrightarrow E\Big[C\big[Q\{^{P}/_{X}\}\big] \mid D\big[R\big]\Big]$$

(R-SUB-UPD)
$$E\Big[C\big[l[P]\big] \mid D\big[l\langle\!\langle (X).Q\rangle\!\rangle.R\big]\Big] \longrightarrow E\Big[C\big[\mathbf{0}\big] \mid D\big[Q\{^{P}/_{X}\} \mid R\big]\Big]$$

(R-STR)
$$\frac{P \equiv P' \quad P' \longrightarrow Q' \quad Q' \equiv Q}{P \longrightarrow Q}$$

Fig. 4. Reduction semantics for adaptable processes..

3.3. Adaptable processes

Syntax. We consider *prefixes* π and *processes* $P, Q, \ldots$ defined as:

$$\begin{array}{lcl} \pi & ::= & x \;\mid \overline{x} \mid\; l\langle\!\langle (X).Q\rangle\!\rangle \;\mid\; l\{(X).Q\} \\ P, Q & ::= & \mathbf{0} \mid \pi.P \mid\; !\pi.P \mid (\nu x)P \mid P \mid Q \mid l[P] \mid X \end{array}$$

We consider input and output prefixes (denoted x and $\overline{x}$, respectively) as well as the update prefixes $l\langle\!\langle (X).Q\rangle\!\rangle$ and $l\{(X).Q\}$ for subjective and objective update, respectively. We assume that Q may contain zero or more occurrences of the process variable X.

Although here we consider a process model with both update prefixes, we shall consider target calculi with only one of them: the calculus of adaptable processes with subjective and objective update will be denoted $\mathcal{S}$ and $\mathcal{O}$, respectively.

The syntax includes constructs for inaction ($\mathbf{0}$); action prefix ($\pi.P$); guarded replication ($!\pi.P$), i.e. infinitely many occurrences of P in parallel, which are triggered by prefix π; restriction ($(\nu x)P$); parallel composition ($P \mid Q$); located processes ($l[P]$); and process variables (X). We omit $\mathbf{0}$ whenever possible; we write, e.g., $l\langle\!\langle (X).P\rangle\!\rangle$ instead of $l\langle\!\langle (X).P\rangle\!\rangle.\mathbf{0}$.

Name x is bound in $(\nu x)P$ and process variable X is bound in $l\langle\!\langle (X).Q\rangle\!\rangle$. Given this, the sets of free and bound names for a process P—denoted $\mathtt{fn}(P)$ and $\mathtt{bn}(P)$—are as expected (and similarly for process variables). We rely on expected notions of α-conversion (noted $\equiv_\alpha$) and process substitution: we denote by $P\{^{Q}/_{X}\}$ the process obtained by (capture-avoiding) substitution of Q for X in P.

Operational semantics. Adaptable processes are governed by a reduction semantics, denoted $P \longrightarrow P'$, a relation on processes that relies on *structural congruence* (denoted $\equiv$) and *contexts* (denoted C, D, E).

Definition 3.5 *(Structural congruence).* Structural congruence is the smallest congruence relation on processes that is generated by the following rules, which extend standard rules for the π-calculus with scope extrusion for locations:

$$\begin{array}{lll} P \mid Q \equiv Q \mid P & (\nu x)\mathbf{0} \equiv \mathbf{0} & (\nu x)l[P] \equiv l[(\nu x)P] \text{ if } l \neq x \\ P \mid (Q \mid R) \equiv (P \mid Q) \mid R & (\nu x)(\nu y)P \equiv (\nu y)(\nu x)P & !\pi.P \equiv \pi.P \mid\, !\pi.P \\ P \mid \mathbf{0} \equiv P & Q \mid (\nu x)P \equiv (\nu x)(Q \mid P) \text{ if } x \notin \mathtt{fn}(Q) & P \equiv Q \text{ if } P \equiv_\alpha Q \end{array}$$

Contexts are processes with a *hole* $[\bullet]$; their syntax is defined as follows:

Definition 3.6 *(Evaluation contexts).* The syntax of contexts is given by the following grammar:

$$C[\bullet] ::= [\bullet] \mid l\big[C[\bullet]\big] \mid C[\bullet] \mid P \mid (\nu x)C[\bullet].$$

We write $C[Q]$ to denote the process resulting from filling in the hole $[\bullet]$ in context C with process Q.

Reduction $\longrightarrow$ is the smallest relation on processes induced by the rules in Fig. 4, which we now briefly discuss:

- Rule (R-IN-OUT) formalizes synchronization between processes $\overline{x}.P$ and $x.Q$, enclosed in contexts C and D, respectively.
- Rules (R-SUB-UPD) and (R-OB-UPD) formalize the equations (1) and (2) given in the Introduction. They implement subjective and objective update of a process located at location l that resides in contexts C and E. In general, we shall use one of these two rules, not both.
- Rule (R-STR) is self-explanatory.

We write $\longrightarrow^*$ to denote the reflexive and transitive closure of $\longrightarrow$.

4. The notion of encoding

Our objective is to relate compensable and adaptable processes through *valid encodings* (simply *encodings* in the following). Here we define a basic abstract framework that will help us formalize these relations.

An encoding is a translation of processes of a *source language* into the processes of a *target language*; this translation should satisfy certain *correctness criteria*, which attest to its quality. The existence of an encoding shows that the target language is at least as expressive as the source language.

To define valid encodings, we adopt five correctness criteria formulated by Gorla [12]: (1) *compositionality* and (2) *name invariance* (so-called *structural* criteria) as well as (3) *operational correspondence*, (4) *divergence reflection*, and (5) *success sensitiveness* (so-called *semantic* criteria). Structural criteria describe the static structure of the encoding, whereas the semantic criteria describe its dynamics—how the behavior of encoded terms relates to that of source terms, and vice versa. As stated in [20], structural criteria are needed in order to measure the expressiveness of operators in contrast to expressiveness of terms. As for semantic criteria, operational correspondence is divided in *completeness* and *soundness* properties: the former ensures that the behavior of a source process is preserved by the translation in the target calculus; the latter ensures that the behavior of a translated (target) process corresponds to that of some source process. Divergence reflection ensures that a translation does not introduce spurious infinite computations, whereas success sensitiveness requires that source and translated terms behave in the same way with respect to some notion of *success*.

Following [12], we start by defining an abstract notion of calculus, which we will later instantiate with the three calculi of interest here:

Definition 4.1 *(Calculus).* We define a *calculus* as a triple $(\mathcal{P}, \longrightarrow, \approx)$, where:

- $\mathcal{P}$ is a set of processes;
- $\longrightarrow$ is its associated reduction semantics, which specifies how a process computes on its own;
- $\approx$ is an equality on processes, useful to describe the abstract behavior of a process, which is a congruence at least with respect to parallel composition.

We will further assume that a calculus uses a countably infinite set of names, usually denoted $\mathcal{N}$. Accordingly, the abstract definition of encoding refers to those names.

Definition 4.2 *(Encoding).* Let $\mathcal{N}_{\mathbf{s}}$ and $\mathcal{N}_{\mathbf{t}}$ be countably infinite sets of source and target names, respectively. An *encoding* of the source calculus $(\mathcal{P}_{\mathbf{s}}, \longrightarrow_{\mathbf{s}}, \approx_{\mathbf{s}})$ into the target calculus $(\mathcal{P}_{\mathbf{t}}, \longrightarrow_{\mathbf{t}}, \approx_{\mathbf{t}})$ is a tuple $(\llbracket\cdot\rrbracket, \varphi_{\llbracket\cdot\rrbracket})$ where $\llbracket\cdot\rrbracket : \mathcal{P}_{\mathbf{s}} \longrightarrow \mathcal{P}_{\mathbf{t}}$ denotes a *translation* that satisfies some specific correctness criteria and $\varphi_{\llbracket\cdot\rrbracket} : \mathcal{N}_{\mathbf{s}} \longrightarrow \mathcal{N}_{\mathbf{t}}$ denotes a *renaming policy* for $\llbracket\cdot\rrbracket$.

The renaming policy defines the way names from the source language are translated into the target language. A valid encoding cannot depend on the particular names involved in source processes.

We shall use the following notations. We write $\longrightarrow^*$ to denote the reflexive, transitive closure of $\longrightarrow$. Also, given $k \geq 1$, we will write $P \longrightarrow^k P'$ to denote k consecutive reduction steps leading from P to P'. That is, $P_1 \longrightarrow^k P_{k+1}$ holds whenever there exist $P_2, \ldots, P_k$ such that $P_1 \longrightarrow P_2 \longrightarrow \cdots \longrightarrow P_k \longrightarrow P_{k+1}$.

For compositionality, we use a context to combine the translated subterms, which depends on the source operator that combines the subterms. This context is parametrized on a finite set of names, noted $\mathtt{N}$ below, which contains the set of free names of the respective source term. In a slight departure from usual definitions of compositionality, the set $\mathtt{N}$ may contain transaction names that do not occur free in the term. As we will see, we have an initially empty parameter on the encoding function that is accumulated while translating a source term.

For operational correspondence our encodings follow more strict criteria than in [12]. For divergence reflection we will use the following definition:

Definition 4.3 *(Divergence).* A process P diverges, written $P \longrightarrow^{\omega}$, if there exists an infinite sequence of processes $\{P_i\}_{i \geq 0}$ such that $P = P_0$ and for any i, $P_i \longrightarrow P_{i+1}$.

To formulate success sensitiveness, we assume that both source and target calculi contain the same success process $\checkmark$; also, we assume that $\Downarrow$ is a predicate that asserts reducibility (in a "may" modality) to a process containing an unguarded occurrence of $\checkmark$. This process operator does not affect the operational semantics and behavioral equivalence of the calculi: $\checkmark$ can not reduce and $\mathtt{n}(\checkmark) = \mathtt{fn}(\checkmark) = \mathtt{bn}(\checkmark) = \emptyset$. Therefore, this language extension does not affect the validity of the encodability criteria, except for success sensitiveness.

Definition 4.4 *(Success).* Let $(\mathcal{P}, \longrightarrow, \approx)$ be a calculus. A process $P \in \mathcal{P}$ (may)-succeeds, denoted $P \Downarrow$, if it is reducible to a process containing an unguarded occurrence of $\checkmark$, i.e., if $P \longrightarrow^* P'$ and $P' = C[\checkmark]$ for some P' and context $C[\bullet]$.

The following definition formalizes the five criteria for valid encodings:

Definition 4.5 *(Valid encoding).* Let $\mathcal{L}_{\mathbf{s}} = (\mathcal{P}_{\mathbf{s}}, \longrightarrow_{\mathbf{s}}, \approx_{\mathbf{s}})$ and $\mathcal{L}_{\mathbf{t}} = (\mathcal{P}_{\mathbf{t}}, \longrightarrow_{\mathbf{t}}, \approx_{\mathbf{t}})$ be source and target calculi, respectively, each with countably infinite sets of names $\mathcal{N}_{\mathbf{s}}$ and $\mathcal{N}_{\mathbf{t}}$. An encoding $(\llbracket\cdot\rrbracket, \varphi_{\llbracket\cdot\rrbracket})$, where $\llbracket\cdot\rrbracket : \mathcal{P}_{\mathbf{s}} \longrightarrow \mathcal{P}_{\mathbf{t}}$ and $\varphi_{\llbracket\cdot\rrbracket} : \mathcal{N}_{\mathbf{s}} \longrightarrow \mathcal{N}_{\mathbf{t}}$, is a *valid encoding* if it satisfies the following criteria:

(1) **Compositionality**: $\llbracket\cdot\rrbracket$ is *compositional* if for every n-ary ($n \geq 1$) operator $\mathtt{op}$ on $\mathcal{P}_s$ and for every set of names $\mathtt{N}$ there is an n-adic context $C^{\mathtt{N}}_{\mathtt{op}}[\bullet_1, \dots, \bullet_n]$ such that, for all $P_1, \dots, P_n$ with $\mathtt{fn}(P_1, \dots, P_n) \subseteq \mathtt{N}$ it holds that $\llbracket \mathtt{op}(P_1, \dots, P_n) \rrbracket = C^{\mathtt{N}}_{\mathtt{op}}[\llbracket P_1 \rrbracket, \dots, \llbracket P_n \rrbracket]$.
(2) **Name invariance**: $\llbracket\cdot\rrbracket$ is *name invariant* if for every substitution $\sigma : \mathcal{N}_{\mathbf{s}} \longrightarrow \mathcal{N}_{\mathbf{s}}$ there is a substitution $\sigma' : \mathcal{N}_{\mathbf{t}} \longrightarrow \mathcal{N}_{\mathbf{t}}$ such that (i) for every $a \in \mathcal{N}_{\mathbf{s}} : \varphi_{\llbracket\cdot\rrbracket}(\sigma(a)) = \sigma'(\varphi_{\llbracket\cdot\rrbracket}(a))$ and (ii) $\llbracket \sigma(P) \rrbracket = \sigma'(\llbracket P \rrbracket)$.
(3) **Operational correspondence**: $\llbracket\cdot\rrbracket$ is *operational corresponding* if it satisfies the two requirements:
  a) **Completeness**: If $P \longrightarrow_{\mathbf{s}} Q$ then there exists k such that $\llbracket P \rrbracket \longrightarrow^{k}_{\mathbf{t}} \approx_{\mathbf{t}} \llbracket Q \rrbracket$.
  b) **Soundness**: If $\llbracket P \rrbracket \longrightarrow^{*}_{\mathbf{t}} R$ then there exists P' such that $P \longrightarrow^{*}_{\mathbf{s}} P'$ and $R \longrightarrow^{*}_{\mathbf{t}} \approx_{\mathbf{t}} \llbracket P' \rrbracket$.
(4) **Divergence reflection**: $\llbracket\cdot\rrbracket$ *reflects divergence* if, for every P such that $\llbracket P \rrbracket \longrightarrow^{\omega}_{\mathbf{t}}$, it holds that $P \longrightarrow^{\omega}_{\mathbf{s}}$.
(5) **Success sensitiveness**: $\llbracket\cdot\rrbracket$ is *success sensitive* if, for every $P \in \mathcal{P}_{\mathbf{s}}$, it holds that $P \Downarrow$ if and only if $\llbracket P \rrbracket \Downarrow$.

Concrete instances. We now instantiate Definition 4.1 with the source and target calculi of interest:

Source calculus: $\mathcal{C}^{\lambda}$ The source calculus will be the calculus of compensable processes with discarding semantics defined in § 3.1. The set of processes, which we will denote $\mathcal{C}$, will contain only *well-formed* compensable processes (cf. § 3.2). We shall consider the reduction relation $\longrightarrow$ defined at the end of § 3.1. We shall use structural congruence (Definition 3.2) as behavioral equivalence.

Target calculi: $\mathcal{S}$ **and** $\mathcal{O}$ There will be two target calculi, both based on the calculus of adaptable processes defined in § 3.3. The first one, with set of processes denoted $\mathcal{S}$, uses subjective updates only; its reduction semantics is as given in Fig. 4, with updates governed by Rule (R-Sub-Upd). Similarly, the second calculus, with set of processes denoted $\mathcal{O}$, uses objective updates only; its reduction semantics is governed by Rule (R-Ob-Upd) instead. In both cases, the structural congruence of Definition 3.5 will be used as behavioral equivalence.

As already mentioned, in § 8 we shall consider three variants of the source calculus $\mathcal{C}$ (cf. Definition 8.1).

The purpose of $\approx_{\mathbf{t}}$ in the definition of operational correspondence is to abstract away from "junk" processes, i.e., processes left behind as a result of the translation that do not add any meaningful source behavior to translated processes. As we will see, our translations do not pollute: the inactive process $\mathbf{0}$ will be the only possible junk process. As such, it is trivially *inactive junk* in the sense that it does not perform further reductions on its own nor interact with the surrounding target terms. This is why it suffices to use structural congruences on source and target processes as behavioral equalities.

We now move on to present our encodings of $\mathcal{C}$ into $\mathcal{S}$ and $\mathcal{O}$. To compare these two encodings, we shall define the abstract notion of *efficient encoding*—see Definition 7.1.

5. Encoding $\mathcal{C}$ into $\mathcal{S}$: the case of subjective update

We shall now present our first encoding, which translates the calculus of compensable processes ($\mathcal{C}$, our source calculus) into the calculus of adaptable processes with subjective update ($\mathcal{S}$, our target calculus). We shall prove that this translation is valid, in the sense of Definition 4.5. Before giving a formal presentation of the encoding, we introduce some useful conventions and intuitions.

5.1. Preliminaries

Recall the base sets defined in Definition 3.1; in particular, $\mathcal{N}_t$ denotes the base set of transaction names. Our encodings rely on the following notion of *path*, a sequence of transaction names:

Definition 5.1 *(Paths).* Let $\mathcal{N}_t^k$ (with $k \in \mathbb{N}$) be the set of sequences/tuples of names in $\mathcal{N}_t$. These sequences will be denoted by $\mu, \mu', \dots$; we assume they have pairwise distinct elements. We obtain *paths* from the concatenation of such sequences with ε (the empty path) at the end; paths are denoted by $\rho, \rho', \dots$ (i.e., $\rho = \mu\varepsilon$, $\rho' = \mu'\varepsilon, \dots$). We will sometimes omit writing the tail ε in ρ. By a slight abuse of notation, given a transaction name t and a path ρ, we will write $t \in \rho$ if t occurs in ρ.

We also require sets of *reserved names*. We have the following definition:

Definition 5.2 *(Reserved names).* The sets of *reserved names* $\mathcal{N}_s^r$ and $\mathcal{N}_l^r$ are defined as follows:

- $\mathcal{N}_s^r = \{h_x \mid x \in \mathcal{N}_t\}$ is the set of *reserved synchronization names*, and
- $\mathcal{N}_l^r = \{p_\rho \mid \rho \text{ is a path}\}$ is the set of *reserved location names*.

If $t_1, t_2 \in \mathcal{N}_t$ such that $t_1 \neq t_2$ then $h_{t_1} \neq h_{t_2}$ and $p_{t_1} \neq p_{t_2}$. We let $\mathcal{N}_l^r \subseteq \mathcal{N}_l \setminus \mathcal{N}_t$ and $\mathcal{N}_s \cap \mathcal{N}_s^r = \emptyset$. In what follows we shall use the set $\mathcal{N}_a = \mathcal{N}_l \cup (\mathcal{N}_s \cup \mathcal{N}_s^r)$ for adaptable processes.

We will find it convenient to adopt the following abbreviations for adaptable processes.

Convention 5.1. Recall that $l\langle\!\langle (X).Q \rangle\!\rangle$ and $l\{(X).Q\}$ denote subjective and objective update prefixes, respectively.

- We write $\prod_{i=1}^{n} l[X_i]$ to abbreviate the process $l[X_1] \mid \ldots \mid l[X_n]$.
- We write $t\langle\!\langle \dagger \rangle\!\rangle$ to denote the subjective update prefix $t\langle\!\langle (Y).\mathbf{0} \rangle\!\rangle$, which "kills" both location t and the process it hosts. This way, for instance:

$$s[t[c]] \mid t\langle\!\langle \dagger \rangle\!\rangle \longrightarrow s[\mathbf{0}] \tag{11}$$

  Similarly, we write $t\{\dagger\}$ to stand for the objective update prefix that "kills" t and its content.
- We write $t\langle\!\langle (Y_1, Y_2, \ldots, Y_n).R \rangle\!\rangle$ to abbreviate the *nested update prefix* $t\langle\!\langle (Y_1).t\langle\!\langle (Y_2).\cdots.t\langle\!\langle (Y_n).R \rangle\!\rangle \cdots \rangle\!\rangle \rangle\!\rangle$. For instance:

$$s\Big[t\big[l_1[a] \mid l_1[b] \mid c\big] \mid l_1\langle\!\langle (X_1, X_2).(l_2[X_1] \mid l_2[X_2]) \rangle\!\rangle\Big] \longrightarrow^* s\Big[t\big[c\big] \mid l_2[a] \mid l_2[b]\Big].$$

  Similarly, $t\{(Y_1, Y_2, \ldots, Y_n).R\}$ will stand for the objective prefix $t\{(Y_1).t\{(Y_2).\cdots.t\{(Y_n).R\}\cdots\}\}$.

5.2. The translation, informally

Transactions, protected blocks, and the extraction function that governs compensations (cf. Fig. 1) are the distinguishing constructs in compensable processes; they represent the most interesting process terms to be addressed in our encodings.

We shall use paths (cf. Definition 5.1) to model the hierarchical structure induced by nested transactions. A path can represent and trace the location of the transactions and protected blocks in a process. Our translation of $\mathcal{C}$ into $\mathcal{S}$ will be indexed by a path ρ: it will be denoted $[\![\cdot]\!]_\rho$ (cf. Definition 5.5 below). This way, e.g., the encoding of a protected block found at path ρ will be defined as:

$$[\![\langle P \rangle]\!]_\rho = p_\rho\big[[\![P]\!]_\varepsilon\big]$$

where p_ρ is a reserved name in $\mathcal{N}_l^r$ (cf. Definition 5.2).

A key aspect in our translation is the representation of the extraction function. As we have seen, this function is an external device, embedded in the operational semantics, that formalizes the protection of transactions/protected blocks. Our translation explicitly specifies the extraction function by means of update prefixes. We use the auxiliary process $\mathtt{out}^{\mathtt{s}}(l_1, l_2, n, Q)$, which moves n processes from location l_1 to location l_2, and composes Q in parallel. Using the notations from Convention 5.1, it can be defined as follows:

$$\mathtt{out}^{\mathtt{s}}(l_1, l_2, n, Q) = \begin{cases} Q & \text{if } n = 0 \\ l_1\langle\!\langle (X_1, \ldots, X_n).\big(\prod_{i=1}^{n} l_2[X_i] \mid Q\big) \rangle\!\rangle & \text{if } n > 0 \end{cases} \tag{12}$$

Example 5.2. Consider the process:

$$s\big[t\big[l_1[a] \mid l_1[b] \mid c\big] \mid \mathtt{out}^{\mathtt{s}}(l_1, l_2, 2, Q)\big]$$

We have two reductions:

$$s\big[t\big[l_1[a] \mid l_1[b] \mid c\big] \mid l_1\langle\!\langle (X_1, X_2).(l_2[X_1] \mid l_2[X_2] \mid Q) \rangle\!\rangle\big] \longrightarrow^2 s\big[t\big[c\big] \mid l_2[a] \mid l_2[b] \mid Q\big].$$

The first reduction corresponds to the synchronization between $l_1[a]$ and $l_1\langle\!\langle (X_1, X_2).(l_2[X_1] \mid l_2[X_2] \mid Q) \rangle\!\rangle$, while the second is the synchronization between $l_1[b]$ and $l_1\langle\!\langle (X_2).(l_2[a] \mid l_2[X_2] \mid Q) \rangle\!\rangle$. Fig. 5 depicts these interactions using boxes to denote nested locations.

5.3. The translation, formally

Our translation of compensable processes into adaptable processes relies on a process denoted $\mathtt{extr}\langle\!\langle t, l_1, l_2 \rangle\!\rangle$, which uses $\mathtt{out}^{\mathtt{s}}(l_1, l_2, n, Q)$ to represent the extraction function (Definition 5.4). We need the following functions.

Definition 5.3. Let P be an adaptable process.

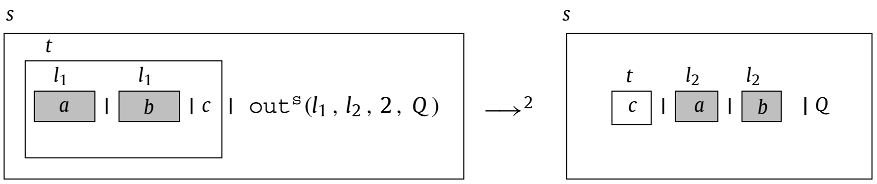

Fig. 5. Example 5.2: Illustrating $\mathtt{out}^{\mathtt{s}}(l_1\,,\,l_2\,,\,2\,,\,Q\,)$.

(1) Function $\mathtt{nl}(l, P)$ denotes the number of occurrences of locations l in process P. It is defined as follows:

$$\begin{array}{ll}
\mathtt{nl}(l_1, l_2[P]) = \mathtt{nl}(l_1, P) + 1 \text{ if } l_1 = l_2 & \mathtt{nl}(l_1, l_2[P]) = \mathtt{nl}(l_1, P) \text{ if } l_1 \neq l_2 \\
\mathtt{nl}(l, (\nu x)\, P) = \mathtt{nl}(l, P) & \mathtt{nl}(l, P \mid Q\,) = \mathtt{nl}(l, P) + \mathtt{nl}(l, Q\,) \\
\mathtt{nl}(l, \mathbf{0}) = \mathtt{nl}(l, !\pi.P) = 0 & \mathtt{nl}(l, l\langle\!\langle (X).Q\,\rangle\!\rangle) = \mathtt{nl}(l, l\{(X).Q\,\}) = 0
\end{array}$$

(2) For a transaction name t and a process P, function $\mathsf{ch}(t, P)$ returns $h_t.\mathbf{0}$ if P equals to an evaluation context with the hole replaced by $h_t.P'$ (for some P'), where the hole is not located within $p_{t,\rho}$, and returns $\mathbf{0}$ otherwise. It is defined as follows:

$$\begin{array}{lll}
\mathsf{ch}(t, h_t.P) = h_t.\mathbf{0} & \mathsf{ch}(s, h_t.P) = \mathbf{0} & \mathsf{ch}(t, \pi.P) = \mathbf{0} \text{ if } \pi \neq h_t \\
\mathsf{ch}(t, l[P]) = \begin{cases} \mathbf{0} & \text{if } l = p_{t,\rho} \\ \mathsf{ch}(t, P) & \text{otherwise} \end{cases} & \mathsf{ch}(t, P \mid Q\,) = \mathsf{ch}(t, P) \mid \mathsf{ch}(t, Q\,) & \mathsf{ch}(t, (\nu x)P) = \mathsf{ch}(t, P) \\
 & \mathsf{ch}(t, \mathbf{0}) = \mathsf{ch}(t, X) = \mathbf{0} & \mathsf{ch}(t, !\pi.P) = \mathbf{0}
\end{array}$$

We are now ready to define process $\mathtt{extr}\langle\!\langle t, l_1, l_2\rangle\!\rangle$:

Definition 5.4 *(Update prefix for extraction).* Let t, l_1, and l_2 be names. We write $\mathtt{extr}\langle\!\langle t, l_1, l_2\rangle\!\rangle$ to stand for the following (subjective) update prefix:

$$\mathtt{extr}\langle\!\langle t, l_1, l_2\rangle\!\rangle = t\langle\!\langle (Y).t[Y] \mid \mathsf{ch}(t, Y) \mid \mathtt{out}^{\mathtt{s}}(l_1\,,\,l_2\,,\,\mathtt{nl}(l_1, Y)\,,\,t\langle\!\langle \dagger\rangle\!\rangle.\overline{h_t})\rangle\!\rangle. \tag{13}$$

Intuitively, process $\mathtt{extr}\langle\!\langle t, l_1, l_2\rangle\!\rangle$ serves to "prepare the ground" for the use of $\mathtt{out}^{\mathtt{s}}(l_1\,,\,l_2\,,\,n\,,\,Q\,)$ which is the one that actually extracts processes from one location and relocates them into another one. Once that occurs, location t is destroyed, which is signaled using name h_t.

We are now ready to formally define the translation of $\mathcal{C}$ into $\mathcal{S}$.

Definition 5.5 *(Translating $\mathcal{C}$ into $\mathcal{S}$).* Let ρ be a path. We define the translation of compensable processes into subjective adaptable processes as a tuple $(\llbracket\cdot\rrbracket_\rho, \varphi_{\llbracket\cdot\rrbracket_\rho})$ where:

(a) The renaming policy $\varphi_{\llbracket\cdot\rrbracket_\rho} : \mathcal{N}_c \longrightarrow \mathcal{P}(\mathcal{N}_a)$ is defined as:

$$\varphi_{\llbracket\cdot\rrbracket_\rho}(x) = \begin{cases} \{x\} & \text{if } x \in \mathcal{N}_s \\ \{x, h_x\} \cup \{p_\rho : x \in \rho\} & \text{if } x \in \mathcal{N}_t \end{cases} \tag{14}$$

(b) The translation $\llbracket\cdot\rrbracket_\rho : \mathcal{C} \longrightarrow \mathcal{S}$ is as in Fig. 6.

Some intuitions are in order. Our renaming function focuses on transaction names: if x is a transaction name, then it is mapped into the set of all (reserved) names that depend on it, including reserved names whose indexed path mentions x. Otherwise, x is mapped into the singleton set $\{x\}$.

We now explain the process mapping in Fig. 6, which is parametric into a path ρ that records the hierarchical structure induced by nested transactions. This way, a process $P \in \mathcal{C}$ is translated as $\llbracket P\rrbracket_\varepsilon$, i.e., P under the empty path ε. Unsurprisingly, the main challenge in the translation is in representing transactions and protected blocks as adaptable processes. More in details:

- The translation of a protected block found at path ρ will be enclosed in the location p_ρ.
- In the translation of $t[P, Q\,]$ we represent processes P and Q independently, using processes in separate locations. More in details:
  - The default activity P is enclosed in a location t while the compensation activity Q is enclosed in a location p_ρ. That is, Q is immediately treated as a protected block.
  - The translation of P is obtained with respect to path t, ρ, thus denoting that t occurs nested within the transactions described by ρ.

$$
\begin{aligned}
[\![\langle P\rangle]\!]_\rho &= p_\rho\big[[\![P]\!]_\varepsilon\big]\\
[\![t[P,Q]]\!]_\rho &= t\Big[[\![P]\!]_{t,\rho}\Big] \mid t.\big(\mathtt{extr}\langle\!\langle t, p_{t,\rho}\,, p_\rho\rangle\!\rangle \mid p_\rho[[\![Q]\!]_\varepsilon]\big)\\
[\![a.P]\!]_\rho &= a.[\![P]\!]_\rho\\
[\![\overline{a}.P]\!]_\rho &= \overline{a}.[\![P]\!]_\rho\\
[\![\overline{t}.P]\!]_\rho &= \overline{t}.h_t.[\![P]\!]_\rho\\
[\![\mathbf{0}]\!]_\rho &= \mathbf{0}\\
[\![(\nu x)P]\!]_\rho &= (\nu x)[\![P]\!]_\rho\\
[\![P_1 \mid P_2]\!]_\rho &= [\![P_1]\!]_\rho \mid [\![P_2]\!]_\rho\\
[\![!\pi.P]\!]_\rho &= ![\![\pi.P]\!]_\rho
\end{aligned}
$$

Fig. 6. Translating $\mathcal{C}$ into $\mathcal{S}$.

- In case of a failure signal $\overline{t}$, our translation activates process $\mathtt{extr}\langle\!\langle t, p_{t,\rho}, p_\rho\rangle\!\rangle$ (cf. Definition 5.4): it extracts all processes located at $p_{t,\rho}$ (which correspond to translations of protected blocks) and moves them to their parent location p_ρ.
- The structure of a transaction and the number of its top-level processes change dynamically. Whenever we need to extract processes located at $p_{t,\rho}$, we first substitute Y in process $\mathtt{out}^{\mathtt{s}}$ (cf. (12)) and in function $\mathsf{ch}(t,\cdot)$ (cf. Definition 5.3), by the current content of the location t.
- We use the reserved name h_t (introduced by $\mathtt{extr}\langle\!\langle t, p_{t,\rho}, p_\rho\rangle\!\rangle$) to control the execution of failure signals; it is particularly useful for error notifications that occur sequentially (one after another in the form of a prefix, e.g. $\overline{t}.\overline{t_1}.\ldots.\overline{t_n}$).
- Once the translation of protected blocks has been moved out of t, the location only contains “garbage”: we can then erase the location t and its contents. To this end, we use the prefix $t\langle\!\langle\dagger\rangle\!\rangle$ (cf. Convention 5.1), which is also introduced by $\mathtt{extr}\langle\!\langle t, p_{t,\rho}, p_\rho\rangle\!\rangle$).
- In case of an internal error notification $\overline{t}$, function $\mathsf{ch}(t,\cdot)$ is particularly useful: it searches for processes of the form $h_t.P$ within the current content at t and replaces them with $h_t.\mathbf{0}$. This is done before the update prefix $t\langle\!\langle\dagger\rangle\!\rangle$ deletes both location t and processes located at t, as described above. Notice that we would need to preserve synchronizations between input h_t and its corresponding output $\overline{h_t}$.

With the above intuitions, translations for the remaining constructs should be self-explanatory.

5.4. Translation correctness

We now establish that the translation $[\![\cdot]\!]_\rho$ is a valid encoding (Definition 4.5). To this end, we address the correctness criteria: compositionality, name invariance, operational correspondence, divergence reflection, and success sensitiveness.

Our results apply for well-formed processes as in Definition 3.4; we briefly discuss this condition. Consider $P = t_1\big[a \mid t_2[b,\overline{b}],\overline{a}\big] \mid \overline{t_1} \mid \overline{t_2}$, the ill-formed process presented in (5). Intuitively, P is not well-formed because it can either compensate t_1 or t_2 in a non-deterministic fashion: if t_1 is compensated then the failure signal on t_2 will not be able to synchronize; if t_2 is compensated then t_1 can still be compensated. That is, $P \longrightarrow^* \langle\overline{b}\rangle \mid \langle\overline{a}\rangle$. Consider how this possibility would be mimicked by $[\![P]\!]_\epsilon$, the encoding of P:

$$
\begin{aligned}
[\![P]\!]_\epsilon = {} & t_1\Big[a \mid t_2\big[b\big] \mid t_2.\Big(t_2\langle\!\langle (Y).t_2[Y] \mid \mathsf{ch}(t_2,Y) \mid t_2\langle\!\langle\dagger\rangle\!\rangle.\overline{h_{t_2}}\rangle\!\rangle \mid p_{t_1}[\overline{b}]\Big)\Big]\\
& \mid t_1.\Big(t_1\langle\!\langle (Y).t_1[Y] \mid \mathsf{ch}(t_1,Y) \mid \mathtt{out}^{\mathtt{s}}(p_{t_1}, p_\epsilon\,, \mathtt{nl}(p_{t_1},Y)\,, t_1\langle\!\langle\dagger\rangle\!\rangle.\overline{h_{t_1}})\rangle\!\rangle \mid p_\epsilon[\overline{a}]\Big) \mid \overline{t_1}.h_{t_1} \mid \overline{t_2}.h_{t_2}\\
\longrightarrow^2 {} & t_1\big[a \mid t_2\big[b\big] \mid t_2\langle\!\langle (Y).t_2[Y] \mid \mathsf{ch}(t_2,Y) \mid t_2\langle\!\langle\dagger\rangle\!\rangle.\overline{h_{t_2}}\rangle\!\rangle \mid p_{t_1,\epsilon}[\overline{b}]\big]\\
& \mid t_1\langle\!\langle (Y).t_1[Y] \mid \mathsf{ch}(t_1,Y) \mid \mathtt{out}^{\mathtt{s}}(p_{t_1}, p_\epsilon\,, \mathtt{nl}(p_{t_1},Y)\,, t_1\langle\!\langle\dagger\rangle\!\rangle.\overline{h_{t_1}})\rangle\!\rangle \mid p_\epsilon[\overline{a}] \mid h_{t_1} \mid h_{t_2}\\
\longrightarrow^4 {} & p_\epsilon[\overline{b}] \mid p_\epsilon[\overline{a}] \mid h_{t_2}.
\end{aligned}
$$

Hence, when applied into ill-formed processes, our encoding induces target processes with “garbage processes” (such as h_{t_2} above), which do not satisfy operational correspondence as defined in Definition 4.5. Specifically, the soundness property would not hold, because $[\![P]\!]_\epsilon$ would have behaviors not present in P. A similar conclusion can be drawn for the other two ill-formed processes presented in (5).

5.4.1. Structural criteria

The compositionality criterion says that the translation of a composite term must be defined in terms of the translations of its subterms. The translation is initially parametrized with ε (i.e., without external names); afterwards, when applied to nested subterms, the list of parameters is extended with transaction names ρ, as specified in Definition 4.5. Accordingly, we consider compositional contexts that depend on an arbitrary list ρ of external transaction names. Nevertheless, our encoding still preserves the main principles of the notion of compositionality. We can translate compensable terms by translating their operator without need to analyze the structure of the subterms. Another peculiarity appears in the process $\mathtt{extr}\langle\langle t, p_{t,\rho}, p_\rho\rangle\rangle$, which is defined in Definition 5.4. It depends on the function $\mathtt{nl}(l_1, Y)$ that dynamically counts the current number of locations l_1 in the content of t. To mediate between these translations of subterms, we define a *context* for each process operator, which depends on free names of the subterms:

Definition 5.6 *(Compositional context).* For every process operator from $\mathcal{C}$, we define a compositional context in $\mathcal{S}$ as follows:

$$C_{\langle\rangle,\rho}[\bullet] = p_\rho\big[[\bullet]\big] \quad C_{t[,],\rho}[\bullet_1, \bullet_2] = t\big[[\bullet_1]\big] \mid t.\big(\mathtt{extr}\langle\langle t, p_{t,\rho}, p_\rho\rangle\rangle \mid p_\rho\big[[\bullet_2]\big]\big) \quad C_{\mid}[\bullet_1, \bullet_2] = [\bullet_1] \mid [\bullet_2]$$
$$C_{a.}[\bullet] = a.[\bullet] \quad C_{\overline{a}.}[\bullet] = \overline{a}.[\bullet] \quad C_{\overline{t}.}[\bullet] = \overline{t}.h_t.[\bullet] \quad C_{(\nu x)}[\bullet] = (\nu x)[\bullet] \quad C_{!\pi.}[\bullet] = !\pi.[\bullet]$$

Using this definition, we may now state the following result:

Theorem 5.3 *(Compositionality for $[\![\cdot]\!]_\rho$). Let ρ be an arbitrary path. For every process operator in $\mathcal{C}$ and for all well-formed compensable processes P and Q it holds that:*

$$[\![\langle P\rangle]\!]_\rho = C_{\langle\rangle,\rho}\,[[\![P]\!]_\varepsilon] \qquad [\![t[P,Q]]\!]_\rho = C_{t[,],\rho}\left[[\![P]\!]_{t,\rho}, [\![Q]\!]_\varepsilon\right] \qquad [\![P \mid Q]\!]_\rho = C_{\mid}\left[[\![P]\!]_\rho, [\![Q]\!]_\rho\right]$$
$$[\![a.P]\!]_\rho = C_{a.}\left[[\![P]\!]_\rho\right] \qquad [\![\overline{t}.P]\!]_\rho = C_{\overline{t}.}\left[[\![P]\!]_\rho\right] \qquad [\![(\nu x)P)]\!]_\rho = C_{(\nu x)}\left[[\![P]\!]_\rho\right]$$
$$[\![\overline{a}.P]\!]_\rho = C_{\overline{a}.}\left[[\![P]\!]_\rho\right] \qquad [\![!\pi.P]\!]_\rho = C_{!\pi.}\left[[\![P]\!]_\rho\right]$$

Proof. Follows directly from the definition of contexts (Definition 5.6) and from the definition of $[\![\cdot]\!]_\rho : \mathcal{C} \longrightarrow \mathcal{S}$ (Fig. 6). See § B.1 at page 40 for further details. □

We now consider *name invariance*. We will say that a function $\sigma : \mathcal{N}_c \to \mathcal{N}_c$ is a *valid substitution* if it is the identity except on a finite set and it respects syntactically the partition of $\mathcal{N}_c$ into subsets $\mathcal{N}_s$ and $\mathcal{N}_t$, i.e., $\sigma(\mathcal{N}_s) \subseteq \mathcal{N}_s$ and $\sigma(\mathcal{N}_t) \subseteq \mathcal{N}_t$. If $\rho = t_1, \ldots, t_n, \varepsilon$, we write $\sigma(\rho)$ to denote the sequence $\sigma(t_1), \ldots, \sigma(t_n), \varepsilon$. We now state name invariance, by relying on the renaming policy in Definition 5.5(a).

Theorem 5.4 *(Name invariance for $[\![\cdot]\!]_\rho$). For every well-formed compensable process P and valid substitution $\sigma : \mathcal{N}_c \to \mathcal{N}_c$ there is a $\sigma' : \mathcal{N}_a \longrightarrow \mathcal{N}_a$ such that:*

$$\textit{(i) for every } x \in \mathcal{N}_c : \varphi_{[\![\cdot]\!]_{\sigma(\rho)}}(\sigma(x)) = \{\sigma'(y) : y \in \varphi_{[\![\cdot]\!]_\rho}(x)\} \qquad \textit{and} \qquad \textit{(ii) } [\![\sigma(P)]\!]_{\sigma(\rho)} = \sigma'([\![P]\!]_\rho).$$

Proof. See § B.2 at page 41. □

5.4.2. Semantic criteria

We prove the three criteria, following the order in which they were introduced in Definition 4.5: operational correspondence, divergence reflection, and success sensitiveness.

Operational correspondence. Among the semantic criteria, operational correspondence is usually the most interesting one, but also the most delicate to prove. We aim to establish a statement of operational correspondence that includes the number of reductions required in $\mathcal{S}$ to correctly mimic a reduction in $\mathcal{C}$. This will allow us to support our claim that subjective updates are more efficient than objective updates (cf. Definition 7.1). To precisely state completeness results we introduce some auxiliary notions.

Definition 5.7. Given a compensable process P, we will write $\mathtt{pb}(P)$ to denote the number of protected blocks in P—see Fig. 7 for a definition.

Given a transaction $t[P,Q]$, the following lemma ensures that the number of protected blocks in the default activity P is equal to the number of locations $p_{t,\rho}$ in $[\![P]\!]_{t,\rho}$ (Definition 5.3).

Lemma 5.5. *Let $t[P,Q]$ and ρ be a well-formed compensable process and an arbitrary path, respectively. Then it holds that* $\mathtt{pb}(P) = \mathtt{nl}(p_{t,\rho}, [\![P]\!]_{t,\rho})$.

Proof. By induction on structure of P.

- $P = \mathbf{0}$ or $P = \pi.P_1$ or $P = !\pi.P_1$: By Definition 5.3, Definition 5.7 and Definition 5.5, we can derive $\mathtt{nl}(p_{t,\rho}, [\![P]\!]_{t,\rho}) = 0 = \mathtt{pb}(P)$.

$$\begin{array}{lll} \mathtt{pb}(\langle P \rangle) = 1 & \mathtt{pb}(P \mid Q) = \mathtt{pb}(P) + \mathtt{pb}(Q) & \mathtt{pb}((\nu x)P) = \mathtt{pb}(P) \\ \mathtt{pb}(!\pi.P) = \mathtt{pb}(\pi.P) = 0 & \mathtt{pb}(t[P, Q]) = 0 & \mathtt{pb}(\mathbf{0}) = 0 \end{array}$$

Fig. 7. Number of protected blocks for discarding semantics.

- $P = \langle P_1 \rangle$: By Definition 5.3, Definition 5.7 and Definition 5.5, $\mathtt{nl}(p_{t,\rho}, [\![\langle P_1 \rangle]\!]_{t,\rho}) = \mathtt{nl}(p_{t,\rho}, p_{t,\rho}\Big[[\![P_1]\!]_\varepsilon\Big]) = 1 = \mathtt{pb}(\langle P_1 \rangle)$.
- $P = s[P_1, Q_1]$: By Definition 5.5, $[\![s[P_1, Q_1]]\!]_{t,\rho} = s\Big[[\![P_1]\!]_{s,t,\rho}\Big] \mid s.\big(\mathtt{extr}\langle\!\langle s, p_{s,t,\rho}\,,\, p_{t,\rho}\rangle\!\rangle \mid p_{t,\rho}[[\![Q_1]\!]_\varepsilon]\big)$. Noticing that $\mathtt{nl}(p_{t,\rho}, [\![P_1]\!]_{s,t,\rho}) = 0$, by application of Definition 5.3 and Definition 5.7, we get $\mathtt{nl}(p_{t,\rho}, [\![s[P_1, Q_1]]\!]_{t,\rho}) = 0 = \mathtt{pb}(s[P_1, Q_1])$.
- $P = P_1 \mid Q_1$: By Definition 5.3 and Definition 5.5, $\mathtt{nl}(p_{t,\rho}, [\![P_1 | Q_1]\!]_{t,\rho}) = \mathtt{nl}(p_{t,\rho}, [\![P_1]\!]_{t,\rho} | [\![Q_1]\!]_{t,\rho}) = \mathtt{nl}(p_{t,\rho}, [\![P_1]\!]_{t,\rho}) + \mathtt{nl}(p_{t,\rho}, [\![Q_1]\!]_{t,\rho})$. By induction hypothesis, we conclude $\mathtt{nl}(p_{t,\rho}, [\![P_1 | Q_1]\!]_{t,\rho}) = \mathtt{pb}(P_1) + \mathtt{pb}(Q_1)$.
- $P = (\nu x)P_1$: By Definition 5.3 and Definition 5.5, $\mathtt{nl}(p_{t,\rho}, [\![(\nu x)P_1]\!]_{t,\rho}) = \mathtt{nl}(p_{t,\rho}, (\nu x)[\![P_1]\!]_{t,\rho}) = \mathtt{nl}(p_{t,\rho}, [\![P_1]\!]_{t,\rho})$. By induction hypothesis and Definition 5.7, $\mathtt{nl}(p_{t,\rho}, [\![(\nu x)P_1]\!]_{t,\rho}) = \mathtt{pb}(P_1) = \mathtt{pb}((\nu x)P_1)$. □

The following example illustrates this claim.

Example 5.6. Let $P = t[P_1, d]$ be a well-formed compensable process, with default activity $P_1 = \langle a \rangle \mid \langle b \rangle \mid c$. By Fig. 7, we have $\mathtt{pb}(P_1) = 2$. Also, by Definition 5.5, we have:

$$[\![P]\!]_\rho = t\Big[[\![P_1]\!]_{t,\rho}\Big] \mid t.\big(\mathtt{extr}\langle\!\langle t, p_{t,\rho}, p_\rho\rangle\!\rangle \mid p_\rho\big[d\big]\big),$$

such that $[\![P_1]\!]_{t,\rho} = p_{t,\rho}\big[a\big] \mid p_{t,\rho}\big[b\big] \mid c$. Now, by Definition 5.3 it is clear that $\mathtt{nl}(p_{t,\rho}, [\![P_1]\!]_{t,\rho}) = 2$.

For the proof of operational correspondence, we introduce a mapping from evaluation contexts of compensable processes into evaluation contexts of adaptable processes.

Definition 5.8. Let ρ be a path. We define mapping $[\![\cdot]\!]_\rho$ from evaluation contexts of compensable processes into evaluation contexts of adaptable processes as follows:

$$\begin{array}{llll} [\![[\bullet]]\!]_\rho = [\bullet] & [\![\langle C[\bullet]\rangle]\!]_\rho = p_\rho[[\![C[\bullet]]\!]_\varepsilon] & [\![C[\bullet] \mid P]\!]_\rho = [\![C[\bullet]]\!]_\rho \mid [\![P]\!]_\rho & [\![(\nu x)C[\bullet]]\!]_\rho = (\nu x)[\![C[\bullet]]\!]_\rho \\ [\![t[C[\bullet], Q]]\!]_\rho = t\big[[\![C[\bullet]]\!]_{t,\rho}\big] \mid t.\big(\mathtt{extr}\langle\!\langle t, p_{t,\rho}, p_\rho\rangle\!\rangle \mid p_\rho[[\![Q]\!]_\varepsilon]\big) & & & \end{array}$$

Convention 5.7. We will use $[\![C]\!]_\rho[P]$ to denote the process that is obtained when the only hole of context $[\![C[\bullet]]\!]_\rho$ is replaced with process P.

We now state our operational correspondence result:

Theorem 5.8 *(Operational correspondence for $[\![\cdot]\!]_\varepsilon$). Let P be a well-formed process in $\mathcal{C}$.*

(1) *If $P \to P'$ then $[\![P]\!]_\varepsilon \longrightarrow^k [\![P']\!]_\varepsilon$ where for*
  a) *$P \equiv E[C[\overline{a}.P_1] \mid D[a.P_2]]$ and $P' \equiv E[C[P_1] \mid D[P_2]]$ it follows $k = 1$,*
  b) *$P \equiv E[C[t[P_1, Q]] \mid D[\overline{t}.P_2]]$ and $P' \equiv E[C[\mathsf{extr}(P_1) \mid \langle Q \rangle] \mid D[P_2]]$ it follows $k = 4 + \mathtt{pb}(P_1)$,*
  c) *$P \equiv C[u[D[\overline{u}.P_1], Q]]$ and $P' \equiv C[\mathsf{extr}(D[P_1]) \mid \langle Q \rangle]$, it follows $k = 4 + \mathtt{pb}(D[P_1])$,*

  for some contexts C, D, E, processes P_1, Q, P_2 and names t, u.

(2) *If $[\![P]\!]_\varepsilon \longrightarrow^n R$ with $n > 0$ then there is P' such that $P \longrightarrow^* P'$ and $R \longrightarrow^* [\![P']\!]_\varepsilon$.*

Proof (Sketch). Here we present an overview to the proof and some auxiliary results.

(1) The proof of completeness is by induction on the derivation of $P \longrightarrow P'$ and uses:
  - Proposition 3.1 (page 7) for determining three base cases. Below we illustrate one of them: the case (*b*) in which reduction corresponds to a synchronization due to an external error notification for a transaction scope.
  - Definition 5.5 (page 14), i.e., the definition of translation.
  - Lemma B.1 (page 43), which maps evaluation contexts in $\mathcal{C}$ into evaluation contexts of $\mathcal{S}$.
  - Lemma B.6 (page 45), which concerns function $\mathsf{ch}(\cdot,\cdot)$.

  We discuss completeness for the particular case (b). We consider $P \equiv E[C[t[P_1, Q]] \mid D[\overline{t}.P_2]]$, with $m = \mathtt{pb}(P_1)$, and $P' \equiv E[C[\mathsf{extr}(P_1) \mid \langle Q \rangle] \mid D[P_2]]$. We have the following derivation, where ρ, ρ', and ρ'' are paths to holes in contexts $E[\bullet]$, $C[\bullet]$, and $D[\bullet]$, respectively:

$$\begin{aligned}
[\![P]\!]_\varepsilon &\equiv [\![E[C[t[P_1,Q]] \mid D[\bar{t}.P_2]]]\!]_\varepsilon = [\![E]\!]_\varepsilon\Big[[\![C[t[P_1,Q]]]\!]_\rho \mid [\![D[\bar{t}.P_2]]\!]_\rho\Big]\\
&= [\![E]\!]_\varepsilon\Big[[\![C]\!]_\rho[[\![t[P_1,Q]]\!]_{\rho'}] \mid [\![D]\!]_\rho[[\![\bar{t}.P_2]\!]_{\rho''}]\Big]\\
&= [\![E]\!]_\varepsilon[[\![C]\!]_\rho\Big[t\big[[\![P_1]\!]_{t,\rho'}\big] \mid t.\big(\mathtt{extr}\langle\!\langle t, p_{t,\rho'}, p_{\rho'}\rangle\!\rangle \mid p_{\rho'}[[\![Q]\!]_\varepsilon]\big)\Big] \mid [\![D]\!]_\rho[\bar{t}.h_t.[\![P_2]\!]_{\rho''}]]\\
&\longrightarrow [\![E]\!]_\varepsilon[[\![C]\!]_\rho\Big[t\big[[\![P_1]\!]_{t,\rho'}\big] \mid \mathtt{extr}\langle\!\langle t, p_{t,\rho'}, p_{\rho'}\rangle\!\rangle \mid p_{\rho'}[[\![Q]\!]_\varepsilon]\Big] \mid [\![D]\!]_\rho[h_t.[\![P_2]\!]_{\rho''}]]\\
&\longrightarrow [\![E]\!]_\varepsilon\Big[[\![C]\!]_\rho\big[t\big[[\![P_1]\!]_{t,\rho'}\big] \mid \mathsf{ch}(t, [\![P_1]\!]_{t,\rho'})\\
&\qquad \mid \mathtt{out}^{\mathsf{s}}(p_{t,\rho'}, p_{\rho'}, \mathtt{nl}(p_{t,\rho'}, [\![P_1]\!]_{t,\rho'}), t\langle\!\langle\dagger\rangle\!\rangle.\overline{h_t}) \mid p_{\rho'}[[\![Q]\!]_\varepsilon]\big] \mid [\![D]\!]_\rho[h_t.[\![P_2]\!]_{\rho''}]\Big]\\
&\longrightarrow^{m+1} [\![E]\!]_\varepsilon\Big[[\![C]\!]_\rho\big[[\![\mathsf{extr}(P_1)]\!]_{\rho'} \mid \overline{h_t} \mid [\![\langle Q\rangle]\!]_{\rho'}\big] \mid [\![D]\!]_\rho\big[h_t.[\![P_2]\!]_{\rho''}\big]\Big]\\
&\longrightarrow [\![E]\!]_\varepsilon\Big[[\![C]\!]_\rho\big[[\![\mathsf{extr}(P_1) \mid \langle Q\rangle]\!]_{\rho'}\big] \mid [\![D]\!]_\rho\big[[\![P_2]\!]_{\rho''}\big]\Big]\\
&= [\![E]\!]_\varepsilon\Big[[\![C[\mathsf{extr}(P_1) \mid \langle Q\rangle]]\!]_\rho \mid [\![D[P_2]]\!]_\rho\Big]\\
&= [\![E[C[\mathsf{extr}(P_1) \mid \langle Q\rangle] \mid D[P_2]]]\!]_\varepsilon\\
&\equiv [\![P']\!]_\varepsilon
\end{aligned}$$

Therefore, we can conclude that for $[\![P]\!]_\varepsilon \longrightarrow^k [\![P']\!]_\varepsilon$ such that $k = 4 + m$.
For more details see B.3.1 (page 42) and § B.3.4 (page 54).

(2) The proof of soundness is by induction on n, i.e., the length of the reduction $[\![P]\!]_\rho \longrightarrow^n R$. We rely crucially on two lemmas (Lemma B.10 and Lemma B.11). Lemma B.10 concerns the shape of processes R and P', whereas Lemma B.11 ensures that the obtained adaptable process R can evolve until reaching a process that corresponds to the translation of a compensable process. More in details:
- By analyzing the processes obtained by translating the composition of a transaction and its externally triggered failure signal (and its computation), we come to Lemma B.8 (page 46), which identifies processes that are created before a synchronization on h_t.
- Similarly, the analysis of the processes obtained by translating a transaction and its internally triggered failure signal (and its computation) leads us to Lemma B.9 (page 49), which identifies processes that are created before a synchronization on h_u.
- In the statement of Lemma B.8 and Lemma B.9 we use the definition of intermediate processes given by Definition B.2 and Definition B.3, respectively. The proofs proceed by case analysis for the step $R \longrightarrow R'$.
- Lemma B.10 (page 50) is about the shape of process R, and also ensures that there is a process P' with an appropriate shape. The proof proceeds by induction on n. The base case uses Lemma B.4 (page 44); in the inductive step, we exploit the fact that the target term R_1 has a specific shape, which is in turn ensured by Lemma B.8 and Lemma B.9.
- Lemma B.11 (page 54) ensures that the adaptable process obtained thanks to Lemma B.8 and Lemma B.9 can evolve until reaching a process that corresponds to the translation of a compensable process.

For full details see B.3.1 (page 42) and § B.20 (page 55). □

In Theorem 5.8, Case (1) concerns completeness, while Case (2) describes soundness. Case (1)–(a) concerns usual synchronizations, which are translated by $[\![\cdot]\!]_\rho$ with an additional synchronization (on name h_t). Cases (1)–(b) and (c) concern synchronizations due to compensation signals; here the analysis distinguishes four cases, as the failure signal can be external or internal (see page 6) and the transaction can be replicated or not. In all cases, the number of reduction steps required to mimic the source transition depends on the number of protected blocks of the transaction being canceled. We illustrate this with an example.

Example 5.9. $P = s\big[t[\langle a\rangle \mid \langle b\rangle \mid c, d], \mathbf{0}\big] \mid \bar{t}.\bar{s}$ is a well-formed compensable process. By the LTS of $\mathcal{C}$ (cf. Fig. 2), we have:

$$P \xrightarrow{\tau} s[\langle a\rangle \mid \langle b\rangle \mid \langle d\rangle, \mathbf{0}] \mid \bar{s} \xrightarrow{\tau} \langle a\rangle \mid \langle b\rangle \mid \langle d\rangle.$$

The encoding of P is obtained by expanding Definition 5.5:

$$\begin{aligned}
[\![P]\!]_\varepsilon &= s\Big[t\big[p_{t,s}[a] \mid p_{t,s}[b] \mid c\big] \mid t.\big(\mathtt{extr}\langle\!\langle t, p_{t,s}, p_s\rangle\!\rangle \mid p_s[d]\big)\Big] \mid s.\mathtt{extr}\langle\!\langle s, p_s, p_\varepsilon\rangle\!\rangle \mid \bar{t}.h_t.\bar{s}.h_s\\
&\longrightarrow^6 s\Big[p_s[a] \mid p_s[b] \mid p_s[d]\Big] \mid s.\mathtt{extr}\langle\!\langle s, p_s, p_\varepsilon\rangle\!\rangle \mid \bar{s}.h_s
\end{aligned}$$

$$\longrightarrow^7 p_\varepsilon[a] \mid p_\varepsilon\big[b\big] \mid p_\varepsilon\big[d\big] = [\![\langle a\rangle \mid \langle b\rangle \mid \langle d\rangle]\!]_\varepsilon.$$

Let us write P_1 to denote the process $\langle a\rangle \mid \langle b\rangle \mid c$ (the default activity of transaction t) and P_2 to denote the process $\langle a\rangle \mid \langle b\rangle \mid \langle d\rangle$ (the process obtained above). Our operational correspondence result ensures that k in $[\![P]\!]_\varepsilon \longrightarrow^k [\![P_2]\!]_\varepsilon$ is equal to

$$\begin{aligned} k &= \underbrace{4+\mathtt{pb}(P_1)}_{\text{for transaction } t} + \underbrace{4+\mathtt{pb}(P_2)}_{\text{for transaction } s} \\ &= 6+7=13 \end{aligned}$$

Let us analyze in detail these reduction steps:

i) The first step corresponds to the synchronization between t and $\overline{t}$.
ii) Once process $\mathtt{extr}\langle\!\langle t, p_{t,s}, p_s\rangle\!\rangle$ is released, the second step is a synchronization on update prefix $t\langle\!\langle (Y).t[Y] \mid \mathsf{ch}(t,Y) \mid \mathtt{out}^{\mathtt{s}}(p_{t,s}\,,\, p_s\,,\, \mathtt{nl}(p_{t,s},Y)\,,\, t\langle\!\langle\dagger\rangle\!\rangle.\overline{h_t})\rangle\!\rangle$ and location $t\big[p_{t,s}\big[a\big] \mid p_{t,s}\big[b\big] \mid c\big]$.
iii) Since we get process $\mathtt{out}^{\mathtt{s}}(p_{t,s}\,,\, p_s\,,\, 2\,,\, t\langle\!\langle\dagger\rangle\!\rangle.\overline{h_t})$, the third and fourth steps correspond to synchronizations between locations $p_{t,s}[a]$ and $p_{t,s}[b]$ with the (nested) update prefix $p_{t,s}\langle\!\langle (X_1,X_2).p_s[X_1] \mid p_s[X_2] \mid t\langle\!\langle\dagger\rangle\!\rangle.\overline{h_t}\rangle\!\rangle$, which relocates the encoding of protected blocks.
iv) The fifth step is the synchronization between update prefix $t\langle\!\langle\dagger\rangle\!\rangle$ and location $t[\ldots]$, whereby the location is deleted together with its content (cf. equation (11));
v) The sixth reduction step is a synchronization on name h_t, which enables behavior corresponding to the encoding of transaction s.

To encode the failure of transaction s, we repeat the exact same steps as before. For location s we have one more reduction step, because in process $\mathtt{out}^{\mathtt{s}}(p_s\,,\, p_\varepsilon\,,\, 3\,,\, s\langle\!\langle\dagger\rangle\!\rangle.\overline{h_s})$ we have three locations $p_s[\ldots]$ that have to be relocated on location $p_\varepsilon[\ldots]$.

We illustrate the encoding also on the *Hotel booking scenario* discussed earlier (§ 2, Example 2.1, page 4).

Example 5.10. Recall process *Reservation* from Example 2.1. We have:

$$\begin{aligned} [\![Reservation]\!]_\varepsilon = \;& t\big[book.pay.invoice\big] \mid t.(\mathtt{extr}\langle\!\langle t, p_t, p_\varepsilon\rangle\!\rangle \mid p_\varepsilon[\overline{refund}]) \mid \overline{book}.\overline{pay}.\overline{t}.h_t.refund \\ \longrightarrow^3 \;& t\big[invoice\big] \mid t\langle\!\langle (Y).t[Y] \mid \mathsf{ch}(t,Y \mid \mathtt{out}^{\mathtt{s}}(p_t\,,\, p_\varepsilon\,,\, \mathtt{nl}(p_t,Y)\,,\, t\langle\!\langle\dagger\rangle\!\rangle.\overline{h_t}))\rangle\!\rangle \\ & \mid p_\varepsilon[\overline{refund}] \mid h_t.refund \\ \longrightarrow \;& t\big[invoice\big] \mid \mathsf{ch}(t, invoice) \mid \mathtt{out}^{\mathtt{s}}(p_t\,,\, p_\varepsilon\,,\, \mathtt{nl}(p_t, invoice)\,,\, t\langle\!\langle\dagger\rangle\!\rangle.\overline{h_t}) \\ & \mid p_\varepsilon[\overline{refund}] \mid h_t.refund \\ \equiv \;& t\big[invoice\big] \mid \mathtt{out}^{\mathtt{s}}(p_t\,,\, p_\varepsilon\,,\, 0\,,\, t\langle\!\langle\dagger\rangle\!\rangle.\overline{h_t}) \mid p_\varepsilon[\overline{refund}] \mid h_t.refund \\ = \;& t\big[invoice\big] \mid t\langle\!\langle\dagger\rangle\!\rangle.\overline{h_t} \mid p_\varepsilon[\overline{refund}] \mid h_t.refund \\ \longrightarrow^3 \;& p_\varepsilon[\mathbf{0}]. \end{aligned}$$

Therefore, we get $[\![Reservation]\!]_\varepsilon \longrightarrow^7 p_\varepsilon[\mathbf{0}]$. There are three reduction steps, denoted $\longrightarrow^3$, as a result of synchronizations on input prefixes: *book*, *pay* and t with corresponding outputs. Now, the structure of the default activity of transaction is changed and we have one reduction step for updating its current content. After that, there are three more reduction steps: one for erasing the location t and its content, and two reduction steps as result of synchronizations on input names h_t and *refund* with corresponding outputs.

Divergence reflection. In the following, we are going to prove that the encoding does not introduce divergent computations. We need the following definition, which counts all protected blocks in process P.

Definition 5.9. Given a well-formed compensable process P, we will write $\mathtt{npb}(P)$ to denote the number of protected blocks in P—see Fig. 8 for a definition.

Notice that $\mathtt{npb}(P)$ is different from $\mathtt{pb}(P)$ in Definition 5.7. The difference is in the definition for processes $\langle P\rangle$ and $t[P,Q]$. In Definition 5.7 we count all processes that may become protected, e.g., after a reduction of the considered compensable process. Intuitively, with $\mathtt{npb}(P)$ we are looking for protected blocks at all levels of the observed compensable process.

$$\begin{array}{lll}
\mathtt{npb}(\langle P\rangle) = 1 + \mathtt{npb}(P) & \mathtt{npb}(P \mid Q) = \mathtt{npb}(P) + \mathtt{npb}(Q) & \mathtt{npb}((\nu x)P) = \mathtt{npb}(P) \\
\mathtt{npb}(!\pi.P) = \mathtt{npb}(\pi.P) = 0 & \mathtt{npb}(t[P, Q]) = 1 + \mathtt{npb}(P) + \mathtt{npb}(Q) & \mathtt{npb}(\mathbf{0}) = 0
\end{array}$$

Fig. 8. Number of protected blocks.

To establish divergence reflection, we relate a sequence of adaptable processes and a sequence of compensable processes. One reduction of an adaptable process from the sequence corresponds either to one reduction of the corresponding compensable process or to equal consecutive compensable processes. This reflects that a single reduction of compensable processes is mimicked by several reductions of a corresponding adaptable process. The following lemma proves that such a relation does exist, providing also the upper bound for the number of successive, non-equivalent, adaptable processes with the property that their corresponding adaptable processes are equal. This last property directly induces that the set of compensable processes is infinite, too.

Lemma 5.11. *Let $\{R_i\}_{i\geq 0}$ be a sequence of adaptable processes such that $R_i \longrightarrow R_{i+1}$, with $R_0 = [\![P_0]\!]_\rho$, for some compensable process P_0 and path ρ. Then for every $i \geq 1$ there is P_i such that*

(i) $R_i \longrightarrow^* [\![P_i]\!]_\rho$,
(ii) $P_{i-1} = P_i$ *or* $P_{i-1} \longrightarrow P_i$, *and*
(iii) $R_i \not\equiv R_{i+1} \not\equiv \ldots \not\equiv R_{i+m}$ *and* $P_i = P_{i+1} = \ldots = P_{i+m}$ *imply* $m \leq 4 + \mathtt{npb}(P_0)$.

Proof. See § B.4 at page 57. □

The following theorem concerns *infinite* reduction sequences: it says that an infinite reduction sequence originating from a target term can only arise from an infinite reduction sequence of a corresponding source term. Hence, it suffices to establish divergence reflection, as in Definition 4.5:

Theorem 5.12 *(Divergence reflection for $[\![\cdot]\!]_\rho$). Let $\{R_i\}_{i\geq 0}$ be an infinite sequence of adaptable processes such that*

$$(1)\ R_0 = [\![P_0]\!]_\rho \text{ for some } P_0 \text{ and } \rho, \qquad \text{and} \qquad (2)\quad R_i \longrightarrow R_{i+1} \text{ for any } i \geq 0.$$

Then there is an infinite sequence of adaptable processes $\{P'_j\}_{j\geq 0}$ such that

$$(3)\ P'_0 = P_0, \qquad \text{and} \qquad (4)\quad P'_j \longrightarrow P'_{j+1} \text{ for any } j \geq 0.$$

Proof. By Lemma 5.11, there is a sequence $\{P_i\}_{i\geq 0}$ such that

$$\text{(i) } R_i \longrightarrow^* [\![P_i]\!]_\rho \qquad \text{and} \qquad \text{(ii)}\quad P_{i-1} = P_i \text{ or } P_{i-1} \longrightarrow P_i.$$

Consider now a sequence of compensable processes $P'_0, P'_1, P'_2, \ldots$ such that

(1) $P'_{j-1} \longrightarrow P'_j$, for any $j \geq 1$, and
(2) for every i there is j such that $P_i = P'_j$.

By Lemma 5.11, at most $4+\mathtt{npb}(P_0)$ reduction steps from the sequence $\{R_i\}_{i\geq 0}$ correspond to one reduction step of $\{P'_j\}_{j\geq 0}$. Hence, the number of processes in $\{P'_j\}_{j\geq 0}$ is not less than the number of processes $\{R_i\}_{i\geq 0}$ divided by $4 + \mathtt{npb}(P_0)$. Since the sequence $\{R_i\}_{i\geq 0}$ is infinite, the same holds for $\{P'_j\}_{j\geq 0}$. □

Success sensitiveness. To prove that the translation satisfies success sensitiveness we need first to extend Definition 5.5 with $[\![\checkmark]\!]_\rho = \checkmark$.

Further, we adapt the definition of may-succeed (Definition 4.4) to adaptable and compensable processes. It is defined in exactly the same way for the two calculi, but it relies on different definitions of operational semantics and evaluation contexts.

Definition 5.10. Let P be an adaptable/compensable process. We say that P may-succeeds, denoted $P \Downarrow$, if $P \longrightarrow^* P'$ and $P' = C[\checkmark]$ for some process P' and evaluation context $C[\bullet]$.

Theorem 5.13 *(Success sensitiveness for $[\![\cdot]\!]_\rho$). Let P be a well-formed compensable process and ρ an arbitrary path. Then $P \Downarrow$ if and only if $[\![P]\!]_\rho \Downarrow$.*

Proof. See § B.5 at page 58. □

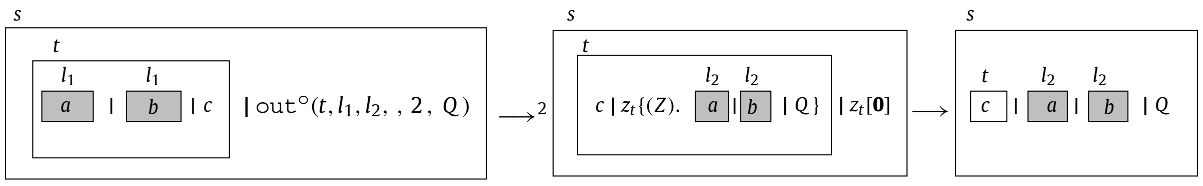

Fig. 9. Example 6.2: Illustrating $\mathtt{out}^{\circ}(t, l_1, l_2, 2, Q)$.

6. Encoding $\mathcal{C}$ into $\mathcal{O}$: the case of objective update

Having detailed a valid encoding of compensable processes into adaptable processes with subjective update, in this section we turn our attention to the case of adaptable processes with *objective update*, in which a located process is reconfigured in its own context by an update prefix residing at a different context (cf. §3.3). Here we define a translation $[\![\cdot]\!]^{\circ}_{\rho}$ which we prove to be a valid encoding (cf. Definition 4.5).

6.1. The translation, informally

To encode transactions and their extraction function we use the auxiliary process $\mathtt{out}^{\circ}(t, l_1, l_2, n, Q)$, which is similar to the process $\mathtt{out}^{s}(l_1, l_2, n, Q)$ (cf. (12)) that we used in the encoding $[\![\cdot]\!]_{\rho}$.

Using objective update prefixes, we define this auxiliary process as follows:

$$\mathtt{out}^{\circ}(t, l_1, l_2, n, Q) = \begin{cases} Q & \text{if } n = 0 \\ l_1\{(X_1, \dots, X_n).z_t\{(Z).\prod_{i=1}^{n} l_2[X_i] \mid Q\}\}.z_t[\mathbf{0}] & \text{if } n > 0. \end{cases} \tag{15}$$

It is instructive to compare processes $\mathtt{out}^{s}$ (12) and $\mathtt{out}^{\circ}$ (15), because differences between them will reflect directly on the efficiency of our encodings. These differences concern parameter n:

Remark 6.1 *(Comparing* $\mathtt{out}^{s}$ *and* $\mathtt{out}^{\circ}$*).* Consider the case $n > 0$: while in $\mathtt{out}^{\circ}$ the process $\prod_{i=1}^{n} l_2[X_i] \mid Q$ appears enclosed inside an update prefix on name z_t, in $\mathtt{out}^{s}$ this is not the case. In $\mathtt{out}^{\circ}$, once n updates on name l_1 have been executed, the resulting process $\prod_{i=1}^{n} l_2[P_i] \mid Q$ will be enclosed in a (wrong) location (say, t). Process $\prod_{i=1}^{n} l_2[P_i] \mid Q$ must be relocated and t must be deleted. In (15), this relocation is achieved via a synchronization on name z_t. In contrast, because $\mathtt{out}^{s}$ uses subjective updates, process reconfiguration follows in the opposite direction. This ensures that, after n updates, process $\prod_{i=1}^{n} l_2[P_i] \mid Q$ will remain in its original location, and so no relocation using z_t is needed—see Example 5.2 and Fig. 5.

Example 6.2 *(Example 5.2, revisited).* Consider process $P' = s\Big[t\big[l_1[a] \mid l_1[b] \mid c\big] \mid \mathtt{out}^{\circ}(t, l_1, l_2, 2, Q)\Big]$, similar to the process in Example 5.2. P' has the following reductions, which are illustrated in Fig. 9:

$$\begin{aligned} P' = \;& s\Big[t\big[l_1[a] \mid l_1[b] \mid c\big] \mid l_1\{(X_1, X_2).z_t\{(Z).l_2[X_1] \mid l_2[X_2] \mid Q\}\}.z_t[\mathbf{0}]\Big] \\ \longrightarrow^2 \;& s\Big[t\big[c \mid z_t\{(Z).l_2[a] \mid l_2[b] \mid Q\}\big] \mid z_t[\mathbf{0}]\Big] \\ \longrightarrow \;& s\Big[t\big[c\big] \mid l_2[a] \mid l_2[b] \mid Q\Big] \end{aligned}$$

In this case, the wrong location is t: the last reduction is needed to move process $l_2[a] \mid l_2[b] \mid Q$ out of z_t.

Notice that the number n of protected blocks in the default activity of the transaction scope is directly related to the number of reduction steps induced by our translations. If $n = 0$ then the number of reduction steps will be the same for subjective and objective updates; otherwise, if $n > 0$, the translation with subjective update will exhibit less reduction steps than the translation with objective update.

6.2. The translation, formally

The function for determining the number of locations $\mathtt{nl}(\cdot, \cdot)$ in an adaptable process and the function $\mathsf{ch}(t, \cdot)$ are as introduced in Definition 5.3. We now define process $\mathtt{extr}\{t, l_1, l_2\}$:

$$
\begin{aligned}
[\![\langle P\rangle]\!]^{\circ}_{\rho} &= p_{\rho}\big[[\![P]\!]^{\circ}_{\varepsilon}\big] \\
[\![t[P,Q]]\!]^{\circ}_{\rho} &= t\Big[[\![P]\!]^{\circ}_{t,\rho}\Big] \mid t.\big(\mathtt{extr}\{t, p_{t,\rho}, p_{\rho}\} \mid p_{\rho}[[\![Q]\!]^{\circ}_{\varepsilon}]\big) \\
[\![a.P]\!]^{\circ}_{\rho} &= a.[\![P]\!]^{\circ}_{\rho} \\
[\![\overline{a}.P]\!]^{\circ}_{\rho} &= \overline{a}.[\![P]\!]^{\circ}_{\rho} \\
[\![\overline{t}.P]\!]^{\circ}_{\rho} &= \overline{t}.h_t.[\![P]\!]^{\circ}_{\rho}
\end{aligned}
$$

Fig. 10. Translating $\mathcal{C}$ into $\mathcal{O}$.

Definition 6.1 *(Update prefix for extraction).* Let t, l_1, and l_2 be names. We write $\mathtt{extr}\{t,l_1,l_2\}$ to stand for the following (objective) update prefix:

$$
\mathtt{extr}\{t,l_1,l_2\} = t\{(Y).t[Y] \mid \mathsf{ch}(t,Y) \mid \mathtt{out}^{\circ}(t,l_1,\,l_2\,,\,\mathtt{nl}(l,Y)\,,\,t\{\dagger\}.\overline{h_t})\}. \tag{16}
$$

The intuitions for process $\mathtt{extr}\{t,l_1,l_2\}$ are just as for process $\mathtt{extr}\langle\!\langle t,l_1,l_2\rangle\!\rangle$ given in Definition 5.4. We can now formally define the translation of $\mathcal{C}$ into $\mathcal{O}$:

Definition 6.2 *(Translation $\mathcal{C}$ into $\mathcal{O}$).* Let ρ be a path. We define the translation of compensable processes into objective adaptable processes as a tuple $([\![\cdot]\!]^{\circ}_{\rho}, \varphi_{[\![\cdot]\!]^{\circ}_{\rho}})$ where:

(a)

$$
\varphi_{[\![\cdot]\!]_{\rho}}(x) = \begin{cases} \{x\} & \text{if } x \in \mathcal{N}_s \\ \{x, h_x, z_x\} \cup \{p_{\rho} : x \in \rho\} & \text{if } x \in \mathcal{N}_t \end{cases} \tag{17}
$$

(b) $[\![\cdot]\!]^{\circ}_{\rho} : \mathcal{C} \longrightarrow \mathcal{O}$ is as defined in Fig. 10 and as a homomorphism for other operators.

Intuitions for the translation of $t[P,Q]$ are as in the case of subjective update. For erasing the location and all unnecessary processes in it, in this case we need an update prefix $t\{\dagger\}$ (cf. Convention 5.1).

6.3. Translation correctness

We prove that the translation $[\![\cdot]\!]^{\circ}_{\rho}$ is a valid encoding (cf. Definition 4.5). We thus consider the five criteria: compositionality, name invariance, and operational correspondence, divergence reflection, and success sensitiveness.

6.3.1. Structural criteria

The first property is compositionality. Compositionality for $[\![\cdot]\!]^{\circ}_{\rho}$ as well as compositionality for $[\![\cdot]\!]_{\rho}$ (cf. Theorem 5.3) includes a path ρ in its formulation.

Theorem 6.3 *(Compositionality for $[\![\cdot]\!]^{\circ}_{\rho}$). Let ρ be an arbitrary path. For every process operator in $\mathcal{C}$ and for all compensable processes P and Q it holds that:*

$$
\begin{array}{lll}
[\![\langle P\rangle]\!]^{\circ}_{\rho} = C_{\langle\rangle,\rho}\big[[\![P]\!]^{\circ}_{\varepsilon}\big] & [\![t[P,Q]]\!]^{\circ}_{\rho} = C_{t[,],\rho}\Big[[\![P]\!]^{\circ}_{t,\rho}, [\![Q]\!]^{\circ}_{\varepsilon}\Big] & [\![P \mid Q]\!]^{\circ}_{\rho} = C_{\mid}\Big[[\![P]\!]^{\circ}_{\rho}, [\![Q]\!]^{\circ}_{\rho}\Big] \\
[\![a.P]\!]^{\circ}_{\rho} = C_{a.}\Big[[\![P]\!]^{\circ}_{\rho}\Big] & [\![\overline{t}.P]\!]^{\circ}_{\rho} = C_{\overline{t}.}\Big[[\![P]\!]^{\circ}_{\rho}\Big] & [\![(\nu x)P)]\!]^{\circ}_{\rho} = C_{(\nu x)}\Big[[\![P]\!]^{\circ}_{\rho}\Big] \\
[\![\overline{a}.P]\!]^{\circ}_{\rho} = C_{\overline{a}.}\Big[[\![P]\!]^{\circ}_{\rho}\Big] & [\![!\pi.P]\!]^{\circ}_{\rho} = C_{!\pi.}\Big[[\![P]\!]^{\circ}_{\rho}\Big] &
\end{array}
$$

Proof. Follows directly from the definition of contexts (Definition 5.6) and from the definition of $[\![\cdot]\!]^{\circ}_{\rho} : \mathcal{C} \longrightarrow \mathcal{O}$ (Fig. 10) and has the same derivation as the proof of Theorem 5.3. □

The second property is name invariance with respect to the renaming policy in Definition 5.5 Case (b).

Theorem 6.4 *(Name invariance for $[\![\cdot]\!]^{\circ}_{\rho}$). For every well-formed compensable processes P and substitution $\sigma : \mathcal{N}_c \longrightarrow \mathcal{N}_c$ there is $\sigma' : \mathcal{N}_a \longrightarrow \mathcal{N}_a$ such that:*

(i) for every $x \in \mathcal{N}_c : \varphi_{[\![\cdot]\!]^{\circ}_{\sigma(\rho)}}(\sigma(x)) = \{\sigma'(y) : y \in \varphi_{[\![\cdot]\!]^{\circ}_{\rho}}(x)\}$ and (ii) $[\![\sigma(P)]\!]^{\circ}_{\sigma(\rho)} = \sigma'([\![P]\!]^{\circ}_{\rho})$.

Proof. The proof proceeds in the same way as the proof of Theorem 5.4 by using $[\![\cdot]\!]^{\circ}_{\rho}$ instead of $[\![\cdot]\!]_{\rho}$. □

6.3.2. Semantic criteria

We consider operational correspondence, divergence reflection, and success sensitiveness.

Operational correspondence. As before, we are interested in precisely accounting for the number of computation steps induced by our translation. We need the following definition.

Definition 6.3. Let P be a well-formed compensable process, then function $\mathtt{Z}(P)$ is defined as follows:

$$\mathtt{Z}(P) = \begin{cases} 0 & \text{if } \mathtt{pb}(P) = 0, \\ 1 & \text{if } \mathtt{pb}(P) > 0. \end{cases}$$

The number of reduction steps required for translating a transaction scope depends on the number of protected blocks in its default activity. As already mentioned, if the default activity contains at least one protected block then the translation of the transaction has an update location on name z_t (it occurs in process $\mathtt{out}^\circ$, cf. (15)); otherwise (if the number of protected blocks is zero) the number of reduction steps is the same as in the subjective case. This fact is presented by using the function $\mathtt{Z}(P)$ in the following theorem for operational correspondence.

Theorem 6.5 *(Operational correspondence for $[\![\cdot]\!]^\circ_\varepsilon$). Let P be a well-formed process in $\mathcal{C}$.*

(1) *If $P \to P'$ then $[\![P]\!]^\circ_\varepsilon \longrightarrow^k [\![P']\!]^\circ_\varepsilon$ where for*
   a) *$P \equiv E[C[\overline{a}.P_1] \mid D[a.P_2]]$ and $P' \equiv E[C[P_1] \mid D[P_2]]$ it follows that $k = 1$.*
   b) *$P \equiv E[C[t[P_1, Q]] \mid D[\overline{t}.P_2]]$ and $P' \equiv E[C[\mathsf{extr}(P_1) \mid \langle Q \rangle] \mid D[P_2]]$ it follows $k = 4 + \mathtt{pb}(P_1) + \mathtt{Z}(P_1)$,*
   c) *$P \equiv C[u[D[\overline{u}.P_1], Q]]$ and $P' \equiv C[\mathsf{extr}(D[P_1]) \mid \langle Q \rangle]$, it follows $k = 4 + \mathtt{pb}(D[P_1]) + \mathtt{Z}(D[P_1])$.*

   for some contexts C, D, E, processes P_1, Q, P_2 and names t, u.
(2) *If $[\![P]\!]^\circ_\varepsilon \longrightarrow^n R$ with $n > 0$ then there is P' such that $P \longrightarrow^* P'$ and $R \longrightarrow^* [\![P']\!]^\circ_\varepsilon$.*

Proof (Sketch). We present an overview to the proof, giving pointers to results in the appendices:

(1) The proof of completeness is by induction on the derivation of $P \longrightarrow P'$ and uses:
   - Proposition 3.1 (page 7) for determining three base cases. Below we illustrate one of them: the case (c) in which reduction corresponds to a synchronization due to an internal error notification for a transaction scope.
   - The definition of translation, given in Definition 6.2 (page 22).
   - Lemma B.1 (page 43) and Lemma B.6 (page 45) hold also for translation $[\![\cdot]\!]^\circ_\rho$, and their role is explained in the proof sketch of Theorem 5.8 (1).

   Now, we illustrate case (c). Therefore, we consider $P \equiv C[u[D[\overline{u}.P_1], Q]]$, with $m = \mathtt{pb}(D[P_1])$, and $P' \equiv C[\mathsf{extr}(D[P_1]) \mid \langle Q \rangle]$. We have the following derivation where paths ρ, ρ', and ρ'' are paths to holes in contexts $E[\bullet]$, $C[\bullet]$, and $D[\bullet]$, respectively:

$$\begin{aligned}
[\![P]\!]^\circ_\varepsilon &\equiv [\![C[u[D[\overline{u}.P_1], Q]]]\!]^\circ_\varepsilon = [\![C]\!]^\circ_\varepsilon[[\![u[D[\overline{u}.P_1], Q]]\!]^\circ_\rho] \\
&= [\![C]\!]^\circ_\varepsilon[u\big[[\![D[\overline{u}.P_1]]\!]^\circ_{u,\rho}\big] \mid u.\big(\mathtt{extr}\{u, p_{u,\rho}, p_\rho\} \mid p_\rho[[\![Q]\!]^\circ_\varepsilon]\big)]] \\
&= [\![C]\!]^\circ_\varepsilon[u\big[[\![D]\!]^\circ_{u,\rho}[\overline{u}.h_u.[\![P_1]\!]^\circ_{\rho'}]\big] \mid u.(\mathtt{extr}\{u, p_{u,\rho}, p_\rho\} \mid p_\rho[[\![Q]\!]^\circ_\varepsilon])] \\
&\longrightarrow [\![C]\!]^\circ_\varepsilon\Big[u\big[[\![D]\!]^\circ_{u,\rho}\big[h_u.[\![P_1]\!]^\circ_{\rho'}\big]\big] \\
&\qquad \mid u\{(Y).u[Y] \mid \mathsf{ch}(u, Y) \mid \mathtt{out}^\circ(u, p_{u,\rho}, p_\rho\,, \mathtt{nl}(p_{u,\rho}, Y)\,, u\{\dagger\}.\overline{h_u})\} \mid p_\rho[[\![Q]\!]^\circ_\varepsilon]\Big] \\
&\longrightarrow [\![C]\!]^\circ_\varepsilon\Big[u\big[[\![D]\!]^\circ_{u,\rho}\big[h_u.[\![P_1]\!]^\circ_{\rho'}\big]\big] \mid \mathsf{ch}(u, [\![D]\!]^\circ_{u,\rho}\big[h_u.[\![P_1]\!]^\circ_{\rho'}\big]) \\
&\qquad \mid \mathtt{out}^\circ(u, p_{u,\rho}, p_\rho\,, \mathtt{nl}(p_{u,\rho}, [\![D]\!]^\circ_{u,\rho}\big[h_u.[\![P_1]\!]^\circ_{\rho'}\big])\,, u\{\dagger\}.\overline{h_u}) \mid p_\rho[[\![Q]\!]^\circ_\varepsilon]\Big] \\
&\longrightarrow^{m+2} [\![C]\!]^\circ_\varepsilon\Big[[\![\mathsf{extr}(D[P_1])]\!]^\circ_\rho \mid \overline{h_u} \mid h_u \mid p_\rho[[\![Q]\!]^\circ_\varepsilon]\Big] \\
&\longrightarrow [\![C]\!]^\circ_\varepsilon[\![\mathsf{extr}(D[P_1])]\!]^\circ_\rho \mid p_\rho[[\![Q]\!]^\circ_\varepsilon]\Big] \\
&= [\![C[\mathsf{extr}(D[P_1]) \mid \langle Q \rangle]\!]^\circ_\varepsilon \\
&\equiv [\![P']\!]^\circ_\varepsilon
\end{aligned}$$

   Therefore, the number of reduction steps is $k = 4 + m + \mathtt{Z}(D[P_1]) = 5 + m$.
   For more details see B.3.1 (page 42) and § C.1.1 (page 62).

(2) To prove soundness, we follow the same ideas as for the proof of soundness for $[\![\cdot]\!]_\rho$ (Theorem 5.8 (2)). The proof is by induction on n, i.e., the length of the reduction $[\![P]\!]^\circ_\rho \longrightarrow^n R$, and proceeds as for Theorem 5.8 (2), because the auxiliary results that we used for $[\![\cdot]\!]_\rho$ hold also for $[\![\cdot]\!]^\circ_\rho$. Most notably, we rely crucially on two lemmas: Lemma B.10, which concerns the shape of processes R and P', and Lemma B.11, which ensures that the obtained adaptable process R can evolve until reaching a process that corresponds to the translation of a compensable process.

For more details see B.3.1 (page 42) and § C.1.1 at page 64. □

In Theorem 6.5, Case (1) concerns completeness, while Case (2) describes soundness. The following example illustrates the operational correspondence property:

Example 6.6. Let P be a process as in Example 5.9 (page 18). Expanding Definition 6.2 the following holds:

$$\begin{array}{rl}
[\![P]\!]^\circ_\varepsilon = & s\Big[t\big[p_{t,s}[a] \mid p_{t,s}[b \mid c] \mid t.(\mathtt{extr}\{t, p_{t,s}, p_s\} \mid p_s[d])\big]\Big] \mid s.\mathtt{extr}\{s, p_s, p_\varepsilon\} \mid \bar{t}.h_t.\bar{s}.h_s \\
= & s\Big[t\big[p_{t,s}[a] \mid p_{t,s}[b] \mid c\big] \mid t.\big(t\{(Y).t[Y] \mid \mathsf{ch}(t,Y) \mid \mathtt{out}^\circ(t, p_{t,s}, p_s\,,\, \mathtt{nl}(p_{t,s}, Y)\,,\, t\{\dagger\}.\overline{h_t})\} \mid p_s[d]\big)\Big] \\
& \mid s.\mathtt{extr}\{s, p_s, p_\varepsilon\} \mid \bar{t}.h_t.\bar{s}.h_s \\
\longrightarrow^2 & s\Big[t\big[p_{t,s}[a] \mid p_{t,s}[b] \mid c\big] \mid \mathtt{out}^\circ(t, p_{t,s}\,,\, p_s\,,\, 2\,,\, t\{\dagger\}.\overline{h_t}) \mid p_s[d]\Big] \mid s.\mathtt{extr}\{s, p_s, p_\varepsilon\} \mid h_t.\bar{s}.h_s \\
\longrightarrow^2 & s\Big[t\big[z_t\{(Z).p_s[a] \mid p_s[b] \mid t\{\dagger\}.h_t\}\big] \mid z_t[\mathbf{0}] \mid p_s[d]\Big] \mid s.\mathtt{extr}\{s, p_s, p_\varepsilon\} \mid \overline{h_t}.\bar{s}.h_s \\
\longrightarrow^3 & s\Big[p_s[a] \mid p_s[b] \mid p_s[d]\Big] \mid s.\mathtt{extr}\{s, p_s, p_\varepsilon\} \mid \bar{s}.h_s \\
\longrightarrow^8 & p_\varepsilon[a] \mid p_\varepsilon[b] \mid p_\varepsilon[d] \\
= & [\![\langle a\rangle \mid \langle b\rangle \mid \langle d\rangle]\!]^\circ_\varepsilon.
\end{array}$$

We have $[\![P]\!]^\circ_\varepsilon \longrightarrow^k [\![P_2]\!]^\circ_\varepsilon$ with $k = 15$:

$$\begin{aligned}
k = & \underbrace{4 + \mathtt{pb}(P_1) + \mathtt{Z}(t[\langle a\rangle \mid \langle b\rangle \mid c,d])}_{\text{for transaction } t} + \underbrace{4 + \mathtt{pb}(P_2) + \mathtt{Z}(s[\langle a\rangle \mid \langle b\rangle \mid \langle d\rangle, \mathbf{0}])}_{\text{for transaction } s} \\
= & 4 + 2 + 1 + 4 + 3 + 1 = 15.
\end{aligned}$$

We briefly analyze the reduction steps that are related to the translation of transactions on name t and s:

- For location t there are 7 reduction steps: synchronization on t and $\bar{t}$, updating location t, two steps as relocation of process on location $p_{t,s}$ by process $\mathtt{out}^\circ$ on location p_s, update on location z_t, erasing location t using $t\{\dagger\}$ and synchronization h_t with corresponding output $\overline{h_t}$.
- For location s there are 8 reduction steps, one more step than for location t; because now location s contains three processes on location p_s that have to be relocated on location p_ε by using process $\mathtt{out}^\circ$.

Example 6.7. Recall process *Reservation* from the hotel booking scenario (§ 2, Example 2.1, page 4). We use encoding with objective update:

$$\begin{array}{rl}
[\![Reservation]\!]^\circ_\varepsilon = & t\big[book.pay.invoice\big] \mid t.(\mathtt{extr}\{t, p_t, p_\varepsilon\} \mid p_\varepsilon[\overline{refund}]) \mid \overline{book}.\overline{pay}.\bar{t}.h_t.refund \\
\longrightarrow^3 & t\big[invoice\big] \mid t\{(Y).t[Y] \mid \mathsf{ch}(t,Y) \\
& \mid \mathtt{out}^\circ(t, p_t\,,\, p_\varepsilon\,,\, \mathtt{nl}(p_t, Y)\,,\, t\{\dagger\}.\overline{h_t})\} \mid p_\varepsilon[\overline{refund}] \mid h_t.refund \\
\longrightarrow & t\big[invoice\big] \mid \mathsf{ch}(t, invoice) \mid \mathtt{out}^\circ(t, p_t\,,\, p_\varepsilon\,,\, \mathtt{nl}(p_t, invoice)\,,\, t\{\dagger\}.\overline{h_t}) \\
& \mid p_\varepsilon[\overline{refund}] \mid h_t.refund \\
\equiv & t\big[invoice\big] \mid \mathtt{out}^\circ(t, p_t\,,\, p_\varepsilon\,,\, 0\,,\, t\{\dagger\}.\overline{h_t}) \mid p_\varepsilon[\overline{refund}] \mid h_t.refund \\
= & t\big[invoice\big] \mid t\{\dagger\}.\overline{h_t} \mid p_\varepsilon[\overline{refund}] \mid h_t.refund \\
\longrightarrow^3 & p_\varepsilon[\mathbf{0}]
\end{array}$$

Therefore, we get $[\![Reservation]\!]^\circ_\varepsilon \longrightarrow^7 p_\varepsilon[\mathbf{0}]$ and so the number and justification for the obtained reduction steps is the same as in Example 5.10. This is because the transaction t does not contain protected blocks. In turn, in our encodings, this means that there are no differences between processes $\mathtt{out}^s$ and $\mathtt{out}^\circ$, i.e., they are equal to some process Q. In this example, $Q = t\langle\!\langle\dagger\rangle\!\rangle.\overline{h_t}$.

Divergence reflection and success sensitiveness. We close our analysis of correctness for the translation $[\![\cdot]\!]^{\circ}_{\rho}$ by considering divergence reflection and success sensitiveness. We state the corresponding results; their proofs proceed similarly as for Theorem 5.12 and Theorem 5.13, respectively.

Theorem 6.8 *(Divergence reflection for $[\![\cdot]\!]^{\circ}_{\rho}$). Let $\{R_i\}_{i\geq 0}$ be an infinite sequence of adaptable processes such that*

$$(1)\ R_0 = [\![P_0]\!]^{\circ}_{\rho} \text{ for some } P_0 \text{ and } \rho, \qquad \text{and} \qquad (2)\ \ R_i \longrightarrow R_{i+1} \text{ for any } i \geq 0.$$

Then there is an infinite sequence of adaptable processes $\{P'_j\}_{j\geq 0}$ such that

$$(3)\ P'_0 = P_0, \qquad \text{and} \qquad (4)\ \ P'_j \longrightarrow P'_{j+1} \text{ for any } j \geq 0.$$

Theorem 6.9 *(Success sensitiveness for $[\![\cdot]\!]^{\circ}_{\rho}$). Let P be a well-formed compensable process and ρ is an arbitrary path. Then $P \Downarrow$ if and only if $[\![P]\!]^{\circ}_{\rho} \Downarrow$.*

7. Comparing subjective vs objective updates

Having introduced two encodings of compensable processes into adaptable processes, here we compare their *efficiency*. We define efficiency in abstract terms, considering the number of reduction steps that a target language requires to mimic the behavior of a source language:

Definition 7.1 *(Efficient encoding).* Let $\mathcal{L}_i = (\mathcal{P}_i, \longrightarrow_i, \approx_i)$ (with $i \in \{1, 2, 3\}$) be calculi as in Definition 4.1. Suppose $[\![\cdot]\!]_1 : \mathcal{P}_1 \longrightarrow \mathcal{P}_2$ and $[\![\cdot]\!]_2 : \mathcal{P}_1 \longrightarrow \mathcal{P}_3$ are encodings as in Definition 4.5. We say that $[\![\cdot]\!]_1$ is *as or more efficient* than $[\![\cdot]\!]_2$ if for every process P from $\mathcal{P}_1$ the following implication holds (with $k_1, k_2 > 0$):

$$\text{If } P \longrightarrow P' \text{ and } [\![P]\!]_1 \longrightarrow^{k_1} [\![P']\!]_1 \text{ and } [\![P]\!]_2 \longrightarrow^{k_2} [\![P']\!]_2 \text{ then } k_1 \leq k_2.$$

We then have the following theorem:

Theorem 7.1. *The encoding $[\![\cdot]\!]_{\rho} : \mathcal{C} \longrightarrow \mathcal{S}$ is as or more efficient than $[\![\cdot]\!]^{\circ}_{\rho} : \mathcal{C} \longrightarrow \mathcal{O}$.*

Proof. The proof follows directly the operational correspondence results for each encoding, given in Theorem 5.8 and Theorem 6.5. Let P be a well-formed compensable process such that $P \longrightarrow P'$ and $[\![P]\!]_{\rho} \longrightarrow^{k_1} [\![P']\!]_{\rho}$ and $[\![P]\!]^{\circ}_{\rho} \longrightarrow^{k_2} [\![P']\!]_{\rho}$. Based on Proposition 3.1, we consider the following three cases, for some contexts C, D, E, processes P_1, Q, P_2 and names t, u:

a) Case $P \equiv E[C[\overline{a}.P_1] \mid D[a.P_2]]$: Here Theorem 5.8 and Theorem 6.5 ensure that $k_1 = k_2 = 1$. Thus, update prefixes make no difference when encoding usual input-output synchronizations.
b) Case $P \equiv E[C[t[P_1, Q]] \mid D[\overline{t}.P_2]]$: In this case, Theorem 5.8 ensures $k_1 = 4 + \mathtt{pb}(P_1)$, while Theorem 6.5 ensures $k_2 = 4 + \mathtt{pb}(P_1) + \mathtt{z}(P_1)$.
c) Case $P \equiv C[u[D[\overline{u}.P_1], Q]]$: Here Theorem 5.8 ensures $k_1 = 4 + \mathtt{pb}(D[P_1])$ and Theorem 6.5 ensures $k_2 = 4 + \mathtt{pb}(D[P_1]) + \mathtt{z}(D[P_1])$.

Thus, in all three cases $k_1 \leq k_2$; by Definition 7.1 we conclude that $[\![\cdot]\!]_{\rho}$ is as or more efficient than $[\![\cdot]\!]^{\circ}_{\rho}$. □

Let us dwell a bit on the content of the previous theorem, to understand better the differences between the two encodings (and between objective and subjective update). Recall that the main difference between our encodings is in the auxiliary processes

$$\mathtt{extr}\langle\!\langle t, l_1, l_2\rangle\!\rangle \quad \text{and} \quad \mathtt{extr}\{t, l_1, l_2\}$$

which are used in the encoding of transaction scopes in the subjective and objective case, respectively. In turn, those auxiliary processes rely on processes $\mathtt{out}^{\mathtt{s}}(l_1, l_2, n, Q)$ and $\mathtt{out}^{\circ}(t, l_1, l_2, n, Q)$ (cf. (12) and (15), respectively), which extract n processes located at l_1 in Q and relocate them to l_2.

A closer look at $\mathtt{out}^{\mathtt{s}}(l_1, l_2, n, Q)$ and $\mathtt{out}^{\circ}(t, l_1, l_2, n, Q)$ reveals that they differ in the use of name z, which is used in the objective case (when $n > 0$) but not in the subjective case. The use of z_t appears indispensable: under a semantics with objective update, after n updates, the located processes will stay at the wrong location (i.e. t). To avoid this, we use z_t as an auxiliary location. This auxiliary location enables us to move processes out of t and to relocate them to their parent location.

This synchronization step on name z_t is the key to the efficiency gains obtained when moving from objective to subjective updates—clearly, the improvement will be proportional to the number of compensation operations in the source process. Consider again Example 5.9 and Example 6.6, which translate process $P = s\big[t[\langle a\rangle \mid \langle b\rangle \mid c,d],\mathbf{0}\big] \mid \bar{t}.\bar{s}$ using subjective and objective updates, respectively. Here the subjective encoding outperforms the objective encoding by two reduction steps. In Example 6.6, these two steps correspond to two synchronizations on name z_t, which are not needed in Example 5.9.

Finally, as the proof of Theorem 7.1 makes explicit, the two encodings are equivalent, in terms of efficiency, when $n = 0$ in $\mathtt{out}^{\mathtt{s}}(l_1, l_2, n, Q)$ and $\mathtt{out}^{\mathtt{o}}(t, l_1, l_2, n, Q)$. This is because in this case we do not need to save any process from the default activity of the transaction scope—we need an equal number of reduction steps for achieving operational correspondence.

8. Extensions

In this section we discuss extensions to the source calculus of compensable processes that we have considered here, and how our encoding into (subjective) adaptable processes can account for such extensions.

8.1. Extensions to compensable processes

In the paper [15], Lanese et al. present different variants of the calculus of compensable processes:

- compensations may admit *static* or *dynamic* recovery (i.e., recovery without/with compensation updates); and
- nested transactions and protected blocks can be kept after failures via *discarding*, *preserving*, and *aborting* semantics.

Up to here, we have considered compensable processes with discarding semantics (§ 3.1). We now discuss and explain the main differences with respect to the other alternatives.

Static compensations. To motivate the differences between discarding semantics and preserving and aborting semantics, we consider their corresponding extraction function. The extraction function for preserving and aborting semantics (denoted $\mathsf{extr}_{\mathtt{P}}(\cdot)$ and $\mathsf{extr}_{\mathtt{A}}(\cdot)$, respectively) are different from the function for discarding semantics (cf. Fig. 1) only when the process P is a transaction scope:

- $\mathsf{extr}_{\mathtt{P}}(P)$ keeps protected blocks and transactions at the top-level in P. Other processes are discarded.

$$\mathsf{extr}_{\mathtt{P}}(t[P_1, Q]) = t[P_1, Q] \tag{18}$$

- $\mathsf{extr}_{\mathtt{A}}(P)$ keeps all protected blocks in P, including protected blocks from all nested transactions in P and their respective compensation activities. Other processes are discarded.

$$\mathsf{extr}_{\mathtt{A}}(t[P_1, Q]) = \mathsf{extr}_{\mathtt{A}}(P_1) \mid \langle Q\rangle \tag{19}$$

This way, discarding, preserving, and aborting semantics define different levels of protection for protected blocks. As an example, consider the process $P = t\big[t_1[P_1, Q_1] \mid t_2[\langle P_2\rangle, Q_2] \mid \langle P_3\rangle, Q_5\big]$ (cf. (4)), where process P_1 does not contain protected blocks and transaction scopes. Writing $\xrightarrow{\tau}_{\mathtt{D}}$, $\xrightarrow{\tau}_{\mathtt{P}}$, and $\xrightarrow{\tau}_{\mathtt{A}}$ to denote the LTSs induced by the different extraction functions, we would have:

$$\begin{array}{lll}
\text{Discarding}: & \bar{t} \mid P \xrightarrow{\tau}_{\mathtt{D}} \langle P_3\rangle \mid \langle Q_5\rangle \\
\text{Preserving}: & \bar{t} \mid P \xrightarrow{\tau}_{\mathtt{P}} \langle P_3\rangle \mid t_1[P_1, Q_1] \mid t_2[\langle P_2\rangle, Q_2] \mid \langle Q_5\rangle \\
\text{Aborting}: & \bar{t} \mid P \xrightarrow{\tau}_{\mathtt{A}} \langle P_3\rangle \mid \langle P_2\rangle \mid \langle Q_1\rangle \mid \langle Q_2\rangle \mid \langle Q_5\rangle
\end{array}$$

Unlike the discarding semantics, the preserving semantics protects also the nested transactions t_1 and t_2. The aborting semantics preserves all protected blocks and compensation activities in the default activity for t, including those in nested transactions, such as $\langle P_2\rangle$. Therefore, aborting semantics preserves more behaviors than discarding semantics (including protected blocks in nested transactions), while preserving semantics has the highest level of protection.

Dynamic compensations. Compensations can be dynamic rather than static, in the following sense: given a transaction $t[P, Q]$, process P can use *compensation updates* to specify an update for the compensation behavior Q. This is achieved by the operator $\mathtt{inst}\lfloor\lambda X.Q\rfloor.P$, where $\lambda X.Q$ is a function (with parameter X) that represents the compensation update. As a simple example, consider the following transition:

$$t\big[\mathtt{inst}\lfloor\lambda X.R\rfloor.P_1, Q\big] \xrightarrow{\tau} t\big[P_1, R\{^{Q}/_{X}\}\big] \tag{20}$$

This way, $\mathtt{inst}\lfloor\lambda X.R\rfloor.P$ produces a new compensation behavior $R\{^{Q}/_{X}\}$ after an internal transition. As variable X may not occur in R, this step may fully discard the previous compensation activity Q. A compensation update has priority over other transitions; that is, if process P in transaction $t[P, Q]$ has a compensation update at top-level then it will be performed before any change of the current state.

Definition 8.1. Building upon the syntax in § 3.1, we shall write $\mathcal{C}_{\mathtt{P}}$ and $\mathcal{C}_{\mathtt{A}}$ o denote compensable processes with preserving and aborting semantics, respectively. Also, we write $\mathcal{C}^{\lambda}$ to denote compensable processes with compensation updates.

Remark 8.1. The notion of *well-formed* compensable processes (cf. Section 3.2) extends to $\mathcal{C}_{\mathtt{P}}$ and $\mathcal{C}_{\mathtt{A}}$ as expected. For $\mathcal{C}^{\lambda}$, the definition of well-formed processes must account for compensation updates (cf. Remark 8.8). Our translations of $\mathcal{C}_{\mathtt{P}}$, $\mathcal{C}_{\mathtt{A}}$ and $\mathcal{C}^{\lambda}$ into $\mathcal{S}$ will be defined for well-formed compensable processes.

8.2. Extensions to our (efficient) encoding

We discuss key modifications required to encode $\mathcal{C}_{\mathtt{P}}$, $\mathcal{C}_{\mathtt{A}}$ and $\mathcal{C}^{\lambda}$ into $\mathcal{S}$. We focus on $\mathcal{S}$ as target language, as we have already seen that this language induces encodings that are more efficient than encodings into $\mathcal{O}$. In all cases, we focus on highlighting the modifications required to define the translations, omitting details on their associated correctness properties.

8.2.1. Translating $\mathcal{C}_{\mathtt{P}}$ into $\mathcal{S}$

The translation $\mathcal{C}_{\mathtt{P}}$ into $\mathcal{S}$, denoted $(\!|\cdot|\!)_{\rho}$, uses very similar ideas as the encoding $[\![\cdot]\!]_{\rho}$. This way, the translation of a protected block found at path ρ, is defined as:

$$(\!|\langle P\rangle|\!)_{\rho} = p_{\rho}\big[(\!|P|\!)_{\varepsilon}\big].$$

To encode a preserving semantics we extend the base sets given in Definition 5.1 as follows:

- $\mathcal{N}_s^r = \{h_x, j_x, r_x \mid x \in \mathcal{N}_t\}$ is the set of *reserved synchronization names*;
- $\mathcal{N}_l^r = \{p_\rho, \beta_\rho \mid \rho \text{ is a path}\}$ is the set of *reserved location names*.

We extend the set of *reserved location names* with name β_ρ, because besides protected blocks we have to keep transactions that are in default activity P (cf. (18)) in the case that a failure signal exists. We use a revised auxiliary process, denoted $\mathtt{outp}^{\mathtt{s}}(t, l_1, l_1', l_2, l_2', n, m)$, which (i) moves n processes from location l_1 to location l_1'; (ii) moves m processes from location l_2 to location l_2'. To define process $\mathtt{outp}^{\mathtt{s}}$, we need some auxiliary notions. In the case, when we move m processes from location l_2 to location l_2' it will be necessary to remove some names from the path in processes that are enclosed in l_2'. The following function removes a name from a path:

Definition 8.2. Let $\rho = t_1, \ldots, t_n$ be a path and r be a name in $\mathcal{N}_t$. We define the function ρ/r as follows:

$$\rho/r = \begin{cases} t_1, t_2, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n & \text{if } t_i = r \\ \\ \rho & \text{if } t_i \neq r \text{ and } 1 \leq i \leq n. \end{cases}$$

It should be noted that name r can occur only one time in ρ (cf. Definition 3.4 (i)).

The following definition serves to remove names mentioned in an adaptable process. This is important: if a transaction t had nested transactions and location name t is lost, then we have to remove t from all the paths that contained it.

Definition 8.3. Let P be an adaptable process, and let ρ be a path that contains name s. The function $\mathcal{E}(P, s)$ is defined as follows:

$$\mathcal{E}(P, s) = \begin{cases} l[\mathcal{E}(P_1, s)] & \text{if } P = l[P_1] \text{ and } l \notin \mathcal{N}_l^r \\ p_{\rho/s}[\mathcal{E}(P_1, s)] & \text{if } P = p_\rho[P_1] \\ \beta_{\rho/s}[\mathcal{E}(P_1, s)] & \text{if } P = \beta_\rho[P_1] \\ \mathcal{E}(P_1, s) \mid \mathcal{E}(P_2, s) & \text{if } P = P_1 \mid P_2 \\ \pi.\mathcal{E}(P_1, s) & \text{if } P = \pi.P_1 \\ !\pi.\mathcal{E}(P_1, s) & \text{if } P = !\pi.P_1 \\ (\nu x)\mathcal{E}(P_1, s) & \text{if } P = (\nu x)P_1 \\ \mathbf{0} & \text{if } P = \mathbf{0} \\ X & \text{if } P = X. \end{cases}$$

Definition 8.4. Let l be a name and P an adaptable process. Function $\mathtt{top}(l, P)$ denotes the list of location names from P that are nested (at top level) in l. It is defined as follows:

$$\mathtt{top}(l, l'[P]) = \begin{cases} \{l''\} & \text{if } l' = l \text{ and } P \equiv l''[Q] \mid R \text{ for some } Q, R \text{ and } l'' \in \mathcal{N}_t \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathtt{top}(l, P \mid Q) = \mathtt{top}(l, P) \cup \mathtt{top}(l, Q) \qquad \mathtt{top}(l, \mathbf{0}) = \mathtt{top}(l, X) = \emptyset$$

$$\mathtt{top}(l, (\nu x)P) = \mathtt{top}(l, P) \qquad \mathtt{top}(l, \pi.P) = \mathtt{top}(l, !\pi.P) = \emptyset$$

$$\begin{array}{lll} \mathtt{ts}_{\mathtt{P}}(t[P,Q]) = 1 & \mathtt{ts}_{\mathtt{A}}(t[P,Q]) = 1 + \mathtt{ts}_{\mathtt{A}}(P) & \mathtt{ts}(\pi.P) = \mathtt{ts}(\langle P \rangle) = 0 \\ \mathtt{ts}(P \mid Q) = \mathtt{ts}(P) + \mathtt{ts}(Q) & \mathtt{ts}((\nu x)P) = \mathtt{ts}(P) & \mathtt{ts}(!\pi.P) = \mathtt{ts}(\mathbf{0}) = 0 \end{array}$$

Fig. 11. Number of transactions.

For the definition of $\mathtt{outp}^{\mathtt{s}}(t, P, l_1, l_1', l_2, l_2', n, m)$ we introduce the following auxiliary processes:

$$\mathtt{outp}_1^{\mathtt{s}}(t, l_1, l_1', n) = l_1 \langle\!\langle (X_1, \ldots, X_n).\left(\prod_{i=1}^{n} l_1'[X_i] \mid t\langle\!\langle \dagger \rangle\!\rangle.\overline{j_t}.r_t\right)\rangle\!\rangle;$$

$$\mathtt{outp}_2^{\mathtt{s}}(t, t_1, \ldots, t_m, l_2, l_2', m) = l_2 \langle\!\langle (Y_1, \ldots, Y_m).\left(r_t.\left(\prod_{k=1}^{m}\left(l_2'[\mathcal{E}(Y_k, t)] \mid j_{t_k}.l_2'\langle\!\langle (X).X \rangle\!\rangle.\overline{r_{t_k}}.\overline{h_{t_k}}\right)\right) \mid t\langle\!\langle \dagger \rangle\!\rangle.\overline{j_t}\right)\rangle\!\rangle;$$

$$\begin{aligned} &\mathtt{outp}_3^{\mathtt{s}}(t, t_1, \ldots, t_m, l_1, l_1', l_2, l_2', n, m) = l_1 \langle\!\langle (X_1, \ldots, X_n).l_2 \langle\!\langle (Y_1, \ldots, Y_m). \\ &\quad \left(\prod_{i=1}^{n} l_1'[X_i] \mid r_t.\left(\prod_{k=1}^{m}\left(l_2'[\mathcal{E}(Y_k, t)] \mid j_{t_k}.l_2'\langle\!\langle (X).X \rangle\!\rangle.\overline{r_{t_k}}.\overline{h_{t_k}}\right)\right) \mid t\langle\!\langle \dagger \rangle\!\rangle.\overline{j_t}\right)\rangle\!\rangle\rangle\!\rangle. \end{aligned}$$

The auxiliary process $\mathtt{outp}^{\mathtt{s}}(t, P, l_1, l_1', l_2, l_2', n, m)$, where $\mathtt{top}(l_2, P) = \{t_1, \ldots, t_m\}$ for $m > 0$,is defined as follows:

$$\mathtt{outp}^{\mathtt{s}}(t, P, l_1, l_1', l_2, l_2', n, m) = \begin{cases} t\langle\!\langle \dagger \rangle\!\rangle.\overline{j_t}.r_t & \text{if } n, m = 0 \\ \mathtt{outp}_1^{\mathtt{s}}(t, l_1, l_1', n) & \text{if } n > 0, m = 0 \\ \mathtt{outp}_2^{\mathtt{s}}(t, t_1, \ldots, t_m, l_2, l_2', m) & \text{if } n = 0, m > 0 \\ \mathtt{outp}_3^{\mathtt{s}}(t, t_1, \ldots, t_m, l_1, l_1', l_2, l_2', n, m) & \text{if } n, m > 0 \end{cases} \tag{21}$$

The following example illustrates process $\mathtt{outp}^{\mathtt{s}}$ (cf. (21)). A more detailed explanation is given later on.

Example 8.2. We illustrate the revised auxiliary process:

$$\begin{aligned} & s\Big[t\big[l_1[c] \mid l_1[d] \mid e\big] \mid \mathtt{outp}_1^{\mathtt{s}}(t, l_1, l_1', 2)\Big] \\ = \;& s\Big[t\big[l_1[c] \mid l_1[d] \mid e\big] \mid l_1 \langle\!\langle (X_1, X_2).\big(l_1'[X_1] \mid l_1'[X_2] \mid t\langle\!\langle \dagger \rangle\!\rangle.\overline{j_t}.r_t\big)\rangle\!\rangle\Big] \\ \longrightarrow \;& s\Big[t\big[l_1[d] \mid e\big] \mid l_1 \langle\!\langle (X_2).\big(l_1'[c] \mid l_1'[X_2] \mid t\langle\!\langle \dagger \rangle\!\rangle.\overline{j_t}.r_t\big)\rangle\!\rangle\Big] \\ \longrightarrow \;& s\Big[t\big[e\big] \mid l_1'[c] \mid l_1'[d] \mid t\langle\!\langle \dagger \rangle\!\rangle.\overline{j_t}.r_t\Big] \\ \longrightarrow \;& s\Big[l_1'[c] \mid l_1'[d] \mid \overline{j_t}.r_t\Big] \end{aligned}$$

Above, the two reduction steps are used for relocation of $l_1[c]$ and $l_1[d]$ that are nested in location t (with omitted trailing occurrences of $\mathbf{0}$). The third step is the synchronization between update prefix $t\langle\!\langle \dagger \rangle\!\rangle$ and location $t[e]$, where the update deletes the location and its content.

In order to give a precise account of the number of computation steps used by our translation, we use $\mathtt{pb}(P)$ (as before) but also one additional notion. Given a compensable process P, we will write $\mathtt{ts}_{\mathtt{P}}(P)$ and $\mathtt{ts}_{\mathtt{A}}(P)$ to denote the number of transactions in P for preserving and aborting semantics, respectively. Whenever a notion coincides for the both semantics, we shall avoid decorations $\mathtt{P}$ and $\mathtt{A}$. The function $\mathtt{ts}(\cdot)$ is presented in Fig. 11. It should be noted that the number of protected blocks and transactions in the default activity of the transaction scope correspond to the number of locations $p_{t,\rho}$ and $\beta_{t,\rho}$ after the encoding of protected blocks and transactions in this transaction.

The last ingredient we need to translate $\mathcal{C}_{\mathtt{P}}$ into $\mathcal{S}$ is the following auxiliary process.

Definition 8.5 *(Update prefix for extraction).* Let t, l_1, l_2 be names and P is an adaptable process. We write $\mathtt{extrp}\langle\!\langle t, P, l_1, l_1', l_2, l_2' \rangle\!\rangle$ to stand for the following (subjective) update prefix:

$$\mathtt{extrp}\langle\!\langle t, P, l_1, l_1', l_2, l_2' \rangle\!\rangle = t\langle\!\langle (Y).t[Y] \mid \mathsf{ch}(t, Y) \mid \mathtt{outp}^{\mathtt{s}}(t, P, l_1, l_1', l_2, l_2', \mathtt{nl}(l_1, Y), \mathtt{nl}(l_2, Y))\rangle\!\rangle \tag{22}$$

Now, we may formally define $(\!|\cdot|\!)_\rho$:

Definition 8.6 *(Translating $\mathcal{C}_{\mathtt{P}}$ into $\mathcal{S}$).* Let ρ be a path. We define the translation of compensable processes with preserving semantics into (subjective) adaptable processes as a tuple $((\!|\cdot|\!)_\rho, \varphi_{(\!|\cdot|\!)_\rho})$ where:

$$\begin{aligned}
\llparenthesis \langle P\rangle \rrparenthesis_\rho &= p_\rho\big[\llparenthesis P\rrparenthesis_\varepsilon\big]\\
\llparenthesis t[P,Q]\rrparenthesis_\rho &= \beta_\rho\Big[t\big[\llparenthesis P\rrparenthesis_{t,\rho}\big] \mid t.\big(\mathtt{extrp}\langle\!\langle t, \llparenthesis P\rrparenthesis_{t,\rho}, p_{t,\rho}, p_\rho, \beta_{t,\rho}, \beta_\rho\rangle\!\rangle \mid p_\rho[\llparenthesis Q\rrparenthesis_\varepsilon]\big)\Big] \mid j_t.\beta_\rho\langle\!\langle (X).X\rangle\!\rangle.\overline{r}_t.\overline{h_t}\\
\llparenthesis a.P\rrparenthesis_\rho &= a.\llparenthesis P\rrparenthesis_\rho\\
\llparenthesis \overline{a}.P\rrparenthesis_\rho &= \overline{a}.\llparenthesis P\rrparenthesis_\rho\\
\llparenthesis \overline{t}.P\rrparenthesis_\rho &= \overline{t}.h_t.\llparenthesis P\rrparenthesis_\rho
\end{aligned}$$

Fig. 12. Translating $\mathcal{C}_{\mathtt{P}}$ into $\mathcal{S}$.

$\mathtt{pb}_{\mathtt{A}}(\langle P\rangle) = \mathtt{pb}_{\mathtt{P}}(\langle P\rangle) = 1$	$\mathtt{pb}_{\mathtt{P}}(t[P,Q]) = 0$	$\mathtt{pb}_{\mathtt{A}}(t[P,Q]) = 1 + \mathtt{pb}_{\mathtt{A}}(P)$
$\mathtt{pb}(\mathtt{inst}\lfloor\lambda X.Q\rfloor.P) = 0$	$\mathtt{pb}_{\mathtt{P}}(\mathtt{inst}\lfloor\lambda X.Q\rfloor.P) = 0$	$\mathtt{pb}_{\mathtt{A}}(\mathtt{inst}\lfloor\lambda X.Q\rfloor.P) = 0$

Fig. 13. Number of protected blocks for preserving and aborting semantics and dynamic recovery.

(a) The renaming policy $\varphi_{\llparenthesis\cdot\rrparenthesis_\rho} : \mathcal{N}_c \longrightarrow \mathcal{P}(\mathcal{N}_a)$ is defined with

$$\varphi_{\llparenthesis\cdot\rrparenthesis_\rho}(x) = \begin{cases}\{x\} & \text{if } x \in \mathcal{N}_s\\ \{x, h_x, j_x, r_x\} \cup \{p_\rho, \beta_\rho : x \in \rho\} & \text{if } x \in \mathcal{N}_t\end{cases} \tag{23}$$

(b) The translation $\llparenthesis\cdot\rrparenthesis_\rho : \mathcal{C}_{\mathtt{P}} \longrightarrow \mathcal{S}$ is as in Fig. 12 and as a homomorphism for other operators.

Consider the translation of $t[P,Q]$: as in the encoding $[\![\cdot]\!]_\rho$ (cf. Fig. 6), the structure of a transaction and the number of its top-level processes dynamically changes if there is a failure signal; whenever we need to extract processes located at $p_{t,\rho}$ and $\beta_{t,\rho}$ we will first substitute Y in process $\mathtt{outp}^{\mathtt{s}}$ (cf. (21)) by the content of the location t and count the current number of locations $p_{t,\rho}$ and $\beta_{t,\rho}$. The translation of the transaction body P with location t is nested in location β_ρ, and the compensation activity Q is encoded as a protected block and nested in location p_ρ. If P contains n top-level protected blocks and m top-level transaction scopes (with $n, m > 0$) when the failure signal $\overline{t}$ is activated, after synchronizations on t and updates, the translation will release $n+m$ successive update prefixes by using auxiliary processes $\mathtt{outp}^{\mathtt{s}}$. Indeed, thanks to processes $\mathtt{outp}^{\mathtt{s}}$, n protected blocks at location $p_{t,\rho}$ and m transaction scopes at location $\beta_{t,\rho}$ will be moved to their parent locations (p_ρ and β_ρ, respectively). Subsequently, there is a synchronization on location t that discards it with its content. After that, there are synchronizations on names j_t, β_ρ, r_t, and h_t. As we explained before, we use function $\mathcal{E}(P,s)$ (cf. Definition 8.3) when we have to take the name s out from all the paths that contain it.

The following example illustrates the translation.

Example 8.3. Notably, $P = s\big[t[v[b,\mathbf{0}],c] \mid \langle d\rangle, \mathbf{0}\big] \mid \overline{t}.\overline{s}$ is a well-formed compensable process. By the LTS (cf. Fig. 2), we have

$$P \xrightarrow{\tau}_{\mathtt{P}} s[v[b,\mathbf{0}] \mid \langle c\rangle \mid \langle d\rangle, \mathbf{0}] \mid \overline{s} \xrightarrow{\tau}_{\mathtt{P}} v[b,\mathbf{0}] \mid \langle c\rangle \mid \langle d\rangle.$$

We have that $\mathtt{pb}_{\mathtt{P}}(P) = 0$ and $\mathtt{ts}_{\mathtt{P}}(P) = 1$. Let $P_1 = t[v[b,\mathbf{0}],c]$: by Fig. 13 it follows that $\mathtt{pb}_{\mathtt{P}}(P_1) = 0$; by Fig. 11 that $\mathtt{ts}_{\mathtt{P}}(P_1) = 1$. We have a sequential error notification such that activation starts from nested transactions on name t. By expanding Definition 8.6, we have the following translation and derivation:

$$\begin{aligned}
\llparenthesis P\rrparenthesis_\varepsilon = \beta_\varepsilon\Big[s\Big[\beta_s\Big[t\Big[\beta_{t,s}\Big[v\Big[b\Big] \mid v.\big(\mathtt{extrp}\langle\!\langle v, \llparenthesis b\rrparenthesis_{v,t,s}, p_{v,t,s}, p_{t,s}, \beta_{v,t,s}, \beta_{,t,s}\rangle\!\rangle\big)\Big] \mid j_v.\beta_{t,s}\langle\!\langle (X).X\rangle\!\rangle.\overline{r_v}.\overline{h_v}\Big]\\
&\mid t.\big(\mathtt{extrp}\langle\!\langle t, \llparenthesis v[b,\mathbf{0}]\rrparenthesis_{t,s}, p_{t,s}, p_s, \beta_{t,s}, \beta_s\rangle\!\rangle \mid p_s\big[c\big]\big)\Big] \mid j_t.\beta_s\langle\!\langle (X).X\rangle\!\rangle.\overline{r_t}.\overline{h_t} \mid p_s\big[d\big]\Big]\\
&\mid s.\big(\mathtt{extrp}\langle\!\langle s, \llparenthesis P_1 \mid \langle d\rangle\rrparenthesis_s, p_s, p_\varepsilon, \beta_s, \beta_\varepsilon\rangle\!\rangle\big)\Big] \mid j_s.\beta_\varepsilon\langle\!\langle (X).X\rangle\!\rangle.\overline{r_s}.\overline{h_s} \mid \overline{t}.h_t.\overline{s}.h_s\\
&= \beta_\varepsilon\Big[s\Big[\beta_s\Big[t\Big[\beta_{t,s}\Big[v\Big[b\Big] \mid v.\big(\mathtt{extrp}\langle\!\langle v, \llparenthesis b\rrparenthesis_{v,t,s}, p_{v,t,s}, p_{t,s}, \beta_{v,t,s}, \beta_{,t,s}\rangle\!\rangle\big)\Big] \mid j_v.\beta_{t,s}\langle\!\langle (X).X\rangle\!\rangle.\overline{r_v}.\overline{h_v}\Big]\\
&\mid t.\big(t\langle\!\langle (Y).t[Y] \mid \mathsf{ch}(t,Y) \mid \mathtt{outp}^{\mathtt{s}}(t, \llparenthesis v[b,\mathbf{0}]\rrparenthesis_{t,s}, p_{t,s}, p_s, \beta_{t,s}, \beta_s, \mathtt{nl}(p_{t,s},Y), \mathtt{nl}(\beta_{t,s},Y))\rangle\!\rangle \mid p_s\big[c\big]\big)\Big]\\
&\mid j_t.\beta_s\langle\!\langle (X).X\rangle\!\rangle.\overline{r_t}.\overline{h_t} \mid p_s\big[d\big]\Big]\\
&\mid s.\big(\mathtt{extrp}\langle\!\langle s, \llparenthesis P_1 \mid \langle d\rangle\rrparenthesis_s, p_s, p_\varepsilon, \beta_s, \beta_\varepsilon\rangle\!\rangle\big)\Big] \mid j_s.\beta_\varepsilon\langle\!\langle (X).X\rangle\!\rangle.\overline{r_s}.\overline{h_s} \mid \overline{t}.h_t.\overline{s}.h_s\\
&\longrightarrow^8 \beta_\varepsilon\Big[s\Big[\beta_s\Big[v\Big[b\Big] \mid v.\big(\mathtt{extrp}\langle\!\langle v, \llparenthesis b\rrparenthesis_{v,s}, p_{v,s}, p_s, \beta_{v,s}, \beta_s\rangle\!\rangle\big)\Big] \mid j_v.\beta_s\langle\!\langle (X).X\rangle\!\rangle.\overline{r_v}.\overline{h_v} \mid p_s\big[c\big] \mid p_s\big[d\big]\Big]
\end{aligned}$$

$$
\begin{aligned}
&\mid s.\big(s\langle\!\langle (Y).t[Y] \mid \mathsf{ch}(s,Y) \mid \mathtt{outp^s}(s, (\!|P_1 \mid \langle d\rangle|\!)_s, p_s, p_\varepsilon, \beta_s, \beta_\varepsilon\,, \mathtt{nl}(p_s,Y)\,, \mathtt{nl}(\beta_s,Y))\rangle\!\rangle\big)\Big]\\
&\mid j_s.\beta_\varepsilon\langle\!\langle (X).X\rangle\!\rangle.\overline{r_s}.\overline{h_s} \mid \overline{s}.h_s\\
\longrightarrow^9\ &\beta_\varepsilon\Big[v\Big[b\Big] \mid v.\big(\mathtt{extrp}\langle\!\langle v, (\!|b|\!)_v, p_v, p_\varepsilon, \beta_v, \beta_\varepsilon\rangle\!\rangle\big)\Big] \mid j_v.\beta_\varepsilon\langle\!\langle (X).X\rangle\!\rangle.\overline{r_v}.\overline{h_v} \mid p_\varepsilon\Big[c\Big] \mid p_\varepsilon\Big[d\Big]\\
=\ &(\!|v[b,\mathbf{0}] \mid \langle c\rangle \mid \langle d\rangle|\!)_\varepsilon.
\end{aligned}
$$

Therefore, the number of reduction steps is $k = 17$. Indeed, we have 8 reduction steps for location t and 9 reduction steps and for location s:

i) the first step is a synchronization on name t;

ii) now the process $\mathtt{extrp}\langle\!\langle t, (\!|v[b,\mathbf{0}]|\!)_{t,s}, p_{t,s}, p_s, \beta_{t,s}, \beta_s\rangle\!\rangle$ is released and the second step is synchronization on update prefix $t\langle\!\langle (Y).t[Y] \mid \mathsf{ch}(t,Y) \mid \mathtt{outp^s}(t, (\!|v[b,\mathbf{0}]|\!)_{t,s}, p_{t,s}, p_s, \beta_{t,s}, \beta_s\,, \mathtt{nl}(p_{t,s},Y)\,, \mathtt{nl}(\beta_{t,s},Y))\rangle\!\rangle$ and location $t\Big[\beta_{t,s}\Big[v\Big[b\Big] \mid v.\big(\mathtt{extrp}\langle\!\langle v, (\!|b|\!)_{v,t,s}, p_{v,t,s}, p_{t,s}, \beta_{v,t,s}, \beta_{,t,s}\rangle\!\rangle\big)\Big] \mid j_v.\beta_{t,s}\langle\!\langle (X).X\rangle\!\rangle.\overline{r_v}.\overline{h_v}\Big]$;

iii) as a result, process

$$\mathtt{outp^s}(t, (\!|v[b,\mathbf{0}]|\!)_{t,s}, p_{t,s}, p_s, \beta_{t,s}, \beta_s\,, 0\,, 1) = \beta_{t,s}\langle\!\langle (Y).\Big(r_t.\big(\beta_s[\mathcal{E}(Y,t)] \mid j_t.\beta_s\langle\!\langle (X).X\rangle\!\rangle.\overline{r_t}.\overline{h_t}\big) \mid t\langle\!\langle \dagger\rangle\!\rangle.\overline{j_t}\Big)\rangle\!\rangle$$

triggers the third step: the synchronization of location $\beta_{t,s}[\ldots]$ with update prefix $\beta_{t,s}\langle\!\langle (Y).\ldots\rangle\!\rangle$;

iv) the fourth step is the synchronization between update prefix $t\langle\!\langle \dagger\rangle\!\rangle$ and location $t[\ldots]$, where the update deletes the location and its content (cf. (11));

v) the fifth step is a synchronization on name j_t, which enables an update on $\beta_s\langle\!\langle (X).X\rangle\!\rangle$;

vi) the sixth step is a synchronization between $\beta_s\langle\!\langle (X).X\rangle\!\rangle$ and $\beta_s\Big[\ldots\Big]$, which deletes the location β_s;

vii) the seventh step is a synchronization on name r_t, which releases the process

$$\beta_s\Big[v\Big[b\Big] \mid v.\big(\mathtt{extrp}\langle\!\langle v, (\!|b|\!)_{v,s}, p_{v,s}, p_s, \beta_{v,s}, \beta_s\rangle\!\rangle\big)\Big] \mid j_t.\beta_s\langle\!\langle (X).X\rangle\!\rangle.\overline{r_t}.\overline{h_t};$$

viii) the eighth step is a synchronization on h_t, which activates visit to location on name s.

At this point, we have the same reduction steps but for location s. We have one more reduction step, though, since in process $\mathtt{outp^s}(s, (\!|P_1|\!)_s, p_s, p_\varepsilon, \beta_s, \beta_\varepsilon\,, 1\,, 1)$ we have a location $p_s[\ldots]$ that has to be relocated to $p_\varepsilon[\ldots]$. Consequently, we have 9 reduction steps for handling the location on name s.

We conclude by illustrating the translation on Example 2.1 (cf. page 4).

Example 8.4. We consider the hotel booking scenario where the client cancels a reservation after booking and paying. In compensable processes for preserving semantics we have that:

$$
\begin{aligned}
Reservation &\xrightarrow{\tau}_{\mathrm{P}} t[pay.\overline{invoice},\overline{refund}] \mid \overline{pay}.(invoice + \overline{t}.refund)\\
&\xrightarrow{\tau}_{\mathrm{P}} t[\overline{invoice},\overline{refund}] \mid invoice + \overline{t}.refund \xrightarrow{\tau}_{\mathrm{P}} \langle\overline{refund}\rangle \mid refund \xrightarrow{\tau}_{\mathrm{P}} \langle\mathbf{0}\rangle.
\end{aligned}
$$

We apply the translation (cf. Definition 8.6) on process $Reservation$:

$$
\begin{aligned}
(\!|Reservation|\!)_\varepsilon =\ &\beta_\varepsilon\Big[t\Big[book.pay.invoice.\Big] \mid t.\Big(\mathtt{extrp}\langle\!\langle t, book.pay.invoice., p_t, p_\varepsilon, \beta_t, \beta_\varepsilon\rangle\!\rangle \mid p_\varepsilon[\overline{refund}]\Big)\Big]\\
&\mid j_t.\beta_\varepsilon\langle\!\langle (X).X\rangle\!\rangle.\overline{r_t}.\overline{h_t} \mid \overline{book}.\overline{pay}.\overline{t}.h_t.refund\\
\longrightarrow^3\ &\beta_\varepsilon\Big[t\Big[invoice\Big] \mid t\langle\!\langle (Y).t[Y] \mid \mathsf{ch}(t,Y)\\
&\mid \mathtt{outp^s}(t, book.pay.invoice., p_t, p_\varepsilon, \beta_t, \beta_\varepsilon\,, \mathtt{nl}(p_t,Y)\,, \mathtt{nl}(\beta_t,Y))\rangle\!\rangle \mid p_\varepsilon[\overline{refund}]\Big]\\
&\mid j_t.\beta_\varepsilon\langle\!\langle (X).X\rangle\!\rangle.\overline{r_t}.\overline{h_t} \mid h_t.refund\\
\longrightarrow\ &\beta_\varepsilon\Big[t\Big[invoice\Big] \mid t\langle\!\langle \dagger\rangle\!\rangle.\overline{j_t}.r_t \mid p_\varepsilon[\overline{refund}]\Big] \mid j_t.\beta_\varepsilon\langle\!\langle (X).X\rangle\!\rangle.\overline{r_t}.\overline{h_t} \mid h_t.refund\\
\longrightarrow^5\ &p_\varepsilon[\overline{refund}] \mid refund\\
\longrightarrow\ &p_\varepsilon[\mathbf{0}]
\end{aligned}
$$

Therefore, $(\!|Reservation|\!)_\varepsilon \longrightarrow^{10} p_\varepsilon[\mathbf{0}]$. There are three reduction steps, denoted $\longrightarrow^3$, as a result of synchronizations on names $book$, pay, and t. Now, the structure of the default activity of transaction is changed and we have one reduction step for updating its current content. After that, there are six additional reduction steps: one for erasing location t and its content, a synchronization between j_t with $\overline{j_t}$, an update on location $\beta_\varepsilon\Big[\overline{j_t}.r_t \mid p_\varepsilon[\overline{refund}]\Big]$ with $\beta_\varepsilon\langle\!\langle (X).X\rangle\!\rangle$, and three reduction steps that result from synchronizations on names r_t, h_t, and $refund$.

$$
\begin{aligned}
\llparenthesis \langle P\rangle \rrparenthesis_\rho &= p_\rho\big[\llparenthesis P\rrparenthesis_\varepsilon\big]\\
\llparenthesis t[P,Q]\rrparenthesis_\rho &= t\big[\llparenthesis P\rrparenthesis_{t,\rho}\big] \mid r_t.\big(\mathtt{extra}\langle\langle t,\, p_{t,\rho},\, p_\rho\rangle\rangle \mid p_\rho[\llparenthesis Q\rrparenthesis_\varepsilon]\big) \mid t.t\langle\langle (Y).t[Y] \mid \mathcal{T}_t(Y).\overline{h_t}\rangle\rangle\\
\llparenthesis a.P\rrparenthesis_\rho &= a.\llparenthesis P\rrparenthesis_\rho\\
\llparenthesis \overline{a}.P\rrparenthesis_\rho &= \overline{a}.\llparenthesis P\rrparenthesis_\rho\\
\llparenthesis \overline{t}.P\rrparenthesis_\rho &= \overline{t}.h_t.\llparenthesis P\rrparenthesis_\rho
\end{aligned}
$$

Fig. 14. Translating $\mathcal{C}_{\mathtt{A}}$ nto $\mathcal{S}$.

8.2.2. *Translating $\mathcal{C}_{\mathtt{A}}$ into $\mathcal{S}$*

The translation $\mathcal{C}_{\mathtt{A}}$ into $\mathcal{S}$, denoted $\llparenthesis\cdot\rrparenthesis_\rho$, also relies on the key ideas of our encoding $[\![\cdot]\!]_\rho$. The translation of a protected block found at path ρ is defined as before:

$$\llparenthesis \langle P\rangle \rrparenthesis_\rho = p_\rho\big[\llparenthesis P\rrparenthesis_\varepsilon\big]$$

To translate aborting semantics for a transaction $t[P,Q]$ we use the base sets in Definition 5.1. The set of *reserved location names* is kept unchanged and the set of *reserved synchronization names* is extended such that $\mathcal{N}_s^r = \{h_x, k_x, r_x \mid x \in \mathcal{N}_t\}$. We need some additional auxiliary processes.

The aborting semantics keeps not only top-level protected blocks of a transaction, but also protected blocks from nested transactions (cf. (19)). To handle this, we define the *activation prefixes* of a process, which captures the hierarchical structure of its nested locations. Nested locations arise as a result of a translated transaction with its nested transactions. The activation prefixes contain the names of the nested locations. These names originate exclusively from its corresponding transaction name and the names of its nested transactions (i.e., locations on names p_ρ are not included in the activation prefixes).

Definition 8.7 *(Activation Prefixes).* Given a located process $l[P]$, we denote by $St(l[P])$ the *containment structure* of process $l[P]$: the labeled tree (with root l) in which nodes are labeled with names from $\mathcal{N}_t$ such that sub-trees capture nested locations. The *activation prefixes* for $l[P]$, denoted $\mathcal{T}_l(P)$, are obtained by a post-order search in $St(l[P])$ in which the visit to a node labeled l_i adds prefixes $\overline{r_{l_i}}.k_{l_i}$.

Example 8.5. Given $l[P]$ with $P = l_1[l_2[p_\rho[m_1]] \mid m_2] \mid l_3[m_3 \mid l_4[m_4] \mid l_5[m_5]]$, by Definition 8.7 we have the activation prefixes:

$$\mathcal{T}_l(P) = \overline{r_{l_2}}.k_{l_2}.\overline{r_{l_1}}.k_{l_1}.\overline{r_{l_4}}.k_{l_4}.\overline{r_{l_5}}.k_{l_5}.\overline{r_{l_3}}.k_{l_3}.\overline{r_l}.k_l$$

A failure signal extracts all nested protected blocks and erases nested locations; our translation does the same with the corresponding located processes and nested locations. We define the following auxiliary process:

Definition 8.8 *(Update prefix for extraction).* Let t, l_1, and l_2 be names. We write $\mathtt{extra}\langle\langle t,l_1,l_2\rangle\rangle$ to stand for the following (subjective) update prefix:

$$\mathtt{extra}\langle\langle t,l_1,l_2\rangle\rangle = t\langle\langle (Y).t[Y] \mid \mathsf{ch}(t,Y) \mid \mathtt{out}^{\mathtt{s}}(l_1,l_2\,,\,\mathtt{nl}(l,Y)\,,\,t\langle\langle \dagger\rangle\rangle.\overline{k_t})\rangle\rangle. \tag{24}$$

The translation $\llparenthesis\cdot\rrparenthesis_\rho$ is defined as follows:

Definition 8.9 *(Translation $\mathcal{C}_{\mathtt{A}}$ into $\mathcal{S}$).* Let ρ be a path. We define the translation of compensable processes with aborting semantics into (subjective) adaptable processes as a tuple $(\llparenthesis\cdot\rrparenthesis_\rho, \varphi_{\llparenthesis\cdot\rrparenthesis_\rho})$ where:

(a) The renaming policy $\varphi_{\llparenthesis\cdot\rrparenthesis_\rho} : \mathcal{N}_c \longrightarrow \mathcal{P}(\mathcal{N}_a)$ is defined with

$$\varphi_{\llparenthesis\cdot\rrparenthesis_\rho}(x) = \begin{cases} \{x\} & \text{if } x \in \mathcal{N}_s\\ \{x, h_x, k_x, r_x\} \cup \{p_\rho : x \in \rho\} & \text{if } x \in \mathcal{N}_t \end{cases}$$

(b) The translation $\llparenthesis\cdot\rrparenthesis_\rho : \mathcal{C} \longrightarrow \mathcal{S}$ is as in Fig. 14 and as a homomorphism for other operators.

Consider the translation of $t[P,Q]$: the presence of a failure signal dynamically changes the structure of a located process on transaction name (e.g. t) and the number of its nested processes. Therefore, we need first to substitute Y in activation prefixes $\mathcal{T}_t(Y)$ by the content of location t. For the same reason, whenever we need to extract processes located at $p_{t,\rho}$ we will substitute Y in process $\mathtt{out}^{\mathtt{s}}$ by the content of the location t. Also, we count the current number of locations $p_{t,\rho}$

using function $\mathtt{nl}(\cdot,\cdot)$ (cf. Definition 5.3). As in the translation for $\mathcal{C}$ into $\mathcal{S}$, we use the reserved name h_t to control the execution of failure signals.

The following example illustrates the translation.

Example 8.6. Let $P = s\big[t[\langle a\rangle \mid \langle b\rangle \mid c,d],\mathbf{0}\big] \mid \bar{t}.\bar{s}$ (the same as in Example 5.9). By the LTS (cf. Fig. 2), we have:

$$P \xrightarrow{\tau}_{\mathtt{A}} s[\langle a\rangle \mid \langle b\rangle \mid \langle d\rangle,\mathbf{0}] \mid \bar{s} \xrightarrow{\tau}_{\mathtt{A}} \langle a\rangle \mid \langle b\rangle \mid \langle d\rangle.$$

We apply the translation $(\!|\cdot|\!)_\rho$ on P and illustrate its behaviors:

$$\begin{aligned}
(\!|P|\!)_\varepsilon = {} & s\Big[t\big[p_{t,s}[a] \mid p_{t,s}[b] \mid c\big] \mid r_t.\big(\mathtt{extra}\langle\!\langle t, p_{t,s}, p_s\rangle\!\rangle \mid p_s[d] \mid t.t\langle\!\langle (Y).t[Y] \mid \mathcal{T}_t(Y).\overline{h_t}\rangle\!\rangle\big)\Big] \\
& \mid r_s.\big(\mathtt{extra}\langle\!\langle s, p_s, p_\varepsilon\rangle\!\rangle\big) \mid s.s\langle\!\langle (Y).s[Y] \mid \mathcal{T}_s(Y).\overline{h_s}\rangle\!\rangle \mid \bar{t}.h_t.\bar{s}.h_s \\
\longrightarrow^2 {} & s\Big[t\big[p_{t,s}[a] \mid p_{t,s}[b] \mid c\big] \mid r_t.\big(\mathtt{extra}\langle\!\langle t, p_{t,s}, p_s\rangle\!\rangle \mid p_s[d]\big)\Big] \mid \overline{r_t}.k_t.\overline{h_t} \mid r_s.\big(\mathtt{extra}\langle\!\langle s, p_s, p_\varepsilon\rangle\!\rangle\big) \\
& \mid s.s\langle\!\langle (Y).s[Y] \mid \mathcal{T}_s(Y).\overline{h_s}\rangle\!\rangle \mid h_t.\bar{s}.h_s \\
\longrightarrow^7 {} & s\Big[p_s[a] \mid p_s[b] \mid p_s[d]\Big] \mid r_s.\big(\mathtt{extra}\langle\!\langle s, p_s, p_\varepsilon\rangle\!\rangle\big) \mid s.s\langle\!\langle (Y).s[Y] \mid \mathcal{T}_s(Y).\overline{h_s}\rangle\!\rangle \mid \bar{s}.h_s \\
\longrightarrow^2 {} & s\Big[p_s[a] \mid p_s[b] \mid p_s[d]\Big] \mid r_s.\big(\mathtt{extra}\langle\!\langle s, p_s, p_\varepsilon\rangle\!\rangle\big) \mid \overline{r_s}.k_s.\overline{h_s} \mid h_s \\
\longrightarrow^8 {} & p_\varepsilon[a] \mid p_\varepsilon[b] \mid p_\varepsilon[d].
\end{aligned}$$

The total number of reduction steps is $k = 19$. We have 9 steps for location t and 10 steps for location s:

i) the first step is a synchronization on name t;
ii) the second step is the synchronization between $t\langle\!\langle (Y).t[Y] \mid \mathcal{T}_t(Y).\overline{h_t}\rangle\!\rangle$ and $t\big[p_{t,s}[a] \mid p_{t,s}[b] \mid c\big]$;
iii) the third step is synchronization on name r_t, where $\overline{r_t}$ comes from $\mathcal{T}_t(p_{t,s}[a] \mid p_{t,s}[b] \mid c) = \overline{r_t}.k_t$;
iv) now the process $\mathtt{extrp}\langle\!\langle t, p_{t,s}, p_s\rangle\!\rangle$ is released and the fourth step is the synchronization between update prefix $t\langle\!\langle (Y).t[Y] \mid \mathsf{ch}(t,Y) \mid \mathtt{out}^{\mathtt{s}}(p_{t,s}, p_s\,, \mathtt{nl}(p_{t,s},Y)\,, t\langle\!\langle\dagger\rangle\!\rangle.\overline{k_t})\rangle\!\rangle$ and location $t\big[p_{t,s}[a] \mid p_{t,s}[b] \mid c\big]$;
v) we get process $\mathtt{out}^{\mathtt{s}}(p_{t,s}, p_s\,, 2\,, t\langle\!\langle\dagger\rangle\!\rangle.\overline{j_t})$, which triggers the fifth and sixth reductions: the synchronizations between $p_{t,s}[a]$ and $p_{t,s}[b]$ and update prefixes $p_{t,s}\langle\!\langle (X_1,X_2).p_s[X_1] \mid p_s[X_2] \mid t\langle\!\langle\dagger\rangle\!\rangle.\overline{j_t}\rangle\!\rangle$;
vi) the seventh step is the synchronization between update prefix $t\langle\!\langle\dagger\rangle\!\rangle$ and location $t[c]$, where the update prefix deletes the location together with its content (cf. (11));
vii) the eighth and ninth reduction steps are synchronizations on names k_t and h_t.

At this point, we have the same reduction steps for location s. We have one more reduction step, though, since in process $\mathtt{out}^{\mathtt{s}}(p_s, p_\varepsilon\,, 3\,, s\langle\!\langle\dagger\rangle\!\rangle.\overline{k_s})$ we have 3 locations $p_s[\ldots]$ that have to be relocated to $p_\varepsilon[\ldots]$. Consequently, we have 10 reduction steps for handling location s.

We close this section by illustrating the translation on Example 2.1 (cf. page 4).

Example 8.7. We consider once again the hotel booking scenario. Using compensable processes with aborting semantics we have that:

$$\begin{aligned}
\mathit{Reservation} & \xrightarrow{\tau}_{\mathtt{A}} t[\mathit{pay}.\overline{\mathit{invoice}},\overline{\mathit{refund}}] \mid \overline{\mathit{pay}}.(\mathit{invoice} + \bar{t}.\mathit{refund}) \\
& \xrightarrow{\tau}_{\mathtt{A}} t[\overline{\mathit{invoice}},\overline{\mathit{refund}}] \mid \mathit{invoice} + \bar{t}.\mathit{refund} \xrightarrow{\tau}_{\mathtt{A}} \langle\overline{\mathit{refund}}\rangle \mid \mathit{refund} \xrightarrow{\tau}_{\mathtt{A}} \langle\mathbf{0}\rangle.
\end{aligned}$$

We apply the translation in Definition 8.9 to process *Reservation*:

$$\begin{aligned}
(\!|\mathit{Reservation}|\!)_\varepsilon = {} & t\big[\mathit{book}.\mathit{pay}.\mathit{invoice}\big] \mid r_t.\Big(\mathtt{extra}\langle\!\langle t, p_t, p_\varepsilon\rangle\!\rangle \mid p_\varepsilon[\overline{\mathit{refund}}]\Big) \\
& \mid t.t\langle\!\langle (Y).t[Y] \mid \mathcal{T}_t(Y).\overline{h_t}\rangle\!\rangle \mid \overline{\mathit{book}}.\overline{\mathit{pay}}.\bar{t}.h_t.\mathit{refund} \\
\longrightarrow^4 {} & t\big[\mathit{invoice}\big] \mid r_t.(t\langle\!\langle (Y).t[Y] \mid \mathsf{ch}(t,Y) \\
& \mid \mathtt{out}^{\mathtt{s}}(p_t\,, \mathtt{nl}(p_t,Y)\,, t\langle\!\langle\dagger\rangle\!\rangle.\overline{k_t})\rangle\!\rangle \mid p_\varepsilon[\overline{\mathit{refund}}]) \mid \overline{r_t}.k_t.\overline{h_t} \mid h_t.\mathit{refund} \\
\longrightarrow^2 {} & t\big[\mathit{invoice}\big] \mid \mathtt{out}^{\mathtt{s}}(p_t, p_\varepsilon\,, 0\,, t\langle\!\langle\dagger\rangle\!\rangle.\overline{k_t}) \mid p_\varepsilon[\overline{\mathit{refund}}] \mid k_t.\overline{h_t} \mid h_t.\mathit{refund} \\
\equiv {} & t\big[\mathit{invoice}\big] \mid t\langle\!\langle\dagger\rangle\!\rangle.\overline{k_t} \mid p_\varepsilon[\overline{\mathit{refund}}] \mid k_t.\overline{h_t} \mid h_t.\mathit{refund} \\
\longrightarrow^4 {} & p_\varepsilon[\mathbf{0}] = (\!|\mathbf{0}|\!)_\varepsilon.
\end{aligned}$$

The three steps taken at the beginning are explained in all variants of this example (cf. Example 8.4). The fourth step is the update on location t; the fifth step is a synchronization on name r_t; the sixth step is again an update on the location t; the seventh step is a synchronization on name k_t; the eighth step deletes location t with its content; the last two steps are synchronizations on names h_t and $refund$. Therefore, we get $\langle\!\langle Reservation\rangle\!\rangle_\varepsilon \longrightarrow^{10} p_t[\mathbf{0}]$.

8.2.3. Translating $\mathcal{C}^\lambda$ into $\mathcal{S}$

The translation $\mathcal{C}^\lambda$ into $\mathcal{S}$, denoted $[\![\cdot]\!]^\lambda_\rho$, extends the key ideas of the encoding $[\![\cdot]\!]_\rho$. The set of *reserved location names* $\mathcal{N}^r_l$ is unchanged and the set of *reserved synchronization names* is extended such that $\mathcal{N}^r_s = \{h_x, m_x, k_x, u_x, v_x, e_x, g_x, f_x \mid x \in \mathcal{N}_t\}$.

The function for determining the number of locations (cf. Definition 5.3) is extended as follows:

$$\mathtt{nl}(l, \mathtt{inst}\lfloor\lambda Y.R\rfloor.P) = \mathtt{nl}(l, P). \tag{25}$$

We will use process $\mathtt{out}^s$ as defined for $[\![\cdot]\!]_\rho$ (cf. (12)). We need some additional auxiliary processes.

Definition 8.10 *(Update prefix for extraction).* Let t, l_1, and l_2 be names. We write $\mathtt{extr}\langle\!\langle t, l_1, l_2\rangle\!\rangle$ to stand for the following (subjective) update prefix:

$$\mathtt{extr}\langle\!\langle t, l_1, l_2\rangle\!\rangle = t\langle\!\langle (Y).\Big(t[Y] \mid \mathsf{ch}(t, Y) \mid \mathtt{out}^s(l_1, l_2\,,\, \mathtt{nl}(l, Y)\,,\, \overline{m_t}.\overline{k_t}.t\langle\!\langle\dagger\rangle\!\rangle.\overline{h_t})\Big)\rangle\!\rangle \tag{26}$$

The intuition for the process $\mathtt{extr}\langle\!\langle t, l_1, l_2\rangle\!\rangle$ is the same as in the translation of $\mathcal{C}$ into $\mathcal{S}$ with static recovery (cf. Definition 5.4). The only difference is in the third parameter for process $\mathtt{out}^s$, which enables us to have a controlled execution of adaptable processes, which is important to establish operational correspondence. The prefix $t\langle\!\langle\dagger\rangle\!\rangle$ and name h_t have the same roles as in $[\![\cdot]\!]_\rho$. The differences concern names m_t and k_t: while name m_t ensures that every translation of compensation Q is updated if the translation of compensation update exists, name k_t controls the execution of failure signals.

Remark 8.8 *(Well-formed processes with dynamic recovery).* We revisit the notion of well-formed compensable processes, now with compensation updates. We first present a non well-formed process P_1, and its transition:

$$\times \qquad P_1 = t_1[\mathtt{inst}\lfloor\lambda X.t_2[X,a]\rfloor.b,c] \mid \overline{t_1} \mid \overline{t_2} \xrightarrow{\tau} t_1[b,t_2[c,a]] \mid \overline{t_1} \mid \overline{t_2}. \tag{27}$$

Process P_1 has concurrent error notifications (on t_1 and t_2), and a pair of nested transactions (i.e., (t_1, t_2)) that is hard to capture properly in the representation that we shall give in terms of adaptable processes. In contrast, we would like to consider as well-formed the following process P (where $t_1 \neq t_2$), and we present its transition:

$$\checkmark \qquad P = t_1[\mathtt{inst}\lfloor\lambda X.t_2[X,a]\rfloor.b,c] \mid \overline{t_1}.\overline{t_2} \xrightarrow{\tau} t_1[b,t_2[c,a]] \mid \overline{t_1}.\overline{t_2}. \tag{28}$$

For $\mathcal{C}^\lambda$ processes, the relation for well-formed compensable processes (cf. Fig. 3) should be extended with the following rule:

$$\text{(W-INST)}\ \frac{\Gamma_1;\Delta_1 \vdash_{\gamma_1;\delta_1;p_1} P \quad \Gamma_2;\Delta_2 \vdash_{\gamma_2;\delta_2;p_2} Q \quad \Gamma_3;\Delta_3 \vdash_{\gamma_3;\delta_3;p_3} R \quad f_\lambda(\mathcal{P}(P),\mathcal{P}(Q),\mathcal{P}(R)) = (\Gamma,\Delta) \quad \Gamma^s \cap \Delta^t = \emptyset}{\Gamma;\Delta \vdash_{\bigcup_{i=1}^{3}\gamma_i;\bigcup_{i=1}^{3}\delta_i;\bigvee_{i=1}^{3}p_i} t[\mathtt{inst}\lfloor\lambda X.R\rfloor.P,Q]}$$

where

$$\gamma_1 \times (\gamma_2 \cup \gamma_3) = \{(t',t'') : t' \in \gamma_1 \wedge t'' \in \gamma_2 \cup \gamma_3\} \qquad \text{and} \qquad \{t\} \times \delta = \{(t,t') : t' \in \delta\} \tag{29}$$

and $\mathcal{P}(P) = (\Gamma_1, \Delta_1, \gamma_1, \delta_1)$, $\mathcal{P}(Q) = (\Gamma_2, \Delta_2, \gamma_2, \delta_2)$ and $\mathcal{P}(R) = (\Gamma_3, \Delta_3, \gamma_3, \delta_3)$ and

$$f_\lambda(\mathcal{P}(P),\mathcal{P}(Q),\mathcal{P}(R)) = (\Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup (\gamma_1 \times (\gamma_2 \cup \gamma_3))\,,\, \Delta_1 \cup \Delta_2 \cup \Delta_3 \cup (\{t\} \times (\delta_1 \cup \delta_2 \cup \delta_3 \cup \gamma_1 \cup \gamma_2 \cup \gamma_3))) \tag{30}$$

Rule (W-INST) specifies the conditions for $t[\mathtt{inst}\lfloor\lambda X.R\rfloor.P,Q]$ to be well-formed; it relies on the key ideas of the Rule (W-TRANS). Therefore, $\delta = \{t\}$. The set of pairs of parallel failure signals is the union of the respective sets for P, Q and R and the set whose elements are pairs of failure signals; in the pair, one element belongs to the set of failure signals of P, the second element is from the union of sets of failure signals of Q and R. This extension with $\gamma_1 \times (\gamma_2 \cup \gamma_3)$ is necessary for $t[\mathtt{inst}\lfloor\lambda X.R\rfloor.P,Q]$, because P may contain protected blocks which will be composed in parallel with $R\{^{Q}/_{X}\}$ in case of a failure signal. The set of pairs of nested transactions is obtained from those for P, Q, and R also considering further pairs as specified by $\{t\} \times (\delta_1 \cup \delta_2 \cup \delta_3 \cup \gamma_1 \cup \gamma_2 \cup \gamma_3)$ (cf. (29)). The rule additionally enforces that the sets of parallel failure

$$\begin{aligned}
[\![t[P,Q]]\!]^{\lambda}_{\rho} &= t\Big[[\![P]\!]^{\lambda}_{t,\rho}\Big] \mid t.\Big(\mathtt{extr}\langle\!\langle t, p_{t,\rho}, p_{\rho}\rangle\!\rangle \mid m_t.p_{\rho}\big[v_t\langle\!\langle (X).(X \mid u_t[\overline{f_t}.\overline{g_t}.k_t])\rangle\!\rangle\big]\Big)\\
&\qquad \mid v_t\big[u_t\langle\!\langle (Z).(Z \mid e_t[[\![Q]\!]^{\lambda}_{\varepsilon}] \mid f_t.e_t\langle\!\langle (X).X\rangle\!\rangle.g_t)\rangle\!\rangle\big]\\
[\![\mathtt{inst}\lfloor\lambda Y.R\rfloor.P]\!]^{\lambda}_{t,\rho} &= u_t\Big[e_t\langle\!\langle (Y).(\overline{g_t}.u_t\langle\!\langle (Z).(Z \mid e_t[[\![R]\!]^{\lambda}_{\varepsilon}] \mid f_t.e_t\langle\!\langle (X).X\rangle\!\rangle.g_t)\rangle\!\rangle)\rangle\!\rangle.(\overline{f}_t.e_t[\mathbf{0}])\Big] \mid [\![P]\!]^{\lambda}_{t,\rho}\\
[\![\langle P\rangle]\!]^{\lambda}_{\rho} &= p_{\rho}\big[[\![P]\!]^{\lambda}_{\varepsilon}\big]\\
[\![a.P]\!]^{\lambda}_{\rho} &= a.[\![P]\!]^{\lambda}_{\rho}\\
[\![\overline{a}.P]\!]^{\lambda}_{\rho} &= \overline{a}.[\![P]\!]^{\lambda}_{\rho}\\
[\![\overline{t}.P]\!]^{\lambda}_{\rho} &= \overline{t}.h_t.[\![P]\!]^{\lambda}_{\rho}\\
[\![Y]\!]^{\lambda}_{\rho} &= Y
\end{aligned}$$

Fig. 15. Translating $\mathcal{C}^{\lambda}$ into $\mathcal{S}$.

signals and nested transaction names in the parallel composition are disjoint. For example, for process (28) above we can derive:

$$\emptyset; \{(t_1, t_2)\} \vdash_{\{t_1,t_2\};\{t_1\};\perp} t_1[\mathtt{inst}\lfloor\lambda X.t_2[X,a]\rfloor.b,c] \mid \overline{t_1}.\overline{t_2}.$$

In contrast, process (27) does not satisfy the predicate, since its sets of pairs of parallel failure signals and nested transaction names are not disjoint: they are both equal to $\{(t_1, t_2)\}$.

Using these modifications, the translation of $\mathcal{C}^{\lambda}$ into $\mathcal{S}$ extends Definition 5.5 (page 14) as follows:

Definition 8.11 *(Translating $\mathcal{C}^{\lambda}$ into $\mathcal{S}$).* Let ρ be a path. We define the translation of compensable processes with dynamic recovery into (subjective) adaptable processes as a tuple $([\![\cdot]\!]^{\lambda}_{\rho}, \varphi_{[\![\cdot]\!]^{\lambda}_{\rho}})$ where:

(a) The renaming policy

$$\varphi_{[\![\cdot]\!]^{\lambda}_{\rho}}(x) = \begin{cases} \{x\} & \text{if } x \in \mathcal{N}_s\\ \{x, h_x, m_x, k_x, u_x, v_x, e_x, g_x, f_x\} \cup \{p_{\rho} : x \in \rho\} & \text{if } x \in \mathcal{N}_t. \end{cases}$$

(b) The translation $[\![\cdot]\!]^{\lambda}_{\rho} : \mathcal{C} \longrightarrow \mathcal{S}$ is as in Fig. 15 and as a homomorphism for other operators.

Key elements in Fig. 15 are the translations of $t[P,Q]$ and $\mathtt{inst}\lfloor\lambda Y.R\rfloor.P_1$, which are closely related to each other. Indeed, these translations share location names u_t, v_t, and e_t (as well as names f_t and g_t) in order to account for the possible replacement of Q in $t[P,Q]$ with R in $\mathtt{inst}\lfloor\lambda Y.R\rfloor.P_1$, using updates.

As stated earlier, $\mathtt{inst}\lfloor\lambda X.R\rfloor.P$ produces a new compensation behavior $R\{^{Q}/_{X}\}$ after an internal transition. The following statement formalizes the encoding of process $R\{^{Q}/_{X}\}$:

Lemma 8.9. *Suppose R is a well-formed compensable process. Then $[\![R\{^{Q}/_{X}\}]\!]^{\lambda}_{\rho} = [\![R]\!]^{\lambda}_{\rho}\{^{[\![Q]\!]^{\lambda}_{\rho}}/_{X}\}$.*

Example 8.10. Let us consider the process $[\![t[\mathtt{inst}\lfloor\lambda X.R\rfloor.P_1, Q]]\!]^{\lambda}_{\rho}$. Some intuitions follow:

i) In $[\![\mathtt{inst}\lfloor\lambda X.R\rfloor.P_1]\!]^{\lambda}_{t,\rho}$ we find process $[\![R]\!]^{\lambda}_{\varepsilon}$ on location u_t, which is composed in parallel with process $[\![P]\!]^{\lambda}_{t,\rho}$. This location may synchronize with the update prefix on name u_t that is implemented in $[\![t[P,Q]]\!]^{\lambda}_{\rho}$: such a step would move $[\![R]\!]^{\lambda}_{\varepsilon}$ from location t to location v_t, leaving $[\![P]\!]^{\lambda}_{t,\rho}$ in t.
ii) In the translation of $t[P,Q]$, process $[\![Q]\!]^{\lambda}_{\varepsilon}$ resides in location e_t. This location may synchronize with the update prefix implemented in $[\![\mathtt{inst}\lfloor\lambda X.R\rfloor.P_1]\!]^{\lambda}_{t,\rho}$, which contains $[\![R]\!]^{\lambda}_{\varepsilon}$: such a step allows us to obtain $[\![R]\!]^{\lambda}_{\rho}\{^{[\![Q]\!]^{\lambda}_{\rho}}/_{Y}\}$ (cf. Lemma 8.9).
iii) The translations use synchronizations on f_t, e_t, and g_t to preserve operational correspondence.

More concretely, consider the following step (cf. (20)):

$$P = t[\mathtt{inst}\lfloor\lambda Y.R\rfloor.P_1, Q] \xrightarrow{\tau} t[P_1, R\{^{Q}/_{Y}\}] = P'.$$

We then have the following, using S to stand for $t.\Big(\mathtt{extr}\langle\!\langle t, p_t, p_{\varepsilon}\rangle\!\rangle \mid m_t.p_{\varepsilon}\big[v_t\langle\!\langle (X).(X \mid u_t[\overline{f_t}.\overline{g_t}.k_t])\rangle\!\rangle\big]\Big)$:

$$
\begin{aligned}
[\![P]\!]^{\lambda}_{\varepsilon} &= [\![t[\mathtt{inst}\lfloor \lambda Y.R\rfloor.P_1, Q\,]]\!]^{\lambda}_{\varepsilon}\\
&= t\Big[[\![\mathtt{inst}\lfloor \lambda Y.R\rfloor.P_1]\!]^{\lambda}_{t}\Big] \mid S \mid v_t\big[u_t\langle\!\langle (Z).(Z \mid e_t[[\![Q\,]\!]^{\lambda}_{\varepsilon}] \mid f_t.e_t\langle\!\langle (X).X\rangle\!\rangle.g_t)\rangle\!\rangle\big]\\
&= t\Big[u_t\Big[e_t\langle\!\langle (Y).(\overline{g_t}.u_t\langle\!\langle (Z).(Z \mid e_t[[\![R]\!]^{\lambda}_{\varepsilon}] \mid f_t.e_t\langle\!\langle (X).X\rangle\!\rangle.g_t)\rangle\!\rangle)\rangle\!\rangle.(\overline{f_t}.e_t[\mathbf{0}])\Big] \mid [\![P_1]\!]^{\lambda}_{t}\Big] \mid S\\
&\quad\;\; \mid v_t\big[u_t\langle\!\langle (Z).(Z \mid e_t[[\![Q\,]\!]^{\lambda}_{\varepsilon}] \mid f_t.e_t\langle\!\langle (X).X\rangle\!\rangle.g_t)\rangle\!\rangle\big]\\
&\longrightarrow t\Big[[\![P_1]\!]^{\lambda}_{t}\Big] \mid S \mid v_t\big[e_t\langle\!\langle (Y).(\overline{g_t}.u_t\langle\!\langle (Z).(Z \mid e_t[[\![R]\!]^{\lambda}_{\varepsilon}] \mid f_t.e_t\langle\!\langle (X).X\rangle\!\rangle.g_t)\rangle\!\rangle)\rangle\!\rangle.(\overline{f_t}.e_t[\mathbf{0}])\\
&\quad\;\; \mid e_t[[\![Q\,]\!]^{\lambda}_{\varepsilon}] \mid f_t.e_t\langle\!\langle (X).X\rangle\!\rangle.g_t\big]\\
&\longrightarrow t\Big[[\![P_1]\!]^{\lambda}_{t}\Big] \mid S \mid v_t\big[\overline{g_t}.u_t\langle\!\langle (Z).(Z \mid e_t[[\![R]\!]^{\lambda}_{\varepsilon}\{{}^{[\![Q\,]\!]^{\lambda}_{\varepsilon}}/_{Y}\}] \mid f_t.e_t\langle\!\langle (X).X\rangle\!\rangle.g_t)\rangle\!\rangle \mid \overline{f_t}.e_t[\mathbf{0}] \mid f_t.e_t\langle\!\langle (X).X\rangle\!\rangle.g_t\big]\\
&\longrightarrow^{3} t\Big[[\![P_1]\!]^{\lambda}_{t}\Big] \mid S \mid v_t\big[u_t\langle\!\langle (Z).(Z \mid e_t[[\![R]\!]^{\lambda}_{\varepsilon}\{{}^{[\![Q\,]\!]^{\lambda}_{\varepsilon}}/_{Y}\}] \mid f_t.e_t\langle\!\langle (X).X\rangle\!\rangle.g_t)\rangle\!\rangle\big]\\
&= [\![t[P_1, R\{{}^{Q}/_{Y}\}]]\!]^{\lambda}_{\varepsilon}
\end{aligned}
$$

Therefore, $[\![P]\!]^{\lambda}_{\epsilon} \longrightarrow^{5} [\![P']\!]^{\lambda}_{\epsilon}$. As mentioned above, the first and second steps are updates on locations u and e_t, respectively. The three other steps are: a synchronization on f_t, an update on location e_t, and a synchronization on g_t.

9. Related work

Studies on the expressiveness of process calculi have a long history and constitute a vibrant research area. We refer the reader to [21] for a recent account on modern approaches to formal comparisons between different process calculi. In this paper, we have followed Gorla's framework for formalizing encodability and separation results [12]. With respect to the criteria in [12], our definition of valid encoding (Definition 4.5) presents the following differences. First, to account for the paths ρ in which transactions reside, we consider a notion of compositionality that is slightly less flexible than Gorla's. Second, we rely on a form of operational completeness that, unlike Gorla's, explicitly describes the number of steps required to mimic a step in the source language. Finally, we consider a new criterion, called efficiency, which allows us to precisely compare our two main encodings (Definition 7.1). We do not know of prior works using criteria similar to efficiency.

The closest related works are by Lanese, Vaz, and Ferreira [15] and by Lanese and Zavattaro [16]. The work in [15], already mentioned in the Introduction, analyzes the expressive power of the compensation calculus focusing on three different specification mechanisms for compensations: static recovery, parallel recovery, and dynamic recovery. The authors show that parallel recovery (where the compensation is dynamically built as the parallel composition of compensation elements) can be compositionally encoded using static recovery; they also show the impossibility of encoding dynamic recovery using static recovery. The work in [16] sheds further light on the fundamental differences between static and dynamic recovery: it is shown that termination (i.e., the absence of an infinite computation path starting from a given process) is a decidable property for processes with static recovery but undecidable for processes with dynamic recovery.

Our expressiveness results complement the findings in [15,16] by implementing static and dynamic recovery in compensable processes using the different process framework defined by adaptable processes. In the same line, although slightly less related, Vaz and Ferreira [24] study criteria for determining when a compensable process is *correct* and establish that self-healing compensations are correct. The criteria in [24] are different from the notion of well-formed compensable processes that we developed to formalize our encodings, for which error notifications are crucial.

Bravetti and Zavattaro [4] compare the expressiveness of variants of Milner's CCS extended with the interrupt operator of CSP, the try-catch operator for exception handling, and operators for replication and recursion. Their comparison is based on the (un)decidability of existential and universal termination problems: the former concerns the existence of one terminating computation, whereas the latter asks whether all computations terminate. They prove that in CCS with replication there is no difference between interrupt and try-catch: universal termination is decidable while existential termination is not. In contrast, in CCS with recursion and try-catch, the universal termination problem becomes undecidable, thus revealing an expressiveness gap with respect to the language with recursion and interrupt.

10. Concluding remarks

In this paper, we have developed rigorous connections between programming abstractions for compensation handling (typical of models for services and long-running transactions) and for run-time adaptation. Specifically, we compared from the point of view of relative expressiveness two related and yet fundamentally different process models: the calculus of compensable processes [15] and the calculus of adaptable processes [3]. We developed two encodings of compensable processes (with static compensations under discarding semantics) into adaptable processes with *subjective* and *objective* mobility.

We have shown that our encodings are correct up to five well-established criteria [12]: name invariance, compositionality, operational correspondence (divided into soundness and completeness properties), divergence reflection, and success sensitiveness. Our encodings not only constitute a non trivial application of two sensible forms of mobility for adaptable processes; in our view, they also shed light on the (intricate) semantics of compensable processes. We compared our encodings from the point of view of *efficiency*, a comparison criterion formally defined in terms of the number of target steps required to mimic a source step. In this sense, subjective mobility allows us to encode compensable processes more efficiently than objective mobility. The efficiency gains induced by subjective mobility depend on the number of compensation actions in the source process.

Our encodings are robust because of the correctness criteria they satisfy, but also because they admit extensions to different variants of compensable processes. Indeed, we have informally discussed how to extend our encoding into subjective adaptable processes so as to account for compensable processes under preserving and discarding semantics and with dynamic compensations. These extensions require targeted modifications that largely preserve the essence of the encoding for discarding semantics.

Interestingly, our work uncovers an interesting dichotomy: should one appeal to objective or to subjective updates? A subjective update would appear more "autonomous" than an objective update, because it is determined by a located process itself, not by its environment. Still, we believe that the choice between objective and subjective updates largely depends on the application at hand: it is easy to imagine practical scenarios of dynamic reconfiguration for which each form of update is better suited. Hence, a general specification language should probably include both objective and subjective updates.

In future work, we would like to further study the connection between subjective and objective updates. An initial insight is the following: subjective updates can represent objective updates, at least in an ad-hoc manner. Consider process $S = C_1\big[l[P] \mid R_1\big] \mid C_2\big[l\{(X).Q\}.R_2\big]$, which, as we have seen, reduces to $C_1\big[Q\{^P/_X\} \mid R_1\big] \mid C_2\big[R_2\big]$. Now consider S', a process similar to S but with subjective update prefixes:

$$S' = C_1\big[l[P] \mid l_1\langle\langle(X).X\rangle\rangle \mid R_1\big] \mid C_2\big[l\langle\langle(X).l_1[Q]\rangle\rangle.R_2\big]$$

In S', we assume that name l_1 does not occur in P, Q, R_1, and R_2. Using two reductions, S' emulates the movement induced by the reduction step originated in S:

$$\begin{aligned} S' &\longrightarrow C_1\big[\mathbf{0} \mid l_1\langle\langle(X).X\rangle\rangle \mid R_1\big] \mid C_2\big[l_1[Q\{^P/_X\}].R_2\big] \\ &\longrightarrow C_1\big[Q\{^P/_X\} \mid R_1\big] \mid C_2\big[\mathbf{0} \mid R_2\big] \end{aligned}$$

That is, the update prefix $l_1\langle\langle(X).X\rangle\rangle$ serves as an "anchor" to bring the reconfigured process $Q\{^P/_X\}$ back to its original context C_1.

Similarly, we can represent subjective updates using objective prefixes. Consider process $L = C_1\big[l[P] \mid R_1\big] \mid C_2\big[l\langle\langle(X).Q\rangle\rangle.R_2\big]$, which reduces to $C_1\big[\mathbf{0} \mid R_1\big] \mid C_2\big[Q\{^P/_X\} \mid R_2\big]$. Now consider process L':

$$L' = C_1\big[l[P] \mid R_1\big] \mid C_2\big[l\{(X).l_1\{(Y).Q\}.\mathbf{0}\}.R_2 \mid l_1[\mathbf{0}]\big]$$

As in process S', in L' we assume that name l_1 is fresh; also, we assume that P and Q do not contain free occurrences of variable Y. Process L' uses two reduction steps to mimic the reduction step originated in L:

$$\begin{aligned} L' &\longrightarrow C_1\big[l_1\{(Y).Q\{^P/_X\}\}.\mathbf{0} \mid R_1\big] \mid C_2\big[R_2 \mid l_1[\mathbf{0}]\big] \\ &\longrightarrow C_1\big[\mathbf{0} \mid R_1\big] \mid C_2\big[R_2 \mid Q\{^P/_X\}\big] \end{aligned}$$

Here, we use location $l_1[\mathbf{0}]$ to bring the reconfigured process $Q\{^P/_X\}$ back to its original context C_2.

Crucially, these examples show that the ability of emulating a certain style of process mobility (subjective or objective) comes at the price of additional reduction steps, which could entail inefficient encodings. This observation reinforces our claim that a specification language should natively support both forms of update.

Having addressed the encodability of compensable processes into adaptable processes, we plan to consider the *reverse direction*, i.e., encodings of adaptable processes into compensable processes. We conjecture that an encoding of adaptable process into a language with static compensations does not exist: compensation updates $\mathtt{inst}\lfloor\lambda X.Q\rfloor.P$ seem essential to model an update prefix $l\{(X).Q\}.P$—the semantics of both constructs induces process substitutions. Still, even by considering a language with dynamic compensations, an encoding of adaptable processes is far from obvious, because the semantics of compensation updates dynamically modifies the behavior of the compensation activity, the inactive part of a transaction. Formalizing these (non) encodability claims is interesting future work.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We are grateful to the anonymous reviewers for their detailed and constructive feedback.

This work was supported by EU COST Action IC1201 ("BETTY"), SFRS#6458932 ("ALADDIN"), MPNTR ("Innovative scientific and artistic research from domain of FTS"), and the Dutch Research Council (NWO) Project No. 016.Vidi.189.046 (Unifying Correctness for Communicating Software).

Appendix A. Omitted proofs and definitions for Section 3

A.1. Proof of Proposition 3.1 (page 7)

Lemma A.1. *Let P be a compensable process.*

(a) *If $P \xrightarrow{a} P'$ then $P = C[a.P_1]$ and $P' = C[P_1]$.*
(b) *If $P \xrightarrow{t} P'$ then $P = C[t[P_1, Q_1]]$ and $P' = C[\mathsf{extr}(P_1) \mid \langle Q_1 \rangle]$.*
(c) *If $P \xrightarrow{\bar{x}} P'$ then $P = C[\bar{x}.P_1]$ and $P' = C[P_1]$*

for some context C, names t, a, x and processes P_1, Q_1.

Proof. The proof is by induction on the derivation of $P \xrightarrow{\alpha} P'$, in each case. □

We repeat the statement from page 7:

Proposition 3.1. *Let P be a compensable process. If $P \xrightarrow{\tau} P'$ then one of the following holds:*

(a) *$P \equiv E[C[\bar{a}.P_1] \mid D[a.P_2]]$ and $P' \equiv E[C[P_1] \mid D[P_2]]$,*
(b) *$P \equiv E[C[t[P_1, Q]] \mid D[\bar{t}.P_2]]$ and $P' \equiv E[C[\mathsf{extr}(P_1) \mid \langle Q \rangle] \mid D[P_2]]$,*
(c) *$P \equiv C[t[D[\bar{t}.P_1], Q]]$ and $P' \equiv C[\mathsf{extr}(D[P_1]) \mid \langle Q \rangle]$,*

for some contexts C, D, E, processes P_1, P_2, Q and names a, t.

Proof. The proof proceeds by induction on the inference of $P \xrightarrow{\tau} P'$. We will show that the proposition is true for the base cases, whereas the inductive step follows directly. By the LTS in Fig. 2, in accordance with the Rules (L-COMM1) and (L-REC-IN) we have two possible cases as follows:

(a)-(b) By Rule (L-COMM1) we have: $P \equiv P_1' \mid P_2'$, $P_1' \xrightarrow{x} P_1''$, $P_2' \xrightarrow{\bar{x}} P_2''$, $P' \equiv P_1'' \mid P_2''$ and by Lemma A.1, we conclude that:
  (a) $P_2' \equiv D[\bar{a}.P_2]$, $P_2'' \equiv D[P_2]$, $P_1' \equiv C[a.P_1]$, and $P_1'' \equiv C[P_1]$, or
  (b) $P_2' \equiv D[\bar{t}.P_2]$, $P_2'' \equiv D[P_2]$, $P_1' \equiv C[t[P_1, Q_1]]$, and $P_1'' \equiv C[\mathsf{extr}(P_1) \mid \langle Q_1 \rangle]$.

(c) By Rule (L-REC-IN) we have: $P \equiv t[P_1', Q]$, $P_1' \xrightarrow{\bar{t}} R$, $P' \equiv \mathsf{extr}(R) \mid \langle Q \rangle$ and by Lemma A.1, we conclude that: $P_1' \equiv D[\bar{t}.P_1]$ and $R \equiv D[P_1]$,

for $E[\bullet] = [\bullet]$ and for some contexts C, D, processes $P_1, P_2, Q_1, P_1', P_2', P_1'', P_2'', Q$, and names a, t. □

A.2. Properties of well-formed processes (Definition 3.4)

In the following we are going to prove that well-formed processes always evolve into well-formed processes. Before giving the statement we show its supporting results.

Lemma A.2 *(Inversion lemma). For some $\Gamma_1, \Delta_1, \gamma_1, \delta_1, p_1, \Gamma_2, \Delta_2, \gamma_2, \delta_2, p_2$, the following holds:*

1) *If $\Gamma; \Delta \vdash_{\gamma;\delta;p} \mathbf{0}$ then $\Gamma, \Delta, \gamma, \delta$ are empty sets and $p = \perp$;*
2) *If $\Gamma; \Delta \vdash_{\gamma;\delta;p} \bar{t}.P$ then there is γ' such that $\gamma = \gamma' \cup \{t\}$ and $\Gamma; \emptyset \vdash_{\gamma';\emptyset;\perp} P$ and $\Delta = \emptyset$.*
3) *If $\Gamma; \Delta \vdash_{\gamma;\delta;p} \bar{a}.P$ then $\Gamma; \emptyset \vdash_{\gamma;\emptyset;\perp} P$ and $\Delta = \emptyset$ and $p = \perp$;*
4) *If $\Gamma; \Delta \vdash_{\gamma;\delta;p} a.P$ then $\Gamma; \emptyset \vdash_{\gamma;\emptyset;\perp} P$ and $\Delta = \emptyset, \delta = \emptyset$ and $\gamma = \emptyset$;*
5) *If $\Gamma; \Delta \vdash_{\gamma;\delta;p} (\nu x)P$ then $\Gamma; \Delta \vdash_{\gamma;\delta;p} P$;*
6) *If $\Gamma; \Delta \vdash_{\gamma;\delta;p} \langle P \rangle$ then $p = \top$ and $\Gamma; \Delta \vdash_{\gamma;\delta;p'} P$ for some $p' \in \{\top, \perp\}$;*

7) *If* $\Gamma; \Delta \vdash_{\gamma;\delta;p} !\pi.P$ *then* $\Gamma'; \emptyset \vdash_{\gamma;\emptyset;\bot} \pi.P$ *and* $\Gamma = \gamma \times \gamma$ *and* $\Delta = \emptyset, \delta = \emptyset$ *and* $p = \bot$*;*
8) *If* $\Gamma; \Delta \vdash_{\gamma;\delta;p} P \mid Q$ *then* $\Gamma_1; \Delta_1 \vdash_{\gamma_1;\delta_1;p_1} P$ *and* $\Gamma_2; \Delta_2 \vdash_{\gamma_2;\delta_2;p_2} Q$ *and* $\Gamma = \Gamma_1 \cup \Gamma_2 \cup (\gamma_1 \times \gamma_2)$ *and* $\Delta = \Delta_1 \cup \Delta_2$ *and* $\gamma = \gamma_1 \cup \gamma_2$ *and* $\delta = \delta_1 \cup \delta_2$ *and* $p = p_2 \vee p_2$ *and* $\Gamma^s \cap \Delta^t = \emptyset$*;*
9) *If* $\Gamma; \Delta \vdash_{\gamma;\delta;p} t[P, Q]$ *then* $\Gamma_1; \Delta_1 \vdash_{\gamma_1;\delta_1;p_1} P$ *and* $\Gamma_2; \Delta_2 \vdash_{\gamma_2;\delta_2;p_2} Q$ *and* $\Gamma = \Gamma_1 \cup \Gamma_2 \cup (\gamma_1 \times \gamma_2)$ *and* $\Delta = \Delta_1 \cup \Delta_2 \cup (\{t\} \times (\delta_1 \cup \delta_2 \cup \gamma_1 \cup \gamma_2))$ *and* $\gamma = \gamma_1 \cup \gamma_2$ *and* $\delta = \delta_1 \cup \delta_2$ *and* $p = p_2 \vee p_2$ *and* $\Gamma^s \cap \Delta^t = \emptyset$*.*

Proof. The proof follows directly from the auxiliary relation for well-formed compensable processes, cf. Fig. 3. □

In the following we introduce auxiliary statements that are needed for proving that well-formedness of compensable process is preserved by the rules in Fig. 2.

Lemma A.3. *If* $\Gamma; \Delta \vdash_{\gamma;\delta;p} t[P, Q]$ *then there are* Δ_1, δ_1 *such that* $\Gamma, \Delta_1 \vdash_{\gamma;\delta_1;\top} P \mid \langle Q \rangle$ *and* $\Delta_1 \subseteq \Delta$*.*

Proof. Let $\Gamma; \Delta \vdash_{\gamma;\delta;p} t[P, Q]$.

- By Lemma A.2 it follows $\Gamma'_1; \Delta'_1 \vdash_{\gamma'_1;\delta'_1;p'_1} P$ and $\Gamma'_2; \Delta'_2 \vdash_{\gamma'_2;\delta'_2;p'_2} Q$ where $\Gamma = \Gamma'_1 \cup \Gamma'_2 \cup (\gamma'_1 \times \gamma'_2)$ and $\Delta = \Delta'_1 \cup \Delta'_2 \cup (\{t\} \times (\delta'_1 \cup \delta'_2 \cup \gamma'_1 \cup \gamma'_2))$ and $\gamma = \gamma'_1 \cup \gamma'_2$ and $p = p'_1 \vee p'_2$ and $\Gamma^s \cap \Delta^t = \emptyset$.
- By formation on Rule (W-BLOCK) we get that $\Gamma'_2; \Delta'_2 \vdash_{\gamma'_2;\delta'_2;\top} \langle Q \rangle$.
- By formation on Rule (W-PAR) we get that $\Gamma, \Delta_1 \vdash_{\gamma;\delta_1;\top} P \mid \langle Q \rangle$ where $\Delta_1 = \Delta'_1 \cup \Delta'_2$ and it is clear that $\Delta_1 \subseteq \Delta$.
- It should be noted that for $\Delta_1 \subseteq \Delta$, based on basic properties of set operations from $\Gamma^s \cap \Delta^t = \emptyset$, that it follows $\Gamma^s \cap \Delta_1^t = \emptyset$. □

Lemma A.4. *If* $\Gamma; \Delta \vdash_{\gamma;\delta;p} P$ *then there are* $\Gamma_1, \Delta_1, \gamma_1, \delta_1$ *and* p_1 *such that* $\Gamma_1, \Delta_1 \vdash_{\gamma_1;\delta_1;p_1} \mathsf{extr}(P)$ *and* $\Gamma_1 \subseteq \Gamma, \Delta_1 \subseteq \Delta$ *and* $\gamma_1 \subseteq \gamma$*.*

Proof. The proof proceeds by induction over the structure of process P. We consider one base case and the three most interesting cases of the six cases for process P, and they are: parallel composition, protected block and transaction scope.

<u>Base case</u>: The statement holds for $P = \mathbf{0}$ since $\mathsf{extr}(\mathbf{0}) = \mathbf{0}$ and $\emptyset; \emptyset \vdash_{\emptyset;\emptyset;\bot} \mathbf{0}$.

<u>Induction step</u>:

- ***<u>Case 1:</u>*** Let $P = P' \mid Q'$ and $\Gamma; \Delta \vdash_{\gamma;\delta;p} P' \mid Q'$.
  - By Lemma A.2 it follows $\Gamma'_1; \Delta'_1 \vdash_{\gamma'_1;\delta'_1;p'_1} P'$ and $\Gamma'_2; \Delta'_2 \vdash_{\gamma'_2;\delta'_2;p'_2} Q'$, where $\Gamma = \Gamma'_1 \cup \Gamma'_2 \cup (\gamma'_1 \times \gamma'_2)$ (cf. (7)), $\Delta = \Delta'_1 \cup \Delta'_2$, $\delta = \delta'_1 \cup \delta'_2$ and $p = p'_1 \vee p'_2$, and hold that $\Gamma^s \cap \Delta^t = \emptyset$.
  - By definition of extraction function (cf. Fig. 1) we get: $\mathsf{extr}(P' \mid Q') = \mathsf{extr}(P') \mid \mathsf{extr}(Q')$.
  - For process P' by induction hypothesis there are $\Gamma''_1 \subseteq \Gamma'_1$ and $\Delta''_1 \subseteq \Delta'_1$ such that: $\Gamma''_1; \Delta''_1 \vdash_{\gamma''_1;\delta''_1;p''_1} \mathsf{extr}(P')$.
  - Similarly, for process Q' by induction hypothesis there are $\Gamma''_2 \subseteq \Gamma'_2$ and $\Delta''_2 \subseteq \Delta'_2$ such that: $\Gamma''_2; \Delta''_2 \vdash_{\gamma''_2;\delta''_2;p''_2} \mathsf{extr}(Q')$.
  - By formation on Rule (W-PAR) we get: $\Gamma_1; \Delta_1 \vdash_{\gamma_1;\delta_1;p_1} \mathsf{extr}(P') \mid \mathsf{extr}(Q')$ where $\Gamma_1 = \Gamma''_1 \cup \Gamma''_2 \cup (\gamma''_1 \times \gamma''_2)$, $\Delta_1 = \Delta''_1 \cup \Delta''_2$, $\delta_1 = \delta''_1 \cup \delta''_2$ and $p_1 = p''_1 \vee p''_2$, also holds $\Gamma_1^s \cap \Delta_1^t = \emptyset$.
  - It is easy to conclude: $\Gamma_1 \subseteq \Gamma, \Delta_1 \subseteq \Delta$ and $\gamma_1 \subseteq \gamma$.
- ***<u>Case 2:</u>*** Let $P = \langle P' \rangle$ and $\Gamma; \Delta \vdash_{\gamma;\delta;p} \langle P' \rangle$. By definition of extraction function (cf. Fig. 1) we get: $\mathsf{extr}(\langle P' \rangle) = \langle P' \rangle$. Therefore, it is easy to be concluded that the statement holds.
- ***<u>Case 3:</u>*** Let $P = t[P', Q']$ and $\Gamma; \Delta \vdash_{\gamma;\delta;p} t[P', Q']$.
  - By Lemma A.2 we get: $\Gamma'_1; \Delta'_1 \vdash_{\gamma'_1;\delta'_1;p'_1} P'$ and $\Gamma'_2; \Delta'_2 \vdash_{\gamma'_2;\delta'_2;p'_2} Q'$, where $\Gamma = \Gamma'_1 \cup \Gamma'_2 \cup (\gamma'_1 \times \gamma'_2)$, $\Delta = \Delta'_1 \cup \Delta'_2 \cup (\{t\} \times (\delta'_1 \cup \delta'_2 \cup \gamma'_1 \cup \gamma'_2))$, $\delta = \delta'_1 \cup \delta'_2$ and $p = p'_1 \vee p'_2$, also condition $\Gamma^s \cap \Delta^t = \emptyset$ holds.
  - By definition of extraction function (cf. Fig. 1) we get $\mathsf{extr}(P) = \mathbf{0}$. Therefore, statement holds directly.

For all other cases for the process P the proof follows directly, because of definition of extraction function. □

Lemma A.5. *If* $\Gamma; \Delta \vdash_{\gamma;\delta;p} P \mid Q$ *then there are* $\Gamma_1, \Delta_1, \gamma_1, \delta_1$ *and* p_1 *such that* $\Gamma_1, \Delta_1 \vdash_{\gamma_1;\delta_1;p_1} \mathsf{extr}(P) \mid Q$ *and* $\Gamma_1 \subseteq \Gamma, \Delta_1 \subseteq \Delta$ *and* $\gamma_1 \subseteq \gamma$*.*

Proof. Let $\Gamma; \Delta \vdash_{\gamma;\delta;p} P \mid Q$.

- By Lemma A.2 we get $\Gamma_1'; \Delta_1' \vdash_{\gamma_1';\delta_1';p_1'} P$ and $\Gamma_2'; \Delta_2' \vdash_{\gamma_2';\delta_2';p_2'} Q$ where $\Gamma = \Gamma_1' \cup \Gamma_2' \cup (\gamma_1' \times \gamma_2')$ (cf. (7)), $\Delta = \Delta_1' \cup \Delta_2', \gamma = \gamma_1' \cup \gamma_2', \delta = \delta_1' \cup \delta'$ and $p = p_1' \vee p_2'$ and condition $\Gamma^s \cap \Delta^t = \emptyset$ holds.
- For process P by Lemma A.4 there are $\Gamma_1'' \subseteq \Gamma_1'$ and $\Delta_1'' \subseteq \Delta'$ such that: $\Gamma_1''; \Delta_1'' \vdash_{\gamma_1'';\delta_1'';p_1''} \mathsf{extr}(P)$.
- For processes $\mathsf{extr}(P)$ and Q by formation on Rule (W-PAR) the following holds: $\Gamma_1; \Delta_1 \vdash_{\gamma_1;\delta_1;p_1} \mathsf{extr}(P) \mid Q$ where $\Gamma_1 = \Gamma_1'' \cup \Gamma_2' \cup (\gamma_1'' \times \gamma_2'), \Delta_1 = \Delta_1'' \cup \Delta_2', \gamma_1 = \gamma_1'' \cup \gamma_2', \delta_1 = \delta_1'' \cup \delta_2'$ and $p_1 = p_1'' \vee p_2'$ and $\Gamma_1^s \cap \Delta_1^t = \emptyset$ holds by basic properties of set operations. □

We may now prove a soundness result, which ensures that well-formedness is preserved under LTS rules. We repeat the theorem's statement at page 9:

Theorem 3.4. *If $\Gamma; \Delta \vdash_{\gamma;\delta;p} P$ and $P \xrightarrow{\alpha} P'$ then there are $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$ such that $\Gamma'; \Delta' \vdash_{\gamma';\delta';p'} P'$.*

Proof. The proof proceeds by induction on the depth of the derivation $P \xrightarrow{\alpha} P'$.

Base cases: In the following we consider four base cases.

- ***Base case 1:*** Assume that $\Gamma; \Delta \vdash_{\gamma;\delta;p} \bar{t}.P$ and if the last applied rule is (L-OUT) then $\bar{t}.P \xrightarrow{\bar{t}} P$. By Lemma A.2 Case 2) we get $\Gamma; \emptyset \vdash_{\gamma \cup \{t\};\delta;\bot} P$.
- ***Base case 2:*** Assume that $\Gamma; \Delta \vdash_{\gamma;\delta;p} \bar{a}.P$ and if the last applied rule is (L-OUT) then $\bar{a}.P \xrightarrow{\bar{a}} P$. By Lemma A.2 Case 3) it holds that $\Gamma; \emptyset \vdash_{\gamma;\delta;\bot} P$.
- ***Base case 3:*** Assume that $\Gamma; \Delta \vdash_{\gamma;\delta;p} a.P$ and if the last applied rule is (L-IN) then $a.P \xrightarrow{a} P$. By Lemma A.2 Case 4) we have that $\Gamma; \emptyset \vdash_{\gamma;\delta;\bot} P$.
- ***Base case 4:*** Assume that $\Gamma; \Delta \vdash_{\gamma;\delta;p} t[P,Q]$ and if the last applied rule is (L-REC-OUT) then $t[P,Q] \xrightarrow{t} \mathsf{extr}(P) \mid \langle Q \rangle$.
  - By Lemma A.3 there is $\Gamma', \Delta', \gamma', \delta'$ and p' such that $\Delta' \subseteq \Delta$ and the following holds: $\Gamma'; \Delta' \vdash_{\gamma';\delta';p'} P \mid \langle Q \rangle$.
  - By Lemma A.4 and Lemma A.5 we finally get: $\Gamma_1'; \Delta_1' \vdash_{\gamma_1';\delta_1';p_1'} \mathsf{extr}(P) \mid \langle Q \rangle$ where $\Gamma_1' \subseteq \Gamma, \Delta_1' \subseteq \Delta$ and $\gamma_1' \subseteq \gamma$.

Induction step:

- ***Case 1:*** Assume that $\Gamma; \Delta \vdash_{\gamma;\delta;p} P_1 \mid Q$ and if the last applied rule is (L-PAR1) then $P_1 \xrightarrow{\alpha} P_1'$ and $P_1 \mid Q \xrightarrow{\alpha} P_1' \mid Q$.
  - By Lemma A.2 Case 2) the following holds: $\Gamma_1; \Delta_1 \vdash_{\gamma_1;\delta_1;p_1} P_1$ and $\Gamma_2; \Delta_2 \vdash_{\gamma_2;\delta_2;p_2} Q$ such that $\Gamma = \Gamma_1 \cup \Gamma_2 \cup (\gamma_1 \times \gamma_2)$ (cf. (7)), $\Delta = \Delta_1 \cup \Delta_2, \gamma = \gamma_1 \cup \gamma_2, \delta = \delta_1 \cup \delta_2$ and $p = p_1 \vee p_2$ and condition $\Gamma^s \cap \Delta^t = \emptyset$ holds.
  - By induction hypothesis there are $\Gamma_1' \subseteq \Gamma_1$ and $\Delta_1' \subset \Delta_1$, such that $\Gamma_1'; \Delta_1' \vdash_{\gamma_1';\delta_1';p_1'} P'$.
  - We get that condition $(\Delta_1' \cup \Delta_2)^t \cap (\Gamma_1' \cup \Gamma_2 \cup (\gamma_1' \times \gamma_2))^s = \emptyset$ holds based on basic properties of set operations.
  - For processes P_1' and Q by formation on Rule (W-PAR) we get: $\Gamma'; \Delta' \vdash_{\gamma';\delta';p'} P_1' \mid Q$ where $\Gamma' = \Gamma_1' \cup \Gamma_2 \cup (\gamma_1' \times \gamma_2), \Delta = \Delta_1' \cup \Delta_2, \gamma = \gamma_1' \cup \gamma_2, \delta = \delta_1' \cup \delta_2$ and $p = p_1' \vee p_2$.
- ***Case 2:*** Assume that $\Gamma; \Delta \vdash_{\gamma;\delta;p} P \mid Q$ and if the last applied rule is (L-COMM1) then $P \xrightarrow{x} P'$ and $Q \xrightarrow{\bar{x}} Q'$ and $P \mid Q \xrightarrow{\tau} P' \mid Q'$.
  - By Lemma A.2 Case 2) the following holds: $\Gamma_1; \Delta_1 \vdash_{\gamma_1;\delta_1;p_1} P$ and $\Gamma_2; \Delta_2 \vdash_{\gamma_2;\delta_2;p_2} Q$ such that $\Gamma = \Gamma_1 \cup \Gamma_2 \cup (\gamma_1 \times \gamma_2), \Delta = \Delta_1 \cup \Delta_2, \gamma = \gamma_1 \cup \gamma_2, \delta = \delta_1 \cup \delta_2$ and $p = p_1 \vee p_2$ and condition $\Gamma^s \cap \Delta^t = \emptyset$ holds.
  - For process P' by induction hypothesis there are $\Gamma_1' \subseteq \Gamma_1$ and $\Delta_1' \subset \Delta_1$, such that $\Gamma_1'; \Delta_1' \vdash_{\gamma_1';\delta_1';p_1'} P'$.
  - For process Q' by induction hypothesis there are $\Gamma_2' \subseteq \Gamma_2$ and $\Delta_2' \subset \Delta_2$, such that $\Gamma_2'; \Delta_2' \vdash_{\gamma_2';\delta_2';p_2'} Q'$.
  - We get that condition $(\Delta_1' \cup \Delta_2')^t \cap (\Gamma_1' \cup \Gamma_2' \cup (\gamma_1' \times \gamma_2'))^s = \emptyset$ holds based on basic properties of set operations.
  - For processes P' and Q' by formation on Rule (W-PAR) we get: $\Gamma'; \Delta' \vdash_{\gamma';\delta';p'} P' \mid Q'$ where $\Gamma' = \Gamma_1' \cup \Gamma_2' \cup (\gamma_1' \times \gamma_2'), \Delta = \Delta_1' \cup \Delta_2', \gamma = \gamma_1' \cup \gamma_2', \delta = \delta_1' \cup \delta_2'$ and $p = p_1' \vee p_2'$.
- ***Case 3:*** Assume that $\gamma \times \gamma; \Delta \vdash_{\gamma;\delta;p} !\pi.P$ and if the last applied rule is (L-REP) then $\pi.P \xrightarrow{\alpha} P'$ and $!\pi.P \xrightarrow{\alpha} P' \mid !\pi.P$.
  - By Lemma A.2 we get $\Gamma'; \emptyset \vdash_{\gamma;\emptyset;\bot} \pi.P$ and $\Delta = \emptyset$.
  - By induction hypothesis there is $\Gamma_1' \subseteq \Gamma'$ such that $\Gamma_1'; \emptyset \vdash_{\gamma';\delta';p'} P'$.

- For processes $!\pi.P$ and P' by formation on Rule (W-Par) we get: $\Gamma_1; \Delta_1 \vdash_{\gamma_1;\delta_1;p_1} P' \mid !\pi.P$ such that $\Gamma_1 = \Gamma_1' \cup (\gamma \times \gamma_1)$, $\Delta_1 = \emptyset$, $\gamma_1 = \gamma \cup \gamma'$, $\delta_1 = \delta \cup \delta'$ and $p_1 = p \vee p'$. Condition $\Gamma_1^s \cap \Delta_1^t = \emptyset$ holds by basic properties of set operations.

- ***Case 4:*** Assume that $\Gamma; \Delta \vdash_{\gamma;\delta;p} t[P, Q]$ and if the last applied rule is (L-Rec-In) then $P \xrightarrow{\bar{t}} P'$ and $t[P, Q] \xrightarrow{\tau}$ $\mathsf{extr}(P') \mid \langle Q \rangle$.
  - Using assumption and by Lemma A.2 we get that for process P the following holds: $\Gamma_1; \Delta_1 \vdash_{\gamma_1;\delta_1;p_1} P$ for some $\Gamma_1, \Delta_1, \gamma_1, \delta_1, p_1$.
  - By induction hypothesis there are $\Gamma_1' \subseteq \Gamma_1$, $\Delta_1' \subseteq \Delta_1$ such that $\Gamma_1'; \Delta_1' \vdash_{\gamma_1';\delta_1';p_1'} P'$.
  - For transaction $t[P, Q]$ by Lemma A.3 there are $\Delta' \subseteq \Delta$ and $\Gamma'; \Delta' \vdash_{\gamma';\delta';p'} P \mid \langle Q \rangle$.
  - For process P' by formation on Rule (W-Par) and by set operations' properties the following holds: $\Gamma_2; \Delta_2 \vdash_{\gamma_2;\delta_2;p_2} P' \mid \langle Q \rangle$.
  - Applying Lemma A.4 on process $\mathsf{extr}(P')$ and Applying Lemma A.5 we finally get: $\Gamma_2'; \Delta_2' \vdash_{\gamma_2';\delta_2';p_2'} \mathsf{extr}(P') \mid \langle Q \rangle$ where $\Gamma_2' \subseteq \Gamma_2$, $\Delta_2' \subseteq \Delta_2$ and $\gamma_2' \subseteq \gamma_2$.
- ***Case 5:*** Assume that $\Gamma; \Delta \vdash_{\gamma;\{t\};p} t[P, Q]$ and if the last applied rule is (L-Scope-Out) then $P \xrightarrow{\alpha} P'$ and $t[P, Q] \xrightarrow{\alpha}$ $t[P', Q]$.
  - By Lemma A.2 we get: $\Gamma_1; \Delta_1 \vdash_{\gamma_1;\delta_1;p_1} P$ and $\Gamma_2; \Delta_2 \vdash_{\gamma_2;\delta_2;p_2} Q$ and $\Gamma = \Gamma_1 \cup \Gamma_2 \cup (\gamma_1 \times \gamma_2)$, $\Delta = \Delta_1 \cup \Delta_2 \cup (\{t\} \times (\delta_1 \cup \delta_2 \cup \gamma_1 \cup \gamma_2))$, $\gamma = \gamma_1 \cup \gamma_2$, $\delta = \{t\}$ and $p = p_1 \vee p_2$, also condition $\Gamma^s \cap \Delta^t = \emptyset$ holds.
  - By inductive hypothesis there are $\Gamma_1' \subseteq \Gamma_1$, $\Delta_1' \subseteq \Delta_1$ and $\delta_1', \gamma_1', p_1'$ such that $\Gamma_1'; \Delta_1' \vdash_{\gamma_1';\delta_1';p_1'} P'$.
  - Based on set operations' properties for $\Gamma = \Gamma_1' \cup \Gamma_2 \cup (\gamma_1 \times \gamma_2)$ and $\Delta = \Delta_1' \cup \Delta_2 \cup (\{t\} \times (\delta_1 \cup \delta_2 \cup \gamma_1 \cup \gamma_2))$ the condition $\Gamma^s \cap \Delta^t = \emptyset$ holds too.
  - For process Q and obtained process P' by formation on Rule (W-Trans) we get: $\Gamma'; \Delta' \vdash_{\gamma';\{t\};p'} t[P', Q]$, where $\Gamma' = \Gamma_1' \cup \Gamma_2 \cup (\gamma_1' \times \gamma_2)$, $\Delta' = \Delta_1' \cup \Delta_2 \cup (\{t\} \times (\delta_1 \cup \delta_2 \cup \gamma_1 \cup \gamma_2))$, $\gamma = \gamma_1' \cup \gamma_2$, $\delta = \{t\}$ and $p = p_1' \vee p_2$.
- ***Case 6:*** Assume that $\Gamma; \Delta \vdash_{\gamma;\delta;p} (\nu x)P$ and if the last applied rule is (L-Res) then $P \xrightarrow{\alpha} P'$ and $(\nu x)P \xrightarrow{\alpha} (\nu x)P'$.
  - By Lemma A.2 we get: $\Gamma; \Delta \vdash_{\gamma;\delta;p} P$.
  - By inductive hypothesis there are $\Gamma' \subseteq \Gamma$, $\Delta' \subseteq \Delta$, γ', δ', p' such that $\Gamma'; \Delta' \vdash_{\gamma';\delta';p'} P'$.
  - For process P' by formating on Rule (W-Res) the following holds: $\Gamma'; \Delta' \vdash_{\gamma';\delta';p'} (\nu x)P'$.
- ***Case 7:*** Assume that $\Gamma; \Delta \vdash_{\gamma;\delta;\top} \langle P \rangle$ and if the last applied rule is (L-Block) then $P \xrightarrow{\alpha} P'$ and $\langle P \rangle \xrightarrow{\alpha} \langle P' \rangle$.
  - By Lemma A.2 we get: $\Gamma; \Delta \vdash_{\gamma;\delta;p} P$.
  - By inductive hypothesis there are $\Gamma' \subseteq \Gamma$, $\Delta' \subseteq \Delta$, γ', δ', p' such that $\Gamma'; \Delta' \vdash_{\gamma';\delta';p'} P'$.
  - For process P' by formating on Rule (W-Block) the following holds: $\Gamma'; \Delta' \vdash_{\gamma';\delta';p'} \langle P \rangle'$. □

In the following, we provide proofs that both encodings are valid, i.e. we prove that compositionality, name invariance operational correspondence, divergence reflection and success sensitiveness are satisfied. We separate these results into two sections, one section for results related to the encoding of $\mathcal{C}$ into $\mathcal{S}$ and the other one for the encoding of $\mathcal{C}$ into $\mathcal{O}$.

Appendix B. Results related to encoding of $\mathcal{C}$ into $\mathcal{S}$ (§ 5)

B.1. Proof of compositionality results: Theorem 5.3

We repeat the statement in page 16:

Theorem 5.3 *(Compositionality for $[\![\cdot]\!]_\rho$). Let ρ be an arbitrary path. For every process operator in $\mathcal{C}$ and for all well-formed compensable processes P and Q it holds that:*

$$\begin{array}{lll}
[\![\langle P\rangle]\!]_\rho = C_{\langle\rangle,\rho}\,[[\![P]\!]_\varepsilon] & [\![t[P,Q]]\!]_\rho = C_{t[,],\rho}\left[[\![P]\!]_{t,\rho}, [\![Q]\!]_\varepsilon\right] & [\![P \mid Q]\!]_\rho = C_{\mid}\left[[\![P]\!]_\rho, [\![Q]\!]_\rho\right] \\
[\![a.P]\!]_\rho = C_{a.}\left[[\![P]\!]_\rho\right] & [\![\overline{t}.P]\!]_\rho = C_{\overline{t}.}\left[[\![P]\!]_\rho\right] & [\![(\nu x)P)]\!]_\rho = C_{(\nu x)}\left[[\![P]\!]_\rho\right] \\
[\![\overline{a}.P]\!]_\rho = C_{\overline{a}.}\left[[\![P]\!]_\rho\right] & [\![!\pi.P]\!]_\rho = C_{!\pi.}\left[[\![P]\!]_\rho\right] &
\end{array}$$

Proof. Follows directly from the definition of contexts (Definition 5.6) and from the definition of $[\![\cdot]\!]_\rho : \mathcal{C} \longrightarrow \mathcal{S}$ (Fig. 6). Indeed, for all operators and all well-formed compensable processes P and Q we have:

$$[\![P \mid Q]\!]_\rho = C_{\mid}\left[[\![P]\!]_\rho, [\![Q]\!]_\rho\right] = [\![P]\!]_\rho \mid [\![Q]\!]_\rho$$
$$[\![t[P,Q]]\!]_\rho = C_{t[,],\rho}\left[[\![P]\!]_{t,\rho}, [\![Q]\!]_\varepsilon\right] = t\left[[\![P]\!]_{t,\rho}\right] \mid t.\left(\mathtt{extr}\langle\!\langle t, p_{t,\rho}, p_\rho\rangle\!\rangle \mid p_\rho[[\![Q]\!]_\varepsilon]\right)$$

$$\begin{aligned}
[\![\langle P\rangle]\!]_\rho &= C_{\langle\rangle,\rho}\,[[\![P]\!]_\varepsilon] = p_\rho\big[[\![P]\!]_\varepsilon\big]\\
[\![a.P]\!]_\rho &= C_{a.}\,\big[[\![P]\!]_\rho\big] = a.[\![P]\!]_\rho\\
[\![\overline{a}.P]\!]_\rho &= C_{\overline{a}.}\,\big[[\![P]\!]_\rho\big] = \overline{a}.[\![P]\!]_\rho\\
[\![\overline{t}.P]\!]_\rho &= C_{\overline{t}.}\,\big[[\![P]\!]_\rho\big] = \overline{t}.h_t.[\![P]\!]_\rho\\
[\![(\nu x)P)]\!]_\rho &= C_{(\nu x)}\,\big[[\![P]\!]_\rho\big] = (\nu x)[\![P]\!]_\rho\\
[\![!\pi.P]\!]_\rho &= C_{!\pi.}\,\big[[\![\pi.P]\!]_\rho\big] = ![\![\pi.P]\!]_\rho. \quad \square
\end{aligned}$$

B.2. Proof of name invariance results: Theorem 5.4

We repeat the statement in page 16:

Theorem 5.4 *(Name invariance for $[\![\cdot]\!]_\rho$). For every well-formed compensable process P and valid substitution $\sigma : \mathcal{N}_c \to \mathcal{N}_c$ there is a $\sigma' : \mathcal{N}_a \longrightarrow \mathcal{N}_a$ such that:*

$$(i)\text{ for every } x \in \mathcal{N}_c : \varphi_{[\![\cdot]\!]_{\sigma(\rho)}}(\sigma(x)) = \{\sigma'(y) : y \in \varphi_{[\![\cdot]\!]_\rho}(x)\} \qquad \text{and} \qquad (ii)\ [\![\sigma(P)]\!]_{\sigma(\rho)} = \sigma'([\![P]\!]_\rho).$$

Proof. We define the substitution σ' as follows:

$$\sigma'(x) = \begin{cases} \sigma(x) & \text{if } x = a \text{ or } x = t\\ h_{\sigma(t)} & \text{if } x = h_t\\ p_{\sigma(\rho)} & \text{if } x = p_\rho. \end{cases} \tag{B.1}$$

Now we provide proofs for (i) and (ii):

(i) Since $\mathcal{N}_c = \mathcal{N}_t \cup \mathcal{N}_s$, we consider two sub-cases for x:
- if $x \in \mathcal{N}_s$ then it follows that:
  $\{\sigma'(y) : y \in \varphi_{[\![\cdot]\!]_\rho}(x)\} = \{\sigma'(y) : y \in \{x\}\} = \{\sigma'(x)\} = \{\sigma(x)\} = \varphi_{[\![\cdot]\!]_\rho}(\sigma(x))$.
- if $x \in \mathcal{N}_t$ then:
  - by Definition 5.5: $\varphi_{[\![\cdot]\!]_{\sigma(\rho)}}(\sigma(x)) = \{\sigma(x), h_{\sigma(x)}\} \cup \{p_{\sigma(\rho)} : \sigma(x) \in \sigma(\rho)\}$
  - by definition of σ':
    $\{\sigma(x), h_{\sigma(x)}\} \cup \{p_{\sigma(\rho)} : \sigma(x) \in \sigma(\rho)\} = \{\sigma'(x), \sigma'(h_x)\} \cup \{\sigma'(p_\rho) : \sigma'(x) \in \sigma'(\rho)\} = \{\sigma'(y) : y \in \{x, h_x\}\} \cup \{\sigma'(y) : y \in \{p_\rho : \sigma(x) \in \sigma(\rho)\}\} = \{\sigma'(y) : y \in \varphi_{[\![\cdot]\!]_\rho}(x)\}$.

(ii) The proof proceeds by structural induction on P. In the following, given a name x, a path ρ, and process P, we write $\sigma x, \sigma\rho$, and σP to stand for $\sigma(x)$, $\sigma(\rho)$, and $\sigma(P)$, respectively.
Base case: The statement holds for $P = \mathbf{0}$: $[\![\sigma(\mathbf{0})]\!]_{\sigma\rho} = \sigma'\big([\![\mathbf{0}]\!]_\rho\big) \Leftrightarrow \mathbf{0} = \mathbf{0}$.
Inductive step: There are six cases, but we content ourselves by showing the following three cases: transaction scope, protected block, and input/output prefix. The proof for all the other cases proceeds similarly.
- ***Case 1:*** Assume that $P = t[P_1, Q_1]$. We first apply the substitution σ on process P:

$$[\![\sigma(t[P_1,Q_1])]\!]_{\sigma\rho} = [\![\sigma t[\sigma(P_1),\sigma(Q_1)]]\!]_{\sigma\rho}.$$

By expanding the definition of the translation in Definition 5.5, we have:

$$[\![\sigma(t[P_1,Q_1])]\!]_{\sigma\rho} = \sigma t\big[[\![\sigma(P_1)]\!]_{\sigma t,\sigma\rho}\big] \mid \sigma t.\big(\mathtt{extr}\langle\!\langle \sigma t, p_{\sigma t,\sigma\rho}, p_{\sigma\rho}\rangle\!\rangle \mid p_{\sigma\rho}[[\![\sigma(Q_1)]\!]_\varepsilon]\big)$$

By induction hypothesis it follows:

$$[\![\sigma(t[P_1,Q_1])]\!]_{\sigma\rho} = \sigma t\big[\sigma'\big([\![P_1]\!]_{t,\rho}\big)\big] \mid \sigma t.\big(\mathtt{extr}\langle\!\langle \sigma t, p_{\sigma t,\sigma\rho}, p_{\sigma\rho}\rangle\!\rangle \mid p_{\sigma\rho}[\sigma'\big([\![Q_1]\!]_\varepsilon\big)]\big) \tag{B.2}$$

On the other side, when we apply definition of substitution σ' on $[\![P]\!]_\rho$ the following holds:

$$\begin{aligned}\sigma'\big([\![t[P_1,Q_1]]\!]_\rho\big) &= \sigma'\big(t\big[[\![P_1]\!]_{t,\rho}\big] \mid t.\big(\mathtt{extr}\langle\!\langle t, p_{t,\rho}, p_\rho\rangle\!\rangle \mid p_\rho[[\![Q_1]\!]_\varepsilon]\big)\big)\\ &= \sigma' t\big[\sigma'\big([\![P_1]\!]_{t,\rho}\big)\big] \mid \sigma' t.(\mathtt{extr}\langle\!\langle \sigma' t, p_{\sigma' t,\sigma'\rho}, p_{\sigma'\rho}\rangle\!\rangle \mid p_{\sigma'\rho}[\sigma'\big([\![Q_1]\!]_\varepsilon\big)]).\end{aligned} \tag{B.3}$$

Given that it is valid $\sigma'(t) = \sigma(t)$ (cf. (B.1)), it is easy to conclude that (B.2) is equal to (B.3).
- ***Case 2:*** Assume that $P = \langle P_1\rangle$. We apply substitution σ on process P:

$$[\![\sigma(\langle P_1\rangle)]\!]_{\sigma\rho} = [\![\langle\sigma(P_1)\rangle]\!]_{\sigma\rho}$$

By Definition 5.5, $[\![\sigma(\langle P_1\rangle)]\!]_{\sigma\rho} = p_{\sigma\rho}\big[[\![\sigma(P_1)]\!]_\varepsilon\big]$, and by induction hypothesis:

$$[\![\sigma(\langle P_1\rangle)]\!]_{\sigma\rho} = p_{\sigma\rho}\big[\sigma'([\![P_1]\!]_{\varepsilon})\big]. \tag{B.4}$$

On the other side, when we apply substitution σ' on $[\![P]\!]_\rho$ the following holds:

$$\sigma'([\![\langle P_1\rangle]\!]_\rho) = \sigma'(p_\rho\big[[\![P_1]\!]_\varepsilon\big]) = p_{\sigma'\rho}\big[\sigma'([\![P_1]\!]_\varepsilon)\big]. \tag{B.5}$$

Based on definition of the function σ', i.e. $\sigma'(p_\rho) = p_{\sigma(\rho)}$ and $\sigma'(t) = \sigma(t)$ (cf. (B.1)), it is easy to conclude that (B.4) is equal to (B.5).

- ***Case 3:*** Here we distinguish two sub-cases. In the first sub-case we consider input on name $a \in \mathcal{N}_s$ (proof follows similarly for output). In the second sub-case we consider that the output message is an error notification on name $t \in \mathcal{N}_t$.
  - ***Case 3a:*** Assume that $P = a.P_1$. We apply substitution σ on process P:

    $$[\![\sigma(a.P_1)]\!]_{\sigma\rho} = [\![\sigma a.\sigma(P_1)]\!]_{\sigma\rho}.$$

    Next, we apply Definition 5.5: $[\![\sigma(a.P_1)]\!]_{\sigma\rho} = \sigma a.[\![\sigma(P_1)]\!]_{\sigma\rho}$. By induction hypothesis it follows:

    $$[\![\sigma(a.P_1)]\!]_{\sigma\rho} = \sigma a.\sigma'([\![P_1]\!]_\rho). \tag{B.6}$$

    We now apply substitution σ' on $[\![P]\!]_\rho$:

    $$\sigma'([\![(a.P_1)]\!]_\rho) = \sigma'(a.[\![P_1]\!]_\rho) = \sigma' a.\sigma'([\![P_1]\!]_\rho). \tag{B.7}$$

    By definition of σ' (cf. (B.1)), $\sigma'(a) = \sigma(a)$ and so we conclude that (B.6) is equal to (B.7).
  - ***Case 3b:*** Assume that $P = \bar{t}.P_1$. We apply substitution σ on process P:

    $$[\![\sigma(\bar{t}.P_1)]\!]_{\sigma\rho} = [\![\sigma\bar{t}.\sigma(P_1)]\!]_{\sigma\rho}.$$

    Next, we apply Definition 5.5: $[\![\sigma(\bar{t}.P_1)]\!]_{\sigma\rho} = \sigma\bar{t}.h_{\sigma t}.[\![\sigma(P_1)]\!]_{\sigma\rho}$. By induction hypothesis:

    $$[\![\sigma(\bar{t}.P_1)]\!]_{\sigma\rho} = \sigma\bar{t}.h_{\sigma t}.\sigma'([\![P_1]\!]_\rho). \tag{B.8}$$

    We apply substitution σ' on $[\![P]\!]_\rho$:

    $$\sigma'([\![(\bar{t}.P_1)]\!]_\rho) = \sigma'(\bar{t}.h_t.[\![P_1]\!]_\rho) = \sigma'\bar{t}.h_{\sigma' t}.\sigma'([\![P_1]\!]_\rho). \tag{B.9}$$

    By definition of σ' (cf. (B.1)), $\sigma'(a) = \sigma(a)$ and so we conclude that (B.8) is equal to (B.9). □

B.3. *Proof of operational correspondence results: Theorem 5.8*

We now shall prove that the translation $[\![\cdot]\!]_\rho$ satisfies operational correspondence (completeness and soundness). The statement and its proof are presented in Appendix B.3.4 (page 54). We first present an overview to the proof and some auxiliary results.

B.3.1. *A roadmap for the proofs*

Part (1) of Theorem 5.8 is ***completeness***, i.e.,

$$\text{If } P \longrightarrow P' \text{ then } [\![P]\!]_\varepsilon \longrightarrow^k [\![P']\!]_\varepsilon$$

where $k \geq 1$ is given precisely by our statement. This property ensures that our translation faithfully simulates the behavior of compensable processes. The proof is by induction on the derivation of $P \longrightarrow P'$ and uses:

- Definition 5.5 (page 14), i.e., the definition of $[\![\cdot]\!]_\rho$;
- Proposition 3.1 (page 7) for three base cases; as key to proving the operational correspondence.
- Lemma B.1 (page 43), which maps evaluation contexts in $\mathcal{C}$ into evaluation contexts of $\mathcal{S}$;
- Lemma B.6 (page 45), which shows that $\mathtt{ch}(t, [\![P]\!]_\rho) = \mathbf{0}$ for all $[\![P]\!]_\rho$. Its proof uses two auxiliary properties of translated terms: Lemma B.2 (page 44) and Lemma B.5 (page 45);
- Definition B.2 and Definition B.3 (page 46 and page 49). These definitions formalize the intermediate processes that appear during derivation.

Part (2) of Theorem 5.8 is ***soundness***, i.e.,

$$\text{If } [\![P]\!]_\varepsilon \longrightarrow^n R \text{ then there is } P' \text{ such that } P \longrightarrow^* P' \text{ and } R \longrightarrow^* [\![P']\!]_\varepsilon$$

This property ensures that target terms never exhibit behavior that can not be attributed to some compensable process. As usual, proving soundness is more challenging than proving completeness. Our proof is by induction on n, i.e., the length of the reduction $[\![P]\!]_\rho \longrightarrow^n R$. We rely on several auxiliary results:

- Lemma B.10 (page 50) is about the shape of process R, and also ensures that there is a process P' with an appropriate shape. The proof proceeds by induction on n. The base case uses Lemma B.4 (page 44); in the inductive step, we exploit that the target term R_1 has a specific shape, which is in turn ensured by Lemma B.8 (page 46) and Lemma B.9 (page 49);
- To prove Lemma B.4 we use Lemma B.2, Corollary B.3 (page 44) and Definition 5.5;
- In the statement of Lemma B.8 and Lemma B.9 we use the definition of intermediate processes given by Definition B.2 and Definition B.3, respectively. The proofs proceed by case analysis for the step $R \longrightarrow R'$, using Lemma B.7;
- Lemma B.11 (page 54) ensures that the adaptable process obtained thanks to Lemma B.8 and Lemma B.9 can evolve until reaching a process that corresponds to the translation of a compensable process.

Using these guidelines as a proof sketch, we now introduce all the ingredients of the proof in full detail.

The same road map, with modified definitions, lemmas, and theorems for translation $[\![\cdot]\!]^\circ_\rho$, will be used also for the proof of operational correspondence for the translation with the objective update, i.e., Theorem 6.5.

B.3.2. Auxiliary results for completeness

To simplify proofs of correctness, we start by defining a mapping of evaluation contexts for compensable processes (cf. Definition 3.3) into evaluation contexts for adaptable processes (cf. Definition 3.6):

Definition B.1. Let ρ be a path. We define the following mapping $[\![\cdot]\!]_\rho$ from evaluation contexts of compensable processes into evaluation contexts of adaptable processes:

$$[\![[\bullet]]\!]_\rho = [\bullet] \qquad [\![\langle C[\bullet]\rangle]\!]_\rho = p_\rho[[\![C[\bullet]]\!]_\varepsilon] \qquad [\![C[\bullet] \mid P]\!]_\rho = [\![C[\bullet]]\!]_\rho \mid [\![P]\!]_\rho \qquad [\![(\nu x)C[\bullet]]\!]_\rho = (\nu x)[\![C[\bullet]]\!]_\rho$$
$$[\![t[C[\bullet], Q]]\!]_\rho = t\big[[\![C[\bullet]]\!]_{t,\rho}\big] \mid t.\big(\mathtt{extr}\langle\!\langle t, p_{t,\rho}, p_\rho\rangle\!\rangle \mid p_\rho[[\![Q]\!]_\varepsilon]\big)$$

Lemma B.1. *Let P be a well-formed compensable process, $C[\bullet]$ an evaluation context, ρ an arbitrary path, and ρ' the path to the hole in $C[\bullet]$. Then, $[\![C[P]]\!]_\rho = [\![C[\bullet]]\!]_\rho[[\![P]\!]_{\rho'}]$.*

Proof. The proof proceeds by induction on the structure of $C[\bullet]$.

<u>Base cases:</u> Assume that $C[\bullet] = [\bullet]$ then $C[P] = P$, and $[\![C[P]]\!]_\rho = [\![P]\!]_\rho$.

<u>Inductive step:</u> There are four cases to consider. They all proceed by Definition 5.5, Definition B.1, and the inductive hypothesis:

- ***<u>Case 1:</u>*** Assume that $C[\bullet] = \langle C_1[\bullet]\rangle$ then $C[P] = \langle C_1[P]\rangle$.

$$\begin{aligned}[\![C[P]]\!]_\rho &= [\![\langle C_1[P]\rangle]\!]_\rho \overset{\text{Def.}}{=} p_\rho[[\![C_1[P]]\!]_\varepsilon] \overset{\text{I.H.}}{=} p_\rho[[\![C_1[\bullet]]\!]_\varepsilon[[\![P]\!]_{\rho'}]] = p_\rho[[\![C_1[\bullet]]\!]_\varepsilon]\big[[\![P]\!]_{\rho'}\big]\\ &\overset{\text{Def.}}{=} [\![C[\bullet]]\!]_\rho\big[[\![P]\!]_{\rho'}\big]\end{aligned}$$

- ***<u>Case 2:</u>*** Assume that $C[\bullet] = C_1[\bullet] \mid Q$ then $C[P] = C_1[P] \mid Q$.

$$\begin{aligned}[\![C[P]]\!]_\rho &= [\![C_1[P] \mid Q]\!]_\rho \overset{\text{Def.}}{=} [\![C_1[P]]\!]_\rho \mid [\![Q]\!]_\rho \overset{\text{I.H.}}{=} [\![C_1[\bullet]]\!]_\rho[[\![P]\!]_{\rho'}] \mid [\![Q]\!]_\rho\\ &= [\![C_1[\bullet] \mid Q]\!]_\rho[[\![P]\!]_{\rho'} = [\![C[\bullet]]\!]_\rho\big[[\![P]\!]_{\rho'}\big]\end{aligned}$$

- ***<u>Case 3:</u>*** Assume that $C[\bullet] = t[C_1[\bullet], Q]$ then $C[P] = t[C_1[P], Q]$.

$$\begin{aligned}[\![C[P]]\!]_\rho &= [\![t[C_1[P], Q]]\!]_\rho \overset{\text{Def.}}{=} t\big[[\![C_1[P]]\!]_{t,\rho}\big] \mid t.\big(\mathtt{extr}\langle\!\langle t, p_{t,\rho}, p_\rho\rangle\!\rangle \mid p_\rho[[\![Q]\!]_\varepsilon]\big)\\ &\overset{\text{I.H.}}{=} t\big[[\![C_1[\bullet]]\!]_{t,\rho}[[\![P]\!]_{\rho'}]\big] \mid t.\big(\mathtt{extr}\langle\!\langle t, p_{t,\rho}, p_\rho\rangle\!\rangle \mid p_\rho[[\![Q]\!]_\varepsilon]\big)\\ &= \big(t\big[[\![C_1[\bullet]]\!]_{t,\rho}\big] \mid t.\big(\mathtt{extr}\langle\!\langle t, p_{t,\rho}, p_\rho\rangle\!\rangle \mid p_\rho[[\![Q]\!]_\varepsilon]\big)\big)\big[[\![P]\!]_{\rho'}\big]\\ &= [\![C[\bullet]]\!]_\rho\big[[\![P]\!]_{\rho'}\big]\end{aligned}$$

- ***<u>Case 4:</u>*** Assume that $C[\bullet] = (\nu x)C_1[\bullet]$ then $C[P] = (\nu x)C_1[P]$.

$$[\![C[P]]\!]_\rho = [\![(\nu x)C_1[P]]\!]_\rho \overset{\text{Def.}}{=} (\nu x)[\![C_1[P]]\!]_\rho \overset{\text{I.H.}}{=} (\nu x)[\![C_1[\bullet]]\!]_\rho[[\![P]\!]_{\rho'}] = [\![C[\bullet]]\!]_\rho\big[[\![P]\!]_{\rho'}\big]. \quad \square$$

B.3.3. Auxiliary results for soundness

For the proof of soundness, we will need the converse of Lemma B.1, which is stated by the following two results.

Lemma B.2. *Let P be a well-formed compensable process and ρ a path. If $[\![P]\!]_\rho \equiv C[P']$ then there are $C_1[\bullet]$ and P_1 such that $C[\bullet] = [\![C_1[\bullet]]\!]_\rho$ and $P' = [\![P_1]\!]_{\rho'}$, where ρ' is the path to the hole in $C_1[\bullet]$.*

Proof. The proof is by induction on structure of context $C[\bullet]$.

Base case: If $C[\bullet] = [\bullet]$ and $[\![P]\!]_\rho = P'$ then it follows directly that $C_1[\bullet] = [\bullet]$ and $P_1 = P$.

Inductive step:

- ***Case 1:*** $C[\bullet] = l[C'[\bullet]]$ and $[\![P]\!]_\rho \equiv l[C'[P']]$.
  By Definition 5.5, we have that $l = p_\rho$ and there is P'_1 such that $[\![P'_1]\!]_\rho = C'[P']$. By the induction hypothesis, there are $C'_1[\bullet]$ and P_1 such that $C'[\bullet] = [\![C'_1[\bullet]]\!]_\rho$ and $P' = [\![P_1]\!]_{\rho'}$, where ρ' is the path to the hole in $C'_1[\bullet]$. By Definition B.1, $C[\bullet] = p_\rho[[\![C'_1[\bullet]]\!]_\rho] = [\![\langle C'_1[\bullet]\rangle]\!]_\rho$, and hence $C_1[\bullet] = \langle C'_1[\bullet]\rangle$.
- ***Case 2:*** $C[\bullet] = (\nu x)C'[\bullet]$ and $[\![P]\!]_\rho \equiv (\nu x)C'[P']$.
  By Definition 5.5, there is P'_1 such that $[\![P'_1]\!]_\rho = C'[P']$. By induction hypothesis, there are $C'_1[\bullet]$ and P_1 such that $C'[\bullet] = [\![C'_1[\bullet]]\!]_\rho$ and $P' = [\![P_1]\!]_{\rho'}$, where ρ' is the path to the hole in $C'_1[\bullet]$. Now, we have that $C[\bullet] = (\nu x)[\![C'_1[\bullet]]\!]_\rho = [\![(\nu x)C'_1[\bullet]]\!]_\rho$ and hence $C_1[\bullet] = (\nu x)C'_1[\bullet]$.
- ***Case 3:*** $C[\bullet] = C'[\bullet] \mid Q$ and $[\![P]\!]_\rho \equiv C'[P'] \mid Q$.
  By Definition 5.5, we have two possibilities:
  (i) If $Q = t.\big(\mathtt{extr}\langle\!\langle t, p_{t,\rho}, p_\rho\rangle\!\rangle \mid p_\rho[[\![Q']\!]_\varepsilon]\big)$ and $C'[P'] \equiv t[[\![P'_1]\!]_{t,\rho}]$ for some t, P'_1, Q', then $[\![P'_1]\!]_{t,\rho} = C'_1[P']$ for some C'_1, P'. By induction hypothesis, there are $C''_1[\bullet]$ and P_1 such that $C'_1[\bullet] = [\![C''_1]\!]_\rho$ and $P' = [\![P_1]\!]_{\rho'}$, where ρ' is the path to the hole in $C''_1[\bullet]$. We complete the proof by choosing $C_1[\bullet] = t[C''_1[\bullet], Q']$ and $P' = [\![P_1]\!]_{t,\rho'}$.
  (ii) If $C'[P'] = [\![Q_1]\!]_\rho$ and $Q = [\![Q_2]\!]_\rho$ for some Q_1, Q_2, then, by induction hypothesis, there are $C'_1[\bullet]$ and P_1 such that $C'[\bullet] = [\![C'_1[\bullet]]\!]_\rho$ and $P' = [\![P_1]\!]_{\rho'}$, where ρ' is the path to the hole in $C'_1[\bullet]$. In this case, $C_1[\bullet] = C'_1[\bullet] \mid Q_2$. □

As a direct consequence of Case 3 in the previous proof, we can identify two possibilities for a process that is obtained via our translation and equals to a parallel composition of processes.

Corollary B.3. *Let P be a well-formed compensable process and ρ a path.*
If $[\![P]\!]_\rho \equiv C[P'] \mid D[Q']$ then either:

(i) *there are $C_1[\bullet]$, $D_1[\bullet]$, P_1, and Q_1 such that*
 - $C[\bullet] = [\![C_1[\bullet]]\!]_\rho$
 - $D[\bullet] = [\![D_1[\bullet]]\!]_\rho$
 - $P' = [\![P_1]\!]_{\rho'}$ *and* $Q' = [\![Q_1]\!]_{\rho''}$, *where ρ' and ρ'' are paths to holes in $C[\bullet]$ and $D[\bullet]$, respectively.*

(ii) *there are $C_1[\bullet]$, Q, t such that $Q' \equiv t.\big(\mathtt{extr}\langle\!\langle t, p_{t,\rho}, p_\rho\rangle\!\rangle \mid p_\rho[[\![Q]\!]_\varepsilon]\big)$, $D[\bullet] = [\bullet]$, and $C[\bullet] = t[C_1[\bullet]]$.*

The proof of soundness proceeds by induction on n. The base case uses the following lemma. In cases (b) and (c), we use a process of the form $I^{(1)}_t([\![P]\!]_{t,\rho''}, [\![Q]\!]_\varepsilon)$, where t is a name and "1" intuitively denotes the first intermediate process in the translation. In fact, processes of the form $I^{(p)}_t([\![P]\!]_{t,\rho''}, [\![Q]\!]_\varepsilon)$, with $p \geq 1$, to be introduced in Fig. B.17, will be important in the proof of soundness.

Lemma B.4. *Suppose $[\![P]\!]_\rho \longrightarrow R$. Then one of the following holds for P and R:*

a) $P \equiv E[C[\overline{a}.P_1] \mid D[a.P_2]]$ *and* $R \equiv [\![E]\!]_\rho\Big[[\![C]\!]_{\rho_1}[[\![P_1]\!]_{\rho'}] \mid [\![D]\!]_{\rho_1}[[\![P_2]\!]_{\rho''}]\Big]$ *or*

b) $P \equiv E\Big[C[\overline{t}.P_1] \mid D[t[P_2, Q]]\Big]$ *and* $R \equiv [\![E]\!]_\rho\Big[[\![C]\!]_{\rho_1}[h_t.[\![P_1]\!]_{\rho'}] \mid [\![D]\!]_{\rho_1}[I^{(1)}_t([\![P_2]\!]_{t,\rho''}, [\![Q]\!]_\varepsilon)]\Big]$
where $I^{(1)}_t([\![P_2]\!]_{t,\rho''}, [\![Q]\!]_\varepsilon) = t\Big[[\![P_2]\!]_{t,\rho''}\Big] \mid \mathtt{extr}\langle\!\langle t, p_{t,\rho''}, p_{\rho''}\rangle\!\rangle \mid p_{\rho''}[[\![Q]\!]_\varepsilon]$ *or*

c) $P \equiv E[u[C[\overline{u}.P_1], Q]]$ *and* $R \equiv [\![E]\!]_\rho\Big[O^{(1)}_u([\![C]\!]_{u,\rho_1}[h_u.[\![P_1]\!]_{\rho'}], [\![Q]\!]_\varepsilon)\Big]$
where $O^{(1)}_u([\![C]\!]_{u,\rho_1}[h_u.[\![P_1]\!]_{\rho'}], [\![Q]\!]_\varepsilon) = u\Big[[\![C]\!]_{u,\rho_1}[h_u.[\![P_1]\!]_{\rho'}]\Big] \mid \mathtt{extr}\langle\!\langle t, p_{u,\rho_1}, p_{\rho_1}\rangle\!\rangle \mid p_\rho[[\![Q]\!]_\varepsilon]$,

for some contexts C, D, E and processes P_1, P_2, Q. Also, paths ρ, ρ', and ρ'' are paths to holes in contexts $E[\bullet]$, $C[\bullet]$, and $D[\bullet]$, respectively.

Proof. The proof is by induction on the reduction $[\![P]\!]_\rho \longrightarrow R$. There are three base cases, which can be obtained by applying Rule (R-In-Out)) with $x = a$ or $x = t$.

a) $[\![P]\!]_\rho = E'[C'[\overline{a}.P'_1] \mid D'[a.P'_2]] \longrightarrow E'[C'[P'_1] \mid D'[P'_2]] = R$:

$$\begin{array}{lll}
\llbracket P\rrbracket_\rho & = \llbracket E\rrbracket_\rho[\llbracket S\rrbracket_{\rho_1}], & \llbracket S\rrbracket_{\rho_1} = C'[\overline{a}.P_1'] \mid D'[a.P_2'] \quad \text{(by Lemma B.2)}\\
& = \llbracket E\rrbracket_\rho[\llbracket C\rrbracket_{\rho_1}[\llbracket S_1\rrbracket_{\rho'}] \mid \llbracket D\rrbracket_{\rho_1}[\llbracket S_2\rrbracket_{\rho''}]], & \llbracket S_1\rrbracket_{\rho'} = \overline{a}.P_1', \llbracket S_2\rrbracket_{\rho''} = a.P_2' \quad \text{(by Corollary B.3)}\\
& = \llbracket E\rrbracket_\rho[\llbracket C\rrbracket_{\rho_1}[\overline{a}.\llbracket P_1\rrbracket_{\rho'}] \mid \llbracket D\rrbracket_{\rho_1}[a.\llbracket P_2\rrbracket_{\rho''}]], & \llbracket P_1\rrbracket_{\rho'} = P_1', \llbracket P_2\rrbracket_{\rho''} = P_2' \quad \text{(by Definition 5.5)}\\
& = \llbracket E\rrbracket_\rho[\llbracket C\rrbracket_{\rho_1}[\llbracket\overline{a}.P_1\rrbracket_{\rho'}] \mid \llbracket D\rrbracket_{\rho_1}[\llbracket a.P_2\rrbracket_{\rho''}]] & \text{(by Definition 5.5)}\\
& = \llbracket E[C[\overline{a}.P_1] \mid D[a.P_2]]\rrbracket_\rho & \text{(by Lemma B.1)}
\end{array}$$

where $\llbracket D\rrbracket_{\rho_1}[\bullet] = D'[\bullet]$, $\llbracket C[\bullet]\rrbracket_{\rho_1} = C'[\bullet]$, and ρ_1, ρ' and ρ'' are paths to holes in $E[\bullet]$, $C[\bullet]$ and $D[\bullet]$, respectively.

b) $\llbracket P\rrbracket_\rho = E'[C'[\overline{t}.P_1'] \mid D'[t.P_2']] \longrightarrow E'[C'[P_1'] \mid D'[P_2']] = R$ and $D'[\bullet] \neq [\bullet]$:

By Lemma B.2, Corollary B.3 and Definition 5.5, we get the following derivation:

$$\begin{array}{lll}
\llbracket P\rrbracket_\rho & = \llbracket E\rrbracket_\rho[\llbracket S\rrbracket_{\rho_1}], & \llbracket S\rrbracket_{\rho_1} = C'[\overline{t}.P_1'] \mid D''[t[P_2''] \mid t.P_2']\\
& = \llbracket E\rrbracket_\rho[\llbracket C\rrbracket_{\rho_1}[\llbracket S_1\rrbracket_{\rho'}] \mid \llbracket D\rrbracket_{\rho_1}[\llbracket S_2\rrbracket_{\rho''}]], & \llbracket S_1\rrbracket_{\rho'} = \overline{t}.P_1', \llbracket S_2\rrbracket_{\rho''} = t[P_2''] \mid t.P_2'\\
& = \llbracket E\rrbracket_\rho[\llbracket C\rrbracket_{\rho_1}[\overline{t}.h_t.\llbracket P_1\rrbracket_{\rho'}] \mid \llbracket D\rrbracket_{\rho_1}[t[\llbracket P_2\rrbracket_{\rho''}] & h_t.\llbracket P_1\rrbracket_{\rho'} = P_1', \llbracket P_2\rrbracket_{\rho''} = P_2'',\\
& \qquad \mid t.\big(\mathtt{extr}\langle\langle t, p_{t,\rho''}, p_{\rho''}\rangle\rangle \mid p_{\rho''}[\llbracket Q\rrbracket_\varepsilon]\big)], & \mathtt{extr}\langle\langle t, p_{t,\rho''}, p_{\rho''}\rangle\rangle \mid p_{\rho''}[\llbracket Q\rrbracket_\varepsilon] = P_2'\\
& = \llbracket E\rrbracket_\rho[\llbracket C\rrbracket_{\rho_1}[\llbracket\overline{t}.P_1\rrbracket_{\rho'}] \mid \llbracket D\rrbracket_{\rho_1}[\llbracket t[P_2, Q]\rrbracket_{\rho''}]] &\\
& = \llbracket E[C[\overline{t}.P_1] \mid D[t[P_2, Q]]]\rrbracket_\rho &
\end{array}$$

where $\llbracket D\rrbracket_{\rho_1}[\bullet] = D''[\bullet]$, $\llbracket C[\bullet]\rrbracket_{\rho_1} = C'[\bullet]$, and ρ_1, ρ', ρ'' are paths to holes in $E[\bullet]$, $C[\bullet]$, $D[\bullet]$, respectively.

c) $\llbracket P\rrbracket_\rho = E'[C'[\overline{u}.P_1'] \mid D'[u.P_2']] \longrightarrow E'[C'[P_1'] \mid D'[P_2']] = R$ and $D'[\bullet] = [\bullet]$ and $C'[\bullet] = t[C''[\bullet]]$:

By Lemma B.2, Corollary B.3 and Definition 5.5, we get the following derivation:

$$\begin{array}{lll}
\llbracket P\rrbracket_\rho & = E'[u[C''[\overline{u}.P_1']] \mid u.P_2'] &\\
& = \llbracket E\rrbracket_\rho[\llbracket S\rrbracket_{\rho_1}], & \llbracket S\rrbracket_{\rho_1} = u[C''[\overline{u}.P_1']] \mid u.P_2'\\
& = \llbracket E\rrbracket_\rho[u[\llbracket P_1''\rrbracket_{u,\rho_1}] & \llbracket P_1''\rrbracket_{u,\rho_1} = C''[\overline{u}.P_1']\\
& \quad \mid u.\big(\mathtt{extr}\langle\langle u, p_{u,\rho_1}, p_{\rho_1}\rangle\rangle \mid p_{\rho_1}[\llbracket Q\rrbracket_\varepsilon]\big)], & \mathtt{extr}\langle\langle t, p_{u,\rho_1}, p_{\rho_1}\rangle\rangle \mid p_{\rho_1}[\llbracket Q\rrbracket_\varepsilon] = P_2'\\
& = \llbracket E\rrbracket_\rho[u[\llbracket C\rrbracket_{u,\rho_1}[\llbracket\overline{u}.P_1\rrbracket_{\rho'}]] \mid u.\big(\mathtt{extr}\langle\langle u, p_{u,\rho_1}, p_{\rho_1}\rangle\rangle \mid p_{\rho_1}[\llbracket Q\rrbracket_\varepsilon]\big)] &\\
& = \llbracket E[u[C[\overline{t}.P_1], Q]]\rrbracket_\rho. &
\end{array}$$

where $\llbracket C[\bullet]\rrbracket_{t,\rho_1} = C''[\bullet]$ and ρ_1 and ρ' are paths to holes in $E[\bullet]$ and $C[\bullet]$.

Note that since we analyze only one (first) reduction step, i.e. $\llbracket P\rrbracket_\rho \longrightarrow R$, the case of a reduction derived by Rule (R-Sub-Upd) is excluded by definition of translation.

Finally, the inductive step considers cases when the last step was derived by Rule (R-Str). In that way, we get case with "$\equiv$" instead of "$=$" in the three base cases. □

Starting from an adaptable process P that results from our translation, we single out those processes that P reduces to but that do not correspond to the translation of any compensable process. Such processes always appear after a synchronization on some name t and before synchronization on the reserved name h_t. We will first consider computations of a process that results from translating the parallel composition of a transaction and its failure signal (possibly with some continuation). Recall that function $\mathsf{ch}(t, R)$ (cf. Definition 5.3) checks whether R is structurally equivalent to a process of the form $C[h_t.S]$, for some context $C[\bullet]$ and process S: if this is not the case, then $\mathsf{ch}(t, R) = \mathbf{0}$. In a process obtained from our translation, process $h_t.S$ always occurs within a process of the form $\overline{t}.h_t.S$ (cf. Fig. 6), directly implying that any process $\llbracket P\rrbracket_\rho$ cannot be congruent with $C[h_t.S]$. This is stated by the following lemma.

Lemma B.5. *Let P be a well-formed compensable process. If $\llbracket P\rrbracket_\rho = \pi.Q$ then $\pi = a$ or $\pi = \overline{a}$ or $\pi = \overline{t}$, for some $a \in \mathcal{N}_s$ and $t \in \mathcal{N}_t$.*

Proof. Follows directly from definition of the translation (cf. Definition 5.5). □

Lemma B.6. *Let P be a well-formed compensable process, t a transaction name, and ρ a path. Then, it holds that $\mathsf{ch}(t, \llbracket P\rrbracket_\rho) = \mathbf{0}$.*

Proof. By contradiction. Suppose, for the sake of contradiction, that $\mathsf{ch}(t, \llbracket P\rrbracket_\rho) = h_t.\mathbf{0}$. Then, $\llbracket P\rrbracket_\rho \equiv C[h_t.S]$. By Lemma B.2, there are $C_1[\bullet]$ and Q such that $\llbracket C_1[\bullet]\rrbracket_\rho = C[\bullet]$ and $\llbracket Q\rrbracket_{\rho'} = h_t.S$, where ρ' is the path to $[\bullet]$ in $C_1[\bullet]$. But this contradicts Lemma B.5: it is not possible that $\llbracket Q\rrbracket_{\rho'} = h_t.S$ since, necessarily, h_t is a reserved name in $\mathcal{N}_s^r$; by Definition 5.2, $\mathcal{N}_s^r \cap \mathcal{N}_t = \emptyset$ and $\mathcal{N}_s^r \cap \mathcal{N}_s = \emptyset$. □

In studying the processes that are obtained by translating the parallel composition of a transaction and its (externally triggered) failure signal (and its computation), we come to the lemmas that identify processes that are created before a synchronization on h_t.

Lemma B.7. *If $\llbracket E\rrbracket_\rho[P] \mid Q \equiv C[S]$ where $S = \pi.R$ or $S = \checkmark$ then there exist contexts E', E'', and E''' such that:*

1. *$\llbracket E[\bullet]\rrbracket_\rho = \llbracket E'[\bullet]\rrbracket_\rho \mid \llbracket E''\rrbracket_\rho[S]$ and for $S = \pi.R$ it holds $\pi \in \{x, \overline{x}\}$, or*
2. *$P \equiv E'''[S]$, or*
3. *$Q \equiv E'''[S]$.*

Proof. The proof proceeds by induction on the structure of context C. □

$$\begin{tikzcd}
 & I_t^{(1)} \arrow[r] & I_t^{(2)} \arrow[r] & I_t^{(3)} \arrow[rrrrd] & & & & \\
\llbracket t[P,Q]\rrbracket_\rho \mid \llbracket \overline{t}\rrbracket_{\rho'} \arrow[ur, "m=0"] \arrow[dr, "m>0"'] & & & & & & & \llbracket \mathsf{extr}(P) \mid \langle Q\rangle\rrbracket_\rho \\
 & I_t^{(1)} \arrow[r] & I_t^{(2)} \arrow[r] & \cdots \arrow[r] & I_t^{(m+2)} \arrow[r] & I_t^{(m+3)} \arrow[urr] & &
\end{tikzcd}$$

Fig. B.16. Process $I_t^{(p)}$.

(p)	$I_t^{(p)}(\llbracket P\rrbracket_{t,\rho}, \llbracket Q\rrbracket_\varepsilon)$ for $\mathtt{nl}(p_{t,\rho}, \llbracket P\rrbracket_{t,\rho}) = 0$
(1)	$t\big[\llbracket P\rrbracket_{t,\rho}\big] \mid \mathtt{extr}\langle\langle t, p_{t,\rho}, p_\rho\rangle\rangle \mid p_\rho[\llbracket Q\rrbracket_\varepsilon] \equiv t\big[\llbracket P\rrbracket_{t,\rho}\big] \mid t\langle\langle (Y).t[Y] \mid \mathsf{ch}(t,Y) \mid t\langle\langle\dagger\rangle\rangle.\overline{h_t}\rangle\rangle \mid p_\rho[\llbracket Q\rrbracket_\varepsilon]$
(2)	$t\big[\llbracket P\rrbracket_{t,\rho}\big] \mid t\langle\langle\dagger\rangle\rangle.\overline{h_t} \mid p_\rho[\llbracket Q\rrbracket_\varepsilon]$
(3)	$\overline{h_t} \mid p_\rho[\llbracket Q\rrbracket_\varepsilon]$

(p)	$I_t^{(p)}(\llbracket P\rrbracket_{t,\rho}, \llbracket Q\rrbracket_\varepsilon)$ for $\mathtt{nl}(p_{t,\rho}, \llbracket P\rrbracket_{t,\rho}) > 0$
(1)	$t\big[\llbracket P\rrbracket_{t,\rho}\big] \mid \mathtt{extr}\langle\langle t, p_{t,\rho}, p_\rho\rangle\rangle \mid p_\rho[\llbracket Q\rrbracket_\varepsilon]$ $\equiv t\big[\llbracket P\rrbracket_{t,\rho}\big] \mid t\langle\langle (Y).t[Y] \mid \mathsf{ch}(t,Y) \mid \mathtt{out}^{\mathtt{s}}(p_{t,\rho}\,,\, p_\rho\,,\, \mathtt{nl}(p_{t,\rho}, Y)\,,\, t\langle\langle\dagger\rangle\rangle.\overline{h_t})\rangle\rangle \mid p_\rho[\llbracket Q\rrbracket_\varepsilon]$
$(j+2)$ $0 \le j \le m-1$	$t\big[\llbracket P\rrbracket_{t,\rho}\big] \mid p_{t,\rho}\langle\langle (X_1,\dots,X_{m-j}).(\prod_{k=1}^{m-j} p_\rho[X_k] \mid t\langle\langle\dagger\rangle\rangle.\overline{h_t})\rangle\rangle \mid \prod_{k=1}^{j} p_\rho[\llbracket P'_k\rrbracket_\varepsilon] \mid p_\rho[\llbracket Q_t\rrbracket_\varepsilon]$
$(m+2)$	$t\big[\llbracket P'\rrbracket_{t,\rho}\big] \mid \prod_{k=1}^{m} p_\rho[\llbracket P'_k\rrbracket_\varepsilon] \mid t\langle\langle\dagger\rangle\rangle.\overline{h_t} \mid p_\rho[\llbracket Q\rrbracket_\varepsilon]$
$(m+3)$	$\prod_{k=1}^{m} p_\rho[\llbracket P'_k\rrbracket_\varepsilon] \mid \overline{h_t} \mid p_\rho[\llbracket Q\rrbracket_\varepsilon]$

Fig. B.17. Process $I_t^{(p)}(\llbracket P\rrbracket_{t,\rho}, \llbracket Q\rrbracket_\varepsilon)$ with $p \ge 1$.

The following definition formalizes all possible forms for the process $I_t^{(p)}(\llbracket P\rrbracket_{t,\rho}, \llbracket Q\rrbracket_\varepsilon)$. Recall that function $\mathtt{nl}(l, P)$, defined in Definition 5.3 (1), returns the number of locations l in process P.

Definition B.2. Let P, Q be well-formed compensable processes. Given a name t, a path ρ, and $p \ge 1$, we define the intermediate processes $I_t^{(p)}(\llbracket P\rrbracket_{t,\rho}, \llbracket Q\rrbracket_\varepsilon)$ (Fig. B.17) depending on $m = \mathtt{nl}(p_{t,\rho}, \llbracket P\rrbracket_{t,\rho})$:

1. if $m = 0$ then $p \in \{1, 2, 3\}$;
2. otherwise, if $m > 0$ then $\llbracket P\rrbracket_{t,\rho} = \prod_{k=1}^{m} p_{t,\rho}[\llbracket P'_k\rrbracket_\varepsilon] \mid S$ and $p \in \{1, \dots, m+3\}$.

Fig. B.16 illustrates how intermediate processes relate to the encoding of well-formed compensable processes.

Lemma B.8. *Let P_1 be a well-formed compensable process such that*

- $\llbracket P_1\rrbracket_\varepsilon \equiv \llbracket E\rrbracket_\varepsilon \Big[\llbracket G\rrbracket_\rho \Big[\llbracket C\rrbracket_{\rho'} \Big[\llbracket t[P_t, Q_t]\rrbracket_{\rho''}\Big] \mid \llbracket D\rrbracket_{\rho'} \Big[\llbracket \overline{t}.S_t\rrbracket_{\rho'''}\Big] \mid M_1\Big] \mid M_2\Big] \mid M_3$ *and*
- $\llbracket P_1\rrbracket_\varepsilon \longrightarrow^{n-1} R \equiv \llbracket E_1\rrbracket_\varepsilon \Big[\llbracket G_1\rrbracket_\rho \Big[\llbracket C_1\rrbracket_{\rho'} \Big[I_t^{(p)}(\llbracket P'_t\rrbracket_{t,\rho''}, \llbracket Q'_t\rrbracket_\varepsilon)\Big] \mid \llbracket D_1\rrbracket_{\rho'} \Big[h_t.\llbracket S_t\rrbracket_{\rho'''}\Big] \mid M'_1\Big] \mid M'_2\Big] \mid M'_3,$

where $I_t^{(p)}(\llbracket P_t\rrbracket_{t,\rho''}, \llbracket Q_t\rrbracket_\varepsilon)$ in R is as in Definition B.2.
If $R \longrightarrow R'$ then either

I) $R' \equiv \llbracket E_1\rrbracket_\varepsilon \Big[\llbracket G_1\rrbracket_\rho \Big[\llbracket C_1\rrbracket_{\rho'} \Big[I_t^{(p+1)}(\llbracket P'_t\rrbracket_{t,\rho''}, \llbracket Q'_t\rrbracket_\varepsilon)\Big] \mid \llbracket D_1\rrbracket_{\rho'} \Big[h_t.\llbracket S_t\rrbracket_{\rho'''}\Big] \mid M'_1\Big] \mid M'_2\Big] \mid M'_3$ *or*
II) $R' \equiv \llbracket E_2\rrbracket_\varepsilon \Big[\llbracket G_2\rrbracket_\rho \Big[\llbracket C_2\rrbracket_{\rho'} \Big[I_t^{(p)}(\llbracket P''_t\rrbracket_{t,\rho''}, \llbracket Q''_t\rrbracket_\varepsilon)\Big] \mid \llbracket D_2\rrbracket_{\rho'} \Big[h_t.\llbracket S_t\rrbracket_{\rho'''}\Big] \mid M''_1\Big] \mid M''_2\Big] \mid M''_3,$

where

- *$n > 1$;*
- *ρ is the path to holes in $\llbracket E[\bullet]\rrbracket_\varepsilon$ and $\llbracket E_k[\bullet]\rrbracket_\varepsilon$ and $k \in \{1, 2\}$;*

- *ρ' is the path to holes in $[\![G[\bullet]]\!]_\rho$, and $[\![G_k[\bullet]]\!]_\rho$ and $k \in \{1,2\}$;*
- *ρ'' is the path to the hole in $[\![C[\bullet]]\!]_{\rho'}$ and $[\![C_k[\bullet]]\!]_{\rho'}$ and $k \in \{1,2\}$;*
- *ρ''' is the path to hole in $[\![D[\bullet]]\!]_{\rho'}$ and $[\![D_k[\bullet]]\!]_{\rho'}$ and $k \in \{1,2\}$.*

Proof. It is important to notice that the path to $I_t^p([\![P_t]\!]_{t,\rho''}, [\![Q_t]\!]_\varepsilon)$ in R is the same as the path to $[\![t[P_t', Q_t']]\!]_{\rho''}$ in $[\![P_1]\!]_\varepsilon$. This means that we will identify all possible transformations of $[\![t[P_t, Q_t]]\!]_{\rho''}$ that can appear during computations of $[\![P_1]\!]_\varepsilon$, before the synchronization on h_t. By Definition 5.5, we get:

$$\begin{aligned}[\![P_1]\!]_\varepsilon \equiv & [\![E]\!]_\varepsilon\Big[[\![G]\!]_\rho\Big[[\![C]\!]_{\rho'}\Big[t\big[[\![P_t]\!]_{t,\rho''}\big] \mid t.\big(\mathtt{extr}\langle\!\langle t, p_{t,\rho''}, p_{\rho''}\rangle\!\rangle \mid p_{\rho''}[[\![Q_t]\!]_\varepsilon]\big)\Big] \\ & \mid [\![D]\!]_{\rho'}\big[\overline{t}.h_t.[\![S_t]\!]_{\rho'''}\big] \mid M_1\Big] \mid M_2\Big] \mid M_3\end{aligned} \tag{B.10}$$

We continue with the proof by case analysis for the step, $R \longrightarrow R'$, that can be realized. The analysis depends on the shape $I_t^{(p)}([\![P_t']\!]_{t,\rho''}, [\![Q_t']\!]_\varepsilon)$. Hence, there are multiple cases, for $p \in \{1,\dots,m+3\}$ and $m \geq 0$. We detail only one case, namely $p = 1$; all other cases proceed similarly.

If $p = 1$ then

$$R \equiv [\![E_1]\!]_\varepsilon\left[[\![G_1]\!]_\rho\left[[\![C_1]\!]_{\rho'}\left[I_t^{(1)}([\![P_t']\!]_{t,\rho''}, [\![Q_t']\!]_\varepsilon)\right] \mid [\![D_1]\!]_{\rho'}\left[h_t.[\![S_t]\!]_{\rho'''}\right] \mid M_1'\right] \mid M_2'\right] \mid M_3'. \tag{B.11}$$

In the analysis, we will use the following representation of process R:

$$\begin{aligned}& R \equiv [\![E_1]\!]_\rho[P'] \mid M_3' \text{ where} \\ & P' \equiv [\![G_1]\!]_\rho\left[[\![C_1]\!]_{\rho'}\left[I_t^{(1)}([\![P_t']\!]_{t,\rho''}, [\![Q_t']\!]_\varepsilon)\right] \mid [\![D_1]\!]_{\rho'}\left[h_t.[\![S_t]\!]_{\rho'''}\right] \mid M_1'\right] \mid M_2'\end{aligned} \tag{B.12}$$

For $R \longrightarrow R'$ we analyze the following two sub-cases, based on the rules from reduction semantics for adaptable processes: Rule (R-IN-OUT) and Rule (R-SUB-UPD) (cf. Fig. 4).

A) By using Rule (R-IN-OUT): $R \equiv E\Big[C\big[\overline{x}.P\big] \mid D\big[x.Q\big]\Big] \longrightarrow E\Big[C\big[P\big] \mid D\big[Q\big]\Big] \equiv R'$. Therefore, we have:

$$R \equiv E'[x.Q] \text{ where } E'[\bullet] = E[C[\overline{x}.P] \mid D[\bullet]] \text{ and}$$
$$R \equiv E''[\overline{x}.P] \text{ where } E''[\bullet] = E[C[\bullet] \mid D[x.Q]].$$

In (B.12), based on Lemma B.7 for $R \equiv E'[x.Q]$, the following holds:
(i) $[\![E_1[\bullet]]\!]_\rho = [\![E_1'[\bullet]]\!]_\rho \mid [\![E_1'']\!]_\rho[x.Q]$, or
(ii) $P' = E'''[x.Q]$, or
(iii) $M_3' = E'''[x.Q]$.
In the following, we present detail analysis only for case (i). Proofs of cases (ii) and (iii) follow in a similar way, i.e., with the case analysis that is result of applying Lemma B.7.
Therefore, if (i) holds then $R \equiv [\![E_1]\!]_\rho[P'] \mid M_3' = [\![E_1']\!]_\rho[P'] \mid [\![E_1'']\!]_\rho[x.Q] \mid M_3'$. For $R \equiv E''[\overline{x}.P]$ the following holds based on Lemma B.7:
(a) $[\![E_1'[\bullet]]\!]_\rho = [\![E_2'[\bullet]]\!]_\rho \mid [\![E_2'']\!]_\rho[\overline{x}.P]$, or
(b) $P' = E_2'''[\overline{x}.P]$ or
(c) $M_3' \mid [\![E_1'']\!]_\rho[x.P] = E_2'''[\overline{x}.P]$.
In the following we analyze the sub-cases. It should be noted that in all sub-cases, the obtained process R' corresponds to the case II) from the statement:
(a) $R \equiv [\![E_2'[P']]\!]_\rho \mid [\![E_2'']\!]_\rho[\overline{x}.P] \mid [\![E_1'']\!]_\rho[x.Q] \mid M_3' \longrightarrow [\![E_2'[P']]\!]_\rho \mid [\![E_2'']\!]_\rho[P] \mid [\![E_1'']\!]_\rho[Q] \mid M_3' \equiv R'$, or
(b) $R \equiv [\![E_1]\!]_\rho[E_2''[\overline{x}.P]] \mid E_1''[x.Q] \mid M_3' \longrightarrow [\![E_1]\!]_\rho[E_2''[P]] \mid E_1''[Q] \mid M_3' \equiv R'$, or
(c) we distinguish two cases based on Lemma B.7:
   - $M_3' \equiv E_1^{iv}[\overline{x}.P]$ and it follows

   $$R \equiv [\![E_1']\!]_\rho[P'] \mid [\![E_1'']\!]_\rho[x.Q] \mid E_1^{iv}[\overline{x}.P] \longrightarrow [\![E_1']\!]_\rho[P'] \mid [\![E_1'']\!]_\rho[Q] \mid E_1^{iv}[P] \equiv R', \text{ or}$$

   - $[\![E_1'']\!]_\rho[\bullet] = [\![E_1^{iv}[\bullet]]\!]_\rho \mid [\![E_1^{v}]\!]_\rho[\overline{x}.P]$ and it follows:

   $$R \equiv [\![E_1^{iv}]\!]_\rho[P'] \mid E_1^{v}[\overline{x}.P] \mid \mid E_1''[x.Q] \mid M_3' \longrightarrow [\![E_1^{iv}]\!]_\rho[P'] \mid E_1^{v}[P] \mid \mid E_1''[Q] \mid M_3' \equiv R'.$$

B) By using Rule (R-SUB-UPD):

$$R \equiv E\Big[C\big[l[P]\big] \mid D\big[l\langle\!\langle(X).Q\rangle\!\rangle.S\big]\Big] \longrightarrow E\Big[C\big[\mathbf{0}\big] \mid D\big[Q\{P/X\} \mid S\big]\Big] \equiv R'.$$

Therefore, we have that $R \equiv E'[l\langle\!\langle(X).Q\rangle\!\rangle.S]$ such that $E'[\bullet] = E[C[l[P]] \mid D[\bullet]]$. In (B.12), based on Lemma B.7, the following holds:

(i) $M_3' \equiv E'''[l\langle\langle (X).Q\rangle\rangle.S]$, or
(ii) $P' \equiv E'''[l\langle\langle (X).Q\rangle\rangle.S]$.

By Definition 5.5, for every process P_1 a location name in $[\![P_1]\!]_\varepsilon$ is either a transaction name or a reserved name $p_{s,\rho}$ for some s, ρ. Therefore, if interaction on them exists then they should be part of some process $I_s^{(p)}([\![P_s']\!]_{s,\rho}, [\![Q_s']\!]_\varepsilon)$, i.e.,

$$I_s^{(1)}([\![P_s']\!]_{s,\rho}, [\![Q_s']\!]_\varepsilon) \equiv s\big[[\![P_s']\!]_{s,\rho''}\big] \mid s\langle\langle (Y).s[Y] \mid \mathsf{ch}(s,Y) \mid \mathtt{out}^{\mathtt{s}}(p_{s,\rho''}\,,\, p_{\rho''}\,,\, \mathtt{nl}(p_{s,\rho''},Y)\,,\, s\langle\langle \dagger\rangle\rangle.\overline{h_s})\rangle\rangle \mid p_{\rho'}[[\![Q_s']\!]_\varepsilon]$$

(cf. Fig. B.17 for the other forms). This directly provides that process in the form $s[P]$ (i.e., $p_{s,\rho}[P]$) and update $s\langle\langle (X).Q\rangle\rangle.S$ (i.e., $p_{s,\rho}\langle\langle (X).Q\rangle\rangle.S$) have to be in parallel composition.

In the following, we analyze cases (i) and (ii). It should be noted that in all obtained cases and sub-cases, except sub-case (2.2.2) below, we have that process R' corresponds to the case I) from the statement. Process R' obtained in (2.2.2) corresponds to the case II).

(1) If (i) holds, then $R \equiv [\![E_1]\!]_\rho[P'] \mid E'''[l\langle\langle (X).Q\rangle\rangle.S]$, cf. (B.12). In the following we analyze where location $l[P]$ can occur. We have the following cases:
- $E'''[\bullet] = E_1'''[\bullet] \mid l[P]$ and for this case it follows that:

$$R' \equiv [\![E_1]\!]_\varepsilon\left[P'\right] \mid M_3'' \text{ where } M_3'' = E_1'''[Q\{^P/_X\} \mid S] \mid \mathbf{0}, \text{ or}$$

- $[\![E_1]\!]_\rho[\bullet] \equiv E_2[\bullet] \mid l[P] \mid M_3'$

$$R' \equiv [\![E_2]\!]_\varepsilon\left[P'\right] \mid M_3'' \text{ where } M_3'' = \mathbf{0} \mid E'''[Q\{^P/_X\} \mid S].$$

(2) If (ii) holds then

$$E'''[l\langle\langle (X).Q\rangle\rangle.R] \equiv [\![G_1]\!]_\rho\left[[\![C_1]\!]_{\rho'}\left[I_t^{(1)}([\![P_t']\!]_{t,\rho''}, [\![Q_t']\!]_\varepsilon)\right] \mid [\![D_1]\!]_{\rho'}\left[h_t.[\![S_t]\!]_{\rho'''}\right] \mid M_1'\right] \mid M_2'.$$

In the following analysis we consider two sub-cases:
(2.1) By exploiting (i) it holds that $[\![G_1]\!]_\rho[P''] \mid M_2'$ where

$$P'' \equiv [\![C_1]\!]_{\rho'}\left[I_t^{(1)}([\![P_t']\!]_{t,\rho''}, [\![Q_t']\!]_\varepsilon)\right] \mid [\![D_1]\!]_{\rho'}\left[h_t.[\![S_t]\!]_{\rho'''}\right] \mid M_1' \text{ and}$$
$$M_2' \equiv E_1'''[l\langle\langle (X).Q'\rangle\rangle.R'.$$

We have that $E'''[\bullet] = E_1'''[\bullet] \mid l[P]$ and for this case the following holds:

$$R' \equiv [\![E_1]\!]_\varepsilon\left[[\![G_1]\!]_\rho\left[P''\right] \mid M_2''\right] \mid M_3'.$$

(2.2) By exploiting case (ii) it holds that $[\![G_1]\!]_\rho[P''] \mid M_2'$

$$P'' \equiv E'''[l\langle\langle (X).Q\rangle\rangle.R] \equiv [\![C_1]\!]_{\rho'}\left[I_t^{(1)}([\![P_t']\!]_{t,\rho''}, [\![Q_t']\!]_\varepsilon)\right] \mid [\![D_1]\!]_{\rho'}\left[h_t.[\![S_t]\!]_{\rho'''}\right] \mid M_1'.$$

We consider the following two sub-cases:
(2.2.1) By exploiting case (i) it holds $[\![C_1]\!]_\rho[P'''] \mid M_1'$ for

$$P''' \equiv [\![C_1]\!]_{\rho'}\left[I_t^{(1)}([\![P_t']\!]_{t,\rho''}, [\![Q_t']\!]_\varepsilon)\right] \mid [\![D_1]\!]_{\rho'}\left[h_t.[\![S_t]\!]_{\rho'''}\right]$$
$$M_1' \equiv E_1'''[l\langle\langle (X).Q'\rangle\rangle.S].$$

We have that $E'''[\bullet] = E_1'''[\bullet] \mid l[P]$ then the following holds:

$$R' \equiv [\![E_1]\!]_\varepsilon\left[[\![G_1]\!]_\rho\left[P''' \mid M_1''\right] \mid M_2'\right] \mid M_3'.$$

(2.2.2) By exploiting case (ii) it holds $[\![C_1]\!]_\rho[P'''] \mid Q'''$ for

$$P''' \equiv E'''[l\langle\langle (X).Q\rangle\rangle.R] \equiv I_t^{(1)}([\![P_t']\!]_{t,\rho''}, [\![Q_t']\!]_\varepsilon) \text{ and}$$
$$Q''' \equiv [\![D_1]\!]_{\rho'}\left[h_t.[\![S_t]\!]_{\rho'''}\right] \mid M_1',$$

and follows directly that

$$I_t^{(1)}([\![P_t']\!]_{t,\rho''}, [\![Q_t']\!]_\varepsilon) \longrightarrow I_t^{(2)}([\![P_t']\!]_{t,\rho''}, [\![Q_t']\!]_\varepsilon) \text{ for } \mathtt{nl}(p_{t,\rho''}, [\![P_t]\!]_{t,\rho''}) > 0, \text{ or}$$
$$I_t^{(1)}([\![P_t']\!]_{t,\rho''}, [\![Q_t']\!]_\varepsilon) \longrightarrow I_t^{(2)}([\![P_t']\!]_{t,\rho''}, [\![Q_t']\!]_\varepsilon) \text{ for } \mathtt{nl}(p_{t,\rho''}, [\![P_t]\!]_{t,\rho''}) = 0.$$

Therefore, process R' is as presented in the following, where $I_t^{(2)}([\![P_t']\!]_{t,\rho''}, [\![Q_t']\!]_\varepsilon$ has an appropriate form, that is described above:

(q)	$O_u^{(q)}(\llbracket F\rrbracket_\rho[h_u.\llbracket P\rrbracket_{\rho'}], \llbracket Q\rrbracket_\varepsilon)$, $\texttt{nl}(p_{u,\rho}, \llbracket F\rrbracket_\rho[h_u.\llbracket P\rrbracket_{\rho'}]) = 0$

(1)	$u\big[\llbracket F\rrbracket_\rho[h_u.\llbracket P\rrbracket_{\rho'}]\big] \mid \texttt{extr}\langle\langle u, p_{u,\rho}, p_\rho\rangle\rangle \mid p_\rho[\llbracket Q\rrbracket_\varepsilon]$ $\equiv u\big[\llbracket F\rrbracket_\rho[h_u.\llbracket P\rrbracket_{\rho'}]\big] \mid u\langle\langle (Y).u[Y] \mid \mathsf{ch}(u,Y) \mid u\langle\langle\dagger\rangle\rangle.\overline{h_u}\rangle\rangle \mid p_\rho[\llbracket Q\rrbracket_\varepsilon]$
(2)	$u\big[\llbracket F\rrbracket_\rho[h_u.\llbracket P\rrbracket_{\rho'}]\big] \mid h_u \mid u\langle\langle\dagger\rangle\rangle.\overline{h_u} \mid p_\rho[\llbracket Q\rrbracket_\varepsilon]$
(3)	$h_u \mid \overline{h_u} \mid p_\rho[\llbracket Q\rrbracket_\varepsilon]$
(4)	$p_\rho[\llbracket Q\rrbracket_\varepsilon]$

(q)	$O_u^{(q)}(\llbracket F\rrbracket_\rho[h_u.\llbracket P\rrbracket_{\rho'}], \llbracket Q\rrbracket_\varepsilon)$, $\texttt{nl}(p_{u,\rho}, \llbracket F\rrbracket_\rho[h_u.\llbracket P\rrbracket_{\rho'}]) > 0$

(1)	$u\big[\llbracket F\rrbracket_\rho[h_u.\llbracket P\rrbracket_{\rho'}]\big] \mid \texttt{extr}\langle\langle u, p_{u,\rho}, p_\rho\rangle\rangle \mid p_\rho[\llbracket Q\rrbracket_\varepsilon]$ $\equiv u\big[\llbracket F\rrbracket_\rho[h_u.\llbracket P\rrbracket_{\rho'}]\big] \mid u\langle\langle (Y).u[Y] \mid \mathsf{ch}(u,Y) \mid \texttt{out}^{\mathsf{s}}(p_{u,\rho}\,,\ p_\rho\,,\ \texttt{nl}(p_{u,\rho}, Y)\,,\ u\langle\langle\dagger\rangle\rangle.\overline{h_u})\rangle\rangle \mid p_\rho[\llbracket Q\rrbracket_\varepsilon]$
$(j+2)$ $0 \le j \le m-1$	$u\big[\llbracket F\rrbracket_\rho[h_u.\llbracket P\rrbracket_{\rho'}]\big] \mid h_u \mid p_{u,\rho}\langle\langle (X_1,\ldots,X_{m-j}).\big(\prod_{k=1}^{m-j} p_\rho[X_k] \mid u\langle\langle\dagger\rangle\rangle.\overline{h_u}\big)\rangle\rangle \mid \prod_{k=1}^{j} p_\rho[\llbracket P'_k\rrbracket_\varepsilon] \mid p_\rho[\llbracket Q\rrbracket_\varepsilon]$
$(m+2)$	$u\big[\llbracket F'\rrbracket_\rho[h_u.\llbracket P\rrbracket_{\rho'}]\big] \mid \prod_{k=1}^{m} p_\rho[\llbracket P'_k\rrbracket_\varepsilon] \mid h_u \mid p_\rho[\llbracket Q\rrbracket_\varepsilon] \mid u\langle\langle\dagger\rangle\rangle.\overline{h_u}$
$(m+3)$	$\prod_{k=1}^{m} p_\rho[\llbracket P'_k\rrbracket_\varepsilon] \mid h_u \mid p_\rho[\llbracket Q\rrbracket_\varepsilon] \mid \overline{h_u}$
$(m+4)$	$\prod_{k=1}^{m} p_\rho[\llbracket P'_k\rrbracket_\varepsilon] \mid p_\rho[\llbracket Q\rrbracket_\varepsilon]$

Fig. B.18. Process $O_u^{(q)}(\llbracket F\rrbracket_\rho[h_u.\llbracket P\rrbracket_{\rho'}], \llbracket Q\rrbracket_\varepsilon)$ with $q \ge 1$.

$$R' \equiv \llbracket E_1\rrbracket_\varepsilon\left[\llbracket G_1\rrbracket_\rho\left[\llbracket C_1\rrbracket_{\rho'}\left[I_t^{(2)}(\llbracket P'_t\rrbracket_{t,\rho''}, \llbracket Q'_t\rrbracket_\varepsilon\right] \mid \llbracket D_1\rrbracket_{\rho'}\left[h_t.\llbracket S_t\rrbracket_{\rho'''}\right] \mid M'_1\right] \mid M'_2\right] \mid M'_3. \quad \square \tag{B.13}$$

The following lemma formalizes all possible forms for the process $O_u^{(q)}(\llbracket F\rrbracket_{\rho''}[h_u.\llbracket P_u\rrbracket_{\rho'''}], \llbracket Q'_u\rrbracket_\varepsilon)$ for $m \ge 0$ and $q \in \{1,\ldots,m+4\}$.

Definition B.3. Let P, Q be well-formed compensable processes. Given a name u, paths ρ, ρ', and $q \ge 1$, we define the intermediate processes $O_u^{(q)}(\llbracket F\rrbracket_\rho[h_u.\llbracket P\rrbracket_{\rho'}], \llbracket Q\rrbracket_\varepsilon)$ (Fig. B.18) depending on $m = \texttt{nl}(p_{u,\rho}, \llbracket F\rrbracket_\rho[h_u.\llbracket P\rrbracket_{\rho'}])$:

1. for $m = 0$ we have $q \in \{1,2,3,4\}$, and
2. for $m > 0$ and $\llbracket F\rrbracket_\rho[h_u.\llbracket P\rrbracket_{\rho'}] = \prod_{k=1}^{m} p_{u,\rho}[\llbracket P'_k\rrbracket_\varepsilon] \mid S$ we have $q \in \{1,\ldots,m+4\}$.

We now continue with the analysis of adaptable processes that can be obtained starting from the translation of a transaction that contains failure signal in its body, which is triggered internally.

Lemma B.9. *Let P_1 be a well-formed compensable process such that*

- $\llbracket P_1\rrbracket_\varepsilon \equiv \llbracket E\rrbracket_\varepsilon\left[\llbracket G\rrbracket_\rho\left[\llbracket L\rrbracket_{\rho'}\left[\llbracket u[F[\overline{u}.P_u], Q_u]\rrbracket_{\rho''}\right] \mid M_1\right] \mid M_2\right] \mid M_3$*, and*
- $\llbracket P_1\rrbracket_\varepsilon \longrightarrow^{n-1} R \equiv \llbracket E_1\rrbracket_\varepsilon\left[\llbracket G_1\rrbracket_\rho\left[\llbracket L_1\rrbracket_{\rho'}\left[O_u^{(q)}(\llbracket F_1\rrbracket_{\rho''}[h_u.\llbracket P_u\rrbracket_{t,\rho'''}], \llbracket Q'_u\rrbracket_\varepsilon)\right] \mid M'_1\right] \mid M'_1\right] \mid M'_3,$

where $O_u^{(q)}(\llbracket F\rrbracket_{\rho''}[h_u.\llbracket P_u\rrbracket_{t,\rho'''}], \llbracket Q_u\rrbracket_\varepsilon)$ in R is a process from Definition B.3.
If $R \longrightarrow R'$ then either

I) $R' \equiv \llbracket E_1\rrbracket_\varepsilon\left[\llbracket G_1\rrbracket_\rho\left[\llbracket L_1\rrbracket_{\rho'}\left[O_u^{(q+1)}(\llbracket F_1\rrbracket_{\rho''}[h_u.\llbracket P_u\rrbracket_{t,\rho'''}], \llbracket Q'_u\rrbracket_\varepsilon)\right] \mid M'_1\right] \mid M'_2\right] \mid M'_3$*, or*
II) $R' \equiv \llbracket E_2\rrbracket_\varepsilon\left[\llbracket G_2\rrbracket_\rho\left[\llbracket L_2\rrbracket_{\rho'}\left[O_u^{(q)}(\llbracket F_2\rrbracket_{\rho''}[h_u.\llbracket P_u\rrbracket_{t,\rho'''}], \llbracket Q''_u\rrbracket_\varepsilon)\right] \mid M''_1\right] \mid M''_2\right] \mid M''_3,$

where:

- $n > 1$*;*

$$\begin{tikzcd}
\llbracket P \rrbracket_{\varepsilon} \arrow[d, "n"'] \arrow[dr, "*"] & \\
R & \llbracket P' \rrbracket_{\varepsilon}
\end{tikzcd}$$

Fig. B.19. Diagram of Lemma B.10.

- *ρ is the path to hole in $\llbracket E \rrbracket_{\varepsilon}[\bullet]$;*
- *ρ' is the path to hole in $\llbracket G \rrbracket_{\rho}[\bullet]$ and $\llbracket G_k \rrbracket_{\rho}[\bullet]$ and $k \in \{1, 2\}$;*
- *ρ'' is the path to the hole in $\llbracket L \rrbracket_{\rho'}[\bullet]$; $\llbracket L_k \rrbracket_{\rho'}[\bullet]$ and $k \in \{1, 2\}$;*
- *ρ''' is the path to the hole in $\llbracket F \rrbracket_{\rho''}[\bullet]$ and $\llbracket F_k \rrbracket_{\rho'}[\bullet]$ and $k \in \{1, 2\}$.*

Proof. By Definition 5.5, we get:

$$\llbracket P_1 \rrbracket_{\varepsilon} \equiv \llbracket E \rrbracket_{\varepsilon}\Big[\llbracket G \rrbracket_{\rho}\Big[\llbracket L \rrbracket_{\rho'}\Big[u\big[\llbracket F \rrbracket_{\rho''}[\overline{u}.h_u.\llbracket P_u \rrbracket_{\rho'''}]\big] \mid u.(\mathtt{extr}\langle\!\langle u, p_{u,\rho''}, p_{\rho''}\rangle\!\rangle \mid p_{\rho''}[\llbracket Q_u \rrbracket_{\varepsilon}])\Big] \mid M_1\Big] \mid M_2\Big] \mid M_3$$

and the proof continues by case analysis for the step, $R \longrightarrow R'$, that can be realized. The proof follows the same idea that is presented for the proof of Lemma B.8. □

The following lemma is crucial for the proof of soundness and it is illustrated in Fig. B.19. Also, we will use the following abbreviations, where we use i, c, k, w as indexes of t, u and F:

$$I^{(p)}_{t_{i,k,w}} = I^{(p)}_{t_{i,k,w}}(\llbracket P'_{t_{i,k,w}} \rrbracket_{t,\rho''}, \llbracket Q'_{t_{i,k,w}} \rrbracket_{\varepsilon}),$$
$$O^{(q)}_{u_{c,k,w}} = O^{(q)}_{u_{c,k,w}}(\llbracket F_{c,k,w} \rrbracket_{\rho''}[h_{u_{c,k,w}}.\llbracket P_{u_{c,k,w}} \rrbracket_{\rho'''}], \llbracket Q'_{u_{c,k,w}} \rrbracket_{\varepsilon}).$$

Lemma B.10. *Let $I^{(p)}_{t_{i,k,w}}$ and $O^{(q)}_{u_{c,k,w}}$ be processes from Definition B.2 and Definition B.3, respectively. If $\llbracket P \rrbracket_{\varepsilon} \longrightarrow^{n} R$, with $n \geq 1$, then*

1)

$$R \equiv \prod_{w=1}^{z} \llbracket E_w \rrbracket_{\varepsilon}\Big[\prod_{k=1}^{s_w} \llbracket G_{k,w} \rrbracket_{\rho_w}\Big[\prod_{i=1}^{l_k} \llbracket C_{i,k,w} \rrbracket_{\rho'_{k,w}}\big[I^{(p)}_{t_{i,k,w}}\big] \mid \prod_{j=1}^{r_k} \llbracket D_{j,k,w} \rrbracket_{\rho'_{k,w}}\big[h_{t_{j,k,w}}.\llbracket S_{t_{j,k,w}} \rrbracket_{\rho''_{k,w}}\big]$$
$$\mid \prod_{c=1}^{m_k} \llbracket L_{c,k,w} \rrbracket_{\rho'_{k,w}}\big[O^{(q)}_{u_{c,k,w}}\big]\Big]\Big] \tag{B.14}$$

and $P \longrightarrow^{*} P'$, where P' is of the following form:

2)

$$P' \equiv \prod_{w=1}^{z} E_w\Big[\prod_{k=1}^{s_w} G_{k,w}\Big[\prod_{i=1}^{l_k} C_{i,k,w}\big[t_{i,k,w}[P_{t_{i,k,w}}, Q_{t_{i,k,w}}]\big] \mid \prod_{j=1}^{r_k} D_{j,k,w}\big[\overline{t_{j,k,w}}.S_{t_{j,k,w}}\big]$$
$$\mid \prod_{c=1}^{m_k} L_{c,k,w}\big[u_{c,k,w}[F_{c,k,w}[\overline{u_{c,k,w}}.P_{u_{c,k,w}}], Q_{u_{c,k,w}}]\big]\Big]\Big], \tag{B.15}$$

for some $E_w[\bullet]$, $G_{k,w}[\bullet]$, $C_{i,k,w}[\bullet]$, $D_{j,k,w}[\bullet]$, and $L_{c,k,w}[\bullet]$ where $w \in \{1, \dots, z\}$, $k \in \{1, \dots, s_w\}$, $i \in \{1, \dots, l_k\}$, $j \in \{1, \dots, r_k\}$, and $c \in \{j, \dots, m_k\}$.

Proof. The proof proceeds by induction on n.

Base case: Assume that $n = 1$, i.e. $\llbracket P \rrbracket_{\varepsilon} \longrightarrow R$. By application of Lemma B.4 there are three possible cases:

a) $P \equiv E'[C'[a.P_1] \mid D'[\overline{a}.P_2]]$ and $R \equiv \llbracket E' \rrbracket_{\varepsilon}\Big[\llbracket C' \rrbracket_{\rho}[\llbracket P_1 \rrbracket_{\rho'}] \mid \llbracket D' \rrbracket_{\rho}[\llbracket P_2 \rrbracket_{\rho''}]\Big]$.
In this case we have: $z = 1$ and $s_1 = 0$ and it holds $E_1[\bullet] = [\bullet] \mid E'\big[C'[a.P_1] \mid D'[\overline{a}.P_2]\big]$ and $P = P'$.

b) $P \equiv E'\Big[C'[t[P_2, Q]] \mid D'[\bar{t}.P_1]]$ and

$$R \equiv [\![E']\!]_\varepsilon\Big[[\![C']\!]_\rho[t\big[[\![P_2']\!]_{t,\rho'}\big] \mid \mathtt{extr}\langle\!\langle t, p_{t,\rho'}, p_{\rho'}\rangle\!\rangle \mid p_{\rho'}[[\![Q']\!]_\varepsilon]] \mid [\![D']\!]_\rho[h_t.[\![P_1]\!]_{\rho''}]\Big].$$

In this case we have: $z=1, s_1=1, l_1=1, r_1=1$ and $m_1=0$. Therefore, the following holds: $E_1[\bullet]=[\bullet], G_{1,1}[\bullet]=E'[\bullet]$ $C_{1,1,1}[\bullet]=C'[\bullet], D_{1,1,1}[\bullet]=D'[\bullet], P_{t_{1,1,1}}=P_2, Q_{t_{1,1,1}}=Q, S_{t_{1,1,1}}=P_1$ and $I^{(1)}_{t_{1,1,1}} \equiv t\big[[\![P_2']\!]_{t,\rho'}\big] \mid \mathtt{extr}\langle\!\langle t, p_{t,\rho'}, p_{\rho'}\rangle\!\rangle \mid p_{\rho'}[[\![Q']\!]_\varepsilon]$ and $P=P'$.

c) $P \equiv C'[u[D'[\overline{u}.P_1], Q]]$ and $R \equiv [\![C']\!]_\varepsilon\Big[u\big[[\![D']\!]_{u,\rho}[h_u.[\![P_1]\!]_{\rho'}]\big] \mid \mathtt{extr}\langle\!\langle u, p_{u,\rho'}, p_{\rho'}\rangle\!\rangle \mid p_{\rho'}[[\![Q']\!]_\varepsilon]\Big]$.
In this case we have: $z=1, s_1=1, l_1=0, r_1=0$ and $m_1=1$. Therefore, the following holds: $E_1[\bullet]=[\bullet], G_{1,1}[\bullet]=C'[\bullet], L_{1,1,1}[\bullet]=D'[\bullet], P_{u_{1,1,1}}=P_1, Q_{u_{1,1,1}}=Q$ and

$$I^{(1)}_{u_{1,1,1}} \equiv u\big[[\![D']\!]_{u,\rho}[h_u.[\![P_1]\!]_{\rho'}]\big] \mid \mathtt{extr}\langle\!\langle u, p_{u,\rho'}, p_{\rho'}\rangle\!\rangle \mid p_{\rho'}[[\![Q']\!]_\varepsilon] \text{ and } P=P'.$$

Inductive hypothesis: Assume that the statement holds for $n-1$ reduction steps, i.e., if $[\![P]\!]_\varepsilon \longrightarrow^{n-1} R_1$ then the statement holds.

Inductive step: We consider that $[\![P]\!]_\varepsilon \longrightarrow^{n-1} R_1 \longrightarrow R$. We know, by inductive hypothesis:

1) R_1 has the following form:

$$R_1 \equiv \prod_{w=1}^{z} [\![E_w]\!]_\varepsilon\Big[\prod_{k=1}^{s_w}[\![G_{k,w}]\!]_{\rho_w}\big[\prod_{i=1}^{l_k}[\![C_{i,k,w}]\!]_{\rho'_{k,w}}\big[I^{(p)}_{t_{i,k,w}}\big] \mid \prod_{j=1}^{r_k}[\![D_{j,k,w}]\!]_{\rho'_{k,w}}\big[h_{t_{j,k,w}}.[\![S_{t_{j,k,w}}]\!]_{\rho''_{k,w}}\big]$$
$$\mid \prod_{c=1}^{m_k}[\![L_{c,k,w}]\!]_{\rho'_{k,w}}\big[O^{(q)}_{u_{c,k,w}}\big]\big]\Big],$$

2) $P \longrightarrow^* P''$ such that P'' has the following form:

$$P'' \equiv \prod_{w=1}^{z} E_w\Big[\prod_{k=1}^{s_w} G_{k,w}\big[\prod_{i=1}^{l_k} C_{i,k,w}\big[t_{i,k,w}[P_{t_{i,k,w}}, Q_{t_{i,k,w}}]\big] \mid \prod_{j=1}^{r_k} D_{j,k,w}\big[\overline{t_{j,k,w}}.S_{t_{j,k,w}}\big]$$
$$\mid \prod_{c=1}^{m_k} L_{c,k,w}\big[u_{c,k,w}[F_{c,k,w}[\overline{u_{c,k,w}}.P_{u_{c,k,w}}], Q_{u_{c,k,w}}]\big]\big]\Big].$$

We continue with the proof by case analysis for the last step, $R_1 \longrightarrow R$, that can be realized. In the following we consider six interesting cases.

(1) Let $I^{(1)}_t$ be a process that has the form as presented in Definition B.2 where $t=t_{1,1,1}, G_1[\bullet]=G_{1,1}[\bullet], C_1[\bullet]=C_{1,1,1}[\bullet], D_1[\bullet]=D_{1,1,1}[\bullet]$ and

$$R_1 \equiv [\![E_1]\!]_\varepsilon\big[[\![G_{1,1}]\!]_{\rho_1}\big[[\![C_{1,1,1}]\!]_{\rho_{1,1}}\big[I^{(1)}_{t_{1,1,1}}([\![P'_{t_{1,1,1}}]\!]_{t,\rho''}, [\![Q'_{t_{1,1,1}}]\!]_\varepsilon)\big] \mid [\![D_{1,1,1}]\!]_{\rho_{1,1}}\big[h_{t_{1,1,1}}.[\![S_{t_{1,1,1}}]\!]_{\rho''_{1,1}}\big]$$
$$\mid M_1'\big] \mid M_2'\big] \mid M_3'.$$

According to the Lemma B.8, it follows that $R_1 \longrightarrow R$ such that R has the form

I)

$$R \equiv [\![E_1]\!]_\varepsilon\big[[\![G_{1,1}]\!]_\rho\big[[\![C_{1,1,1}]\!]_{\rho'}\big[I^{(2)}_{t_{1,1,1}}([\![P'_{t_{1,1,1}}]\!]_{t_{1,1,1},\rho''}, [\![Q'_{t_{1,1,1}}]\!]_\varepsilon)\big] \mid [\![D_1]\!]_{\rho'}\big[h_{t_{1,1,1}}.[\![S_{t_{1,1,1}}]\!]_{\rho'''}\big]$$
$$\mid M_1'\big] \mid M_2'\big] \mid M_3', \text{ or}$$

II)

$$R \equiv [\![E_1']\!]_\varepsilon\big[[\![G'_{1,1}]\!]_\rho\big[[\![C'_{1,1,1}]\!]_{\rho'}\big[I^{(1)}_{t_{1,1,1}}([\![P''_{t_{1,1,1}}]\!]_{t_{1,1,1},\rho''}, [\![Q''_{t_{1,1,1}}]\!]_\varepsilon)\big] \mid [\![D'_{1,1,1}]\!]_{\rho'}\big[h_{t_{1,1,1}}.[\![S_{t_{1,1,1}}]\!]_{\rho'''}\big]$$
$$\mid M_1''\big] \mid M_2''\big] \mid M_3''. \tag{B.16}$$

Here we comment case II), while case I) follows the idea that is given in case a) from Base case.
In case when we get the form II) it directly follows that $P''=P'$.
Similarly, for all $I^{(1)}_t([\![P_t']\!]_{t,\rho''}, [\![Q_t']\!]_\varepsilon)$ with $t \in \{t_{2,1,1}, \dots, t_{l_{s_z}s_z z}\}$.
Similarly, for cases $I^{(p)}_{t_{1,1,1}}([\![P'_{t_{1,1,1}}]\!]_{t_{1,1,1},\rho''}, [\![Q'_{t_{1,1,1}}]\!]_\varepsilon)$ where $p \in \{2, 4, \dots, m+3\}$.

(2) Let $O_u^{(1)}$ be a process that has a form as presented in Definition B.3, where $u = u_{1,1,1}, G_1[\bullet] = G_{1,1}[\bullet], L_1[\bullet] = L_{1,1,1}[\bullet], F_1[\bullet] = F_{1,1,1}[\bullet]$ and

$$R_1 \equiv [\![E_1]\!]_\varepsilon \left[[\![G_{1,1}]\!]_{\rho_1} \left[[\![L_{1,1,1}]\!]_{\rho'_{1,1}} \left[O^{(1)}_{u_{1,1,1}}([\![F_{1,1,1}]\!]_{\rho''}[h_{u_{1,1,1}}.[\![P_{u_{1,1,1}}]\!]_{u_{1,1,1},\rho'''}], [\![Q'_{u_{1,1,1}}]\!]_\varepsilon) \right] \mid M'_1 \right] \mid M'_2 \right] \mid M'_3.$$

According to the Lemma B.9, it follows that $R_1 \longrightarrow R$ such that R has the form

I) $R \equiv [\![E_1]\!]_\varepsilon \left[[\![G_{1,1}]\!]_\rho \left[[\![L_{1,1,1}]\!]_{\rho'} \left[O_u^{(2)}([\![F_{1,1,1}]\!]_{\rho''}[h_{u_{1,1,1}}.[\![P_{u_{1,1,1}}]\!]_{t,\rho'''}], [\![Q'_{u_{1,1,1}}]\!]_\varepsilon) \right] \mid M'_1 \right] \mid M'_2 \right] \mid M'_3$, or

II)

$$R \equiv [\![E'_1]\!]_\varepsilon \left[[\![G'_{1,1}]\!]_\rho \left[[\![L'_{1,1,1}]\!]_{\rho'} \left[O^{(1)}_{u_{1,1,1}}([\![F'_{1,1,1}]\!]_{\rho''}[h_u.[\![P_{u_{1,1,1}}]\!]_{u_{1,1,1},\rho'''}], [\![Q''_{u_{1,1,1}}]\!]_\varepsilon) \right] \mid M''_1 \right] \mid M''_2 \right] \mid M''_3. \tag{B.17}$$

Here we comment on the case II), while case I) follows the idea that is given in case a) from the base case.
In case when we get the form II) it directly follows that $P'' = P'$.
Similarly, for all $O_u^{(1)}([\![F]\!]_{\rho''}[h_u.[\![P_u]\!]_{t,\rho'''}], [\![Q'_u]\!]_\varepsilon)$ with $u \in \{u_{2,1,1}, \dots, u_{m_{s_z},s_z,z}\}$.
Similarly, for cases $O_u^{(q)}([\![F]\!]_{\rho''}[h_u.[\![P_u]\!]_{t,\rho'''}], [\![Q'_u]\!]_\varepsilon)$ where $q \in \{2, 4, \dots, m+4\}$.

(3) Let $I_t^{(3)}$ be a process that has the form as presented in Definition B.2, where $t = t_{1,1,1}, G_1[\bullet] = G_{1,1}[\bullet], C_1[\bullet] = C_{1,1,1}[\bullet], D_1[\bullet] = D_{1,1,1}[\bullet]$ and

$$\begin{aligned} R_1 \equiv [\![E_1]\!]_\varepsilon \Big[[\![G_{1,1}]\!]_{\rho_1} [[\![C_{1,1,1}]\!]_{\rho_{1,1}} [\overline{h_{t_{1,1,1}}} \mid p_{\rho'_{1,1}}[[\![Q'_{t_{1,1,1}}]\!]_\epsilon]] \\ \mid [\![D_{1,1,1}]\!]_{\rho_{1,1}} [h_{t_{1,1,1}}.[\![S_{t_{1,1,1}}]\!]_{\rho''_{1,1}}] \mid M'_1] \mid M'_2 \Big] \mid M'_3. \end{aligned}$$

According to the Lemma B.8, it follows that we can derive $R_1 \longrightarrow R$ such that R has the form of (B.16) or

$$R \equiv [\![E_1]\!]_\varepsilon \Big[[\![G_{1,1}]\!]_{\rho_1} [[\![C_{1,1,1}]\!]_{\rho_{1,1}} [p_{\rho'_{1,1}}[[\![Q'_{t_{1,1,1}}]\!]_\epsilon]] \mid [\![D_{1,1,1}]\!]_{\rho_{1,1}} [[\![S_{t_{1,1,1}}]\!]_{\rho''_{1,1}}] \mid M'_1] \mid M'_2 \Big] \mid M'_3. \tag{B.18}$$

In case when we get the form (B.18) it holds that $P'' \longrightarrow P'$ where:

$$\begin{aligned} P' \equiv\, & E_1 \Big[G_{1,1} [C_{1,1,1} [\langle Q'_{t_{1,1,1}} \rangle] \mid D_{1,1,1} [S_{t_{1,1,1}}] \mid M'_1] \mid M'_2 \Big] \mid M'_3 \\ \equiv\, & E_1 \Big[G_{1,1} [\prod_{i=2}^{l_1} C'_{i,1,1} [t_{i,1,1} [P_{t_{i,1,1}}, Q_{t_{i,1,1}}]] \mid \prod_{j=2}^{r_1} D'_{j,1,1} [\overline{t_{j,1,1}}.S_{t_{j,1,1}}] \\ & \mid \prod_{c=1}^{m_1} L_{c,1,1} [u_{c,1,1} [F_{c,1,1} [\overline{u_{c,1,1}}.P_{u_{c,1,1}}], Q_{u_{c,1,1}}]]] \\ & \mid \prod_{w=2}^{z} E_w \Big[\prod_{k=1}^{s_w} G_{k,w} [\prod_{i=1}^{l_k} C_{i,k,w} [t_{i,k,w} [P_{t_{i,k,w}}, Q_{t_{i,k,w}}]] \mid \prod_{j=1}^{r_k} D_{j,k,w} [\overline{t_{j,k,w}}.S_{t_{j,k,w}}] \\ & \mid \prod_{c=1}^{m_k} L_{c,k,w} [u_{c,k,w} [F_{c,k,w} [\overline{u_{c,k,w}}.P_{u_{c,k,w}}], Q_{u_{c,k,w}}]] \Big], \end{aligned}$$

such that:
- $C'_{2,1,1}[\bullet] = C_{2,1,1}[\bullet] \mid N$, where $N = C_{1,1,1}[\langle Q'_{t_{1,1,1}} \rangle]$ and $C'_{i,1,1}[\bullet] = C_{i,1,1}[\bullet]$ for $i \in \{3, \dots, l_1\}$, and
- $D'_{2,1,1}[\bullet] = D_{2,1,1}[\bullet] \mid N_1$, where $N_1 = D_{1,1,1}[S_{t_{1,1,1}}]$ and $D'_{i,1,1}[\bullet] = D_{i,1,1}[\bullet]$ for $i \in \{3, \dots, l_1\}$.

Similarly, for all $I_t^{(3)}([\![P'_t]\!]_{t,\rho''}, [\![Q'_t]\!]_\varepsilon)$ with $t \in \{t_{2,1,1}, \dots, t_{l_{s_z} s_z z}\}$.
Similarly, for $I_t^{(m+3)}([\![P'_t]\!]_{t,\rho''}, [\![Q'_t]\!]_\varepsilon)$ in Definition B.2.

(4) $O_u^{(3)}$ is a process that has a form as presented in Definition B.3, where $u = u_{1,1,1}, G_1[\bullet] = G_{1,1}[\bullet], L_1[\bullet] = L_{1,1,1}[\bullet], F_1[\bullet] = F_{1,1,1}[\bullet]$ and

$$R_1 \equiv [\![E_1]\!]_\varepsilon \left[[\![G_{1,1}]\!]_{\rho_1} \left[[\![L_{1,1,1}]\!]_{\rho'_{1,1}} \left[h_{u_{1,1,1}} \mid \overline{h_{u_{1,1,1}}} \mid p_{\rho''}[[\![Q'_{u_{1,1,1}}]\!]_\varepsilon] \right] \mid M'_1 \right] \mid M'_2 \right] \mid M'_3.$$

According to the proof of Lemma B.9, it follows that and $R_1 \longrightarrow R$ such that R has the form (B.17) or

$$R_1 \equiv [\![E_1]\!]_\varepsilon \left[[\![G_{1,1}]\!]_{\rho_1} \left[[\![L_{1,1,1}]\!]_{\rho'_{1,1}} \left[p_{\rho''}[[\![Q'_{u_{1,1,1}}]\!]_\varepsilon] \right] \mid M'_1 \right] \mid M'_2 \right] \mid M'_3. \tag{B.19}$$

In case when we get the form (B.19) it holds $P'' \longrightarrow P'$ where

$$
\begin{aligned}
P' \equiv\, & E_1\Big[G_{1,1}\big[L_{1,1,1}[\langle Q'_{u_{1,1,1}}\rangle] \mid M'_1\big] \mid M'_2\Big] \mid M'_3\\
\equiv\, & E_1\Big[G_{1,1}\big[\prod_{i=1}^{l_1} C_{i,1,1}\big[t_{i,1,1}[P_{t_{i,1,1}}, Q_{t_{i,1,1}}]\big] \mid \prod_{j=1}^{r_1} D_{j,1,1}\big[\overline{t_{j,1,1}}.S_{t_{j,1,1}}\big]\\
& \mid \prod_{c=2}^{m_1} L'_{c,1,1}\big[u_{c,1,1}[F_{c,1,1}[\overline{u_{c,1,1}}.P_{u_{c,1,1}}], Q_{u_{c,1,1}}]\big]\Big]\\
& \mid \prod_{w=2}^{z} E_w\Big[\prod_{k=1}^{s_w} G_{k,w}\big[\prod_{i=1}^{l_k} C_{i,k,w}\big[t_{i,k,w}[P_{t_{i,k,w}}, Q_{t_{i,k,w}}]\big] \mid \prod_{j=1}^{r_k} D_{j,k,w}\big[\overline{t_{j,k,w}}.S_{t_{j,k,w}}\big]\\
& \mid \prod_{c=1}^{m_k} L_{c,k,w}\big[u_{c,k,w}[F_{c,k,w}[\overline{u_{c,k,w}}.P_{u_{c,k,w}}], Q_{u_{c,k,w}}]\big]\Big],
\end{aligned}
$$

such that $L'_{2,1,1}[\bullet] = L_{2,1,1}[\bullet] \mid N$, where $N = L_{1,1,1}[\langle Q'_{u_{1,1,1}}\rangle]$ and $L'_{i,1,1}[\bullet] = L_{i,1,1}[\bullet]$ for $i \in \{3, \dots, m_1\}$.
Similarly, for all $O^{(3)}_u(\llbracket F\rrbracket_{\rho''}[h_u.\llbracket P_u\rrbracket_{t,\rho'''}], \llbracket Q'_u\rrbracket_\varepsilon)$ with $u \in \{u_{2,1,1}, \dots, u_{m_{s_z}s_z z}\}$.
Similarly, for case of $O^{(m+3)}_u(\llbracket F\rrbracket_{\rho''}[h_u.\llbracket P_u\rrbracket_{t,\rho'''}], \llbracket Q'_u\rrbracket_\varepsilon)$ in Definition B.3.

(5) In this case let us consider the following context:

$$
G_{1,1}[\bullet] = G'_{1,1}[\bullet] \mid C_{(l_1+1)1,1}[t_{(l_1+1)1,1}[P_{t_{(l_1+1)1,1}}, Q_{t_{(l_k+1)1,1}}]] \mid D_{(r_1+1)1,1}[\overline{t_{(r_1+1)1,1}}.S_{t_{(r_1+1)1,1}}]. \tag{B.20}
$$

Therefore, the following holds:

$$
\begin{aligned}
R_1 \equiv\, & \llbracket E_1\rrbracket_\varepsilon\Big[\llbracket G_{1,1}\rrbracket_{\rho_1}\big[\prod_{i=1}^{l_k}\llbracket C_{i,1,1}\rrbracket_{\rho'_{1,1}}[I^{(p)}_{t_{i,1,1}}] \mid \prod_{j=1}^{r_k}\llbracket D_{j,1,1}\rrbracket_{\rho'_{1,1}}[h_{t_{j,1,1}}.\llbracket S_{t_{j,1,1}}\rrbracket_{\rho''_{1,1}}] \mid M'_1\big] \mid M'_2\Big] \mid M'_3\\
\equiv\, & \llbracket E_1\rrbracket_\varepsilon\Big[\llbracket G'_{1,1}\rrbracket_{\rho_1}\big[\prod_{i=1}^{l_k}\llbracket C_{i,1,1}\rrbracket_{\rho'_{1,1}}[I^{(p)}_{t_{i,1,1}}] \mid \prod_{j=1}^{r_k}\llbracket D_{j,1,1}\rrbracket_{\rho'_{1,1}}[h_{t_{j,1,1}}.\llbracket S_{t_{j,1,1}}\rrbracket_{\rho''_{1,1}}]\\
& \mid \llbracket C_{(l_1+1)1,1}\rrbracket_{\rho'_{1,1}}[t_{(l_1+1)1,1}[\llbracket P\rrbracket_{t_{(l_1+1)1,1},\rho''_{1,1}}] \mid t_{(l_1+1)1,1}.(\mathtt{extr}\langle\!\langle t_{(l_1+1)1,1}, p_{t_{(l_1+1)1,1},\rho''_{1,1}}, p_{\rho''_{1,1}}\rangle\!\rangle \mid p_{\rho''_{1,1}}[\llbracket Q\rrbracket_\varepsilon])]\\
& \mid \llbracket D_{(r_1+1)1,1}\rrbracket_{\rho'_{1,1}}[\overline{t_{(r_1+1)1,1}}.h_{t_{(r_1+1)1,1}}.\llbracket S_{t_{(r_1+1)1,1}}\rrbracket_{\rho''_{1,1}}] \mid M'_1\big] \mid M'_2\Big] \mid M'_3.
\end{aligned}
$$

For process R, which is obtained from $R_1 \longrightarrow R$, one possible reduction is caused by synchronization on name input $t_{(r_1+1)1,1}$, as presented in the following:

$$
\begin{aligned}
R \equiv\, & \llbracket E_1\rrbracket_\varepsilon\Big[\llbracket G'_{1,1}\rrbracket_{\rho_1}\big[\prod_{i=1}^{l_k}\llbracket C_{i,1,1}\rrbracket_{\rho'_{1,1}}[I^{(p)}_{t_{i,1,1}}] \mid \prod_{j=1}^{r_k}\llbracket D_{j,1,1}\rrbracket_{\rho'_{1,1}}[h_{t_{j,1,1}}.\llbracket S_{t_{j,1,1}}\rrbracket_{\rho''_{1,1}}]\\
& \mid \llbracket C_{(l_1+1)1,1}\rrbracket_{\rho'_{1,1}}[I^{(1)}_{t_{(l_1+1)1,1}}] \mid \llbracket D_{(r_1+1)1,1}\rrbracket_{\rho'_{1,1}}[h_{t_{(r_1+1)1,1}}.\llbracket S_{t_{(r_1+1)1,1}}\rrbracket_{\rho''_{1,1}}] \mid M'_1\big] \mid M'_2\Big] \mid M'_3,
\end{aligned} \tag{B.21}
$$

where $I^{(1)}_{t_{(l_1+1)1,1}} = t_{(l_1+1)1,1}[\llbracket P_{t_{(l_1+1)1,1}}\rrbracket_{t_{(l_1+1)1,1},\rho''_{1,1}}] \mid \mathtt{extr}\langle\!\langle t_{(l_1+1)1,1}, p_{t_{(l_1+1)1,1},\rho''_{1,1}}, p_{\rho''_{1,1}}\rangle\!\rangle \mid p_{\rho''_{1,1}}[\llbracket Q_{t_{(l_1+1)1,1}}\rrbracket_\varepsilon]$.
In case when we get (B.21) it follows that $P'' = P'$.
It should be noted that here we considered one particular case. Precisely, we consider scenario where for transaction $t_{(l_1+1)1,1}[P_{t_{(l_1+1)1,1}}, Q_{t_{(l_1+1)1,1}}]$ error notification comes from context $D_{(r_1+1)1,1}[\bullet]$, but that is not the only possible case. For the other cases, when the error notification $\overline{t_{(l_1+1)1,1}}$ comes from some other context, i.e. from $D_{j,1,1}[\bullet]$, $j \in \{1, \dots, r_1\}$ or $C_{i,1,1}[\bullet]$, $i \in \{1, \dots, l_1\}$ or $G'_{1,1}[\bullet]$ or $E_1[\bullet]$, discussion follows similarly.

(6) In this case let us consider that:

$$
G_{1,1}[\bullet] = G'_{1,1}[\bullet] \mid L_{(m_1+1),1,1}\big[u_{(m_1+1),1,1}[F_{(m_k+1)1,1}[\overline{u_{(m_1+1),1,1}}.P_{u_{(m_1+1),1,1}}], Q_{u_{(m_1+1),1,1}}]\big] \tag{B.22}
$$

Therefore, the following holds:

$$
\begin{aligned}
R_1 \equiv\, & \llbracket E_1\rrbracket_\varepsilon\Big[\llbracket G'_{1,1}\rrbracket_{\rho_1}\big[\prod_{c=1}^{m_k}\llbracket L_{c,1,1}\rrbracket_{\rho'_{1,1}}[I^{(q)}_{u_{c,1,1}}]\\
& \mid \llbracket L_{(m_1+1),1,1}\rrbracket_{\rho'_{1,1}}\big[u_{(m_1+1),1,1}[\llbracket F_{(m_k+1),1,1}\rrbracket_{\rho''_{1,1}}[\overline{u_{(m_k+1),1,1}}.h_{u_{(m_1+1),1,1}}.\llbracket P_{u_{(m_1+1),1,1}}\rrbracket_{\rho''_{1,1}}]]\\
& \mid u_{(m_1+1),1,1}.(\mathtt{extr}\langle\!\langle u_{(m_1+1),1,1}, p_{u_{(m_1+1),1,1},\rho''_{1,1}}, p_{\rho''_{1,1}}\rangle\!\rangle \mid p_{\rho''_{1,1}}[\llbracket Q_{u_{(m_1+1),1,1}}\rrbracket_\varepsilon]) \mid M'_1\big] \mid M'_2\Big] \mid M'_3
\end{aligned}
$$

For process R, which is obtained form $R_1 \longrightarrow R$, one possible reduction is caused by a synchronization on name $u_{(m_1+1),1,1}$, as presented in the following:

$$R \equiv [\![E_1]\!]_\varepsilon\Big[[\![G'_{1,1}]\!]_{\rho_1}\big[\prod_{c=1}^{m_k}[\![L_{c,1,1}]\!]_{\rho'_{1,1}}\big[O^{(q)}_{u_{c,1,1}}\big] \mid [\![L_{(m_1+1),1,1}]\!]_{\rho'_{1,1}}\big[O^{(1)}_{u_{(m_1+1),1,1}} \mid M'_1\big] \mid M'_2\Big] \mid M'_3, \tag{B.23}$$

where

$$\begin{aligned} O^{(1)}_{u_{(m_1+1),1,1}} = \; & u_{(m_1+1),1,1}[[\![F_{(m_1+1),1,1}]\!]_{\rho''_{1,1}}[h_{u_{(m_1+1),1,1}}.[\![P_{u_{(m_1+1),1,1}}]\!]_{\rho''_{1,1}}]] \\ & \mid \mathtt{extr}\langle\!\langle u_{(m_1+1),1,1}, p_{u_{(m_1+1),1,1},\rho''_{1,1}}, p_{\rho''_{1,1}}\rangle\!\rangle \mid p_{\rho''_{1,1}}[[\![Q_{u_{(m_1+1),1,1}}]\!]_\varepsilon]. \end{aligned}$$

In case when we get (B.23) it follows that $P'' = P'$. □

Lemma B.11. *Let processes $I^{(p)}_t([\![P'_t]\!]_{t,\rho''}, [\![Q'_t]\!]_\varepsilon)$ and $O^{(q)}_u([\![F]\!]_{\rho''}[h_u.[\![P_u]\!]_{\rho'''}], [\![Q'_u]\!]_\varepsilon)$ be defined as in Definition B.2 and Definition B.3. For any contexts C, D, and L the following holds:*

$$C\big[I^{(p)}_t([\![P'_t]\!]_{t,\rho''}, [\![Q'_t]\!]_\varepsilon)\big] \mid D\big[h_t.[\![S_t]\!]_\rho\big] \longrightarrow^* C\big[[\![\mathsf{extr}(P'_t)]\!]_{\rho'} \mid [\![\langle Q'_t\rangle]\!]_{\rho'}\big] \mid D\big[[\![S_t]\!]_\rho\big] \quad \textit{and} \tag{B.24}$$

$$L\big[O^{(q)}_u([\![F]\!]_{\rho''}[h_u.[\![P_u]\!]_{\rho'''}], [\![Q'_u]\!]_\varepsilon)\big] \longrightarrow^* L\big[[\![\mathsf{extr}(F_1[P_u])]\!]_{\rho'} \mid [\![\langle Q'_u\rangle]\!]_{\rho'}\big] \tag{B.25}$$

Proof. The proof proceeds directly by application of the reduction rules from Fig. 4. □

B.3.4. Operational correspondence

We repeat the statement at page 17:

Theorem 5.8 *(Operational correspondence for $[\![\cdot]\!]_\varepsilon$). Let P be a well-formed process in $\mathcal{C}$.*

(1) *If $P \to P'$ then $[\![P]\!]_\varepsilon \longrightarrow^k [\![P']\!]_\varepsilon$ where for*
  a) *$P \equiv E[C[\overline{a}.P_1] \mid D[a.P_2]]$ and $P' \equiv E[C[P_1] \mid D[P_2]]$ it follows $k = 1$,*
  b) *$P \equiv E[C[t[P_1, Q]] \mid D[\overline{t}.P_2]]$ and $P' \equiv E[C[\mathsf{extr}(P_1) \mid \langle Q\rangle] \mid D[P_2]]$ it follows $k = 4 + \mathtt{pb}(P_1)$,*
  c) *$P \equiv C[u[D[\overline{u}.P_1], Q]]$ and $P' \equiv C[\mathsf{extr}(D[P_1]) \mid \langle Q\rangle]$, it follows $k = 4 + \mathtt{pb}(D[P_1])$,*

  for some contexts C, D, E, processes P_1, Q, P_2 and names t, u.

(2) *If $[\![P]\!]_\varepsilon \longrightarrow^n R$ with $n > 0$ then there is P' such that $P \longrightarrow^* P'$ and $R \longrightarrow^* [\![P']\!]_\varepsilon$.*

Proof. We consider completeness and soundness (Parts (1) and (2)) separately.

(1) **Part (1) – Completeness:** The proof proceeds by induction on the derivation of $P \longrightarrow P'$. We consider three base cases, corresponding to cases a), b) and c) of Proposition 3.1 (page 7). In all cases, we use Definition 5.5 and Lemma B.1 (page 43).
  a) This case concerns an input-output synchronization on a name $a \in \mathcal{N}_s$. Therefore, we observe that $P \equiv E[C[\overline{a}.P_1] \mid D[a.P_2]]$ and $P' \equiv E[C[P_1] \mid D[P_2]]$, and we have the following derivation:

$$\begin{aligned} [\![P]\!]_\varepsilon \;&\equiv\; [\![E\big[C[\overline{a}.P_1] \mid D[a.P_2]\big]]\!]_\varepsilon \\ &=\; [\![E]\!]_\varepsilon\big[[\![C[\overline{a}.P_1] \mid D[a.P_2]]\!]_\rho\big] \\ &=\; [\![E]\!]_\varepsilon[[\![C]\!]_\rho[[\![\overline{a}.P_1]\!]_{\rho'}] \mid [\![D]\!]_\rho[[\![a.P_2]\!]_{\rho''}]] \\ &=\; [\![E]\!]_\varepsilon\big[[\![C]\!]_\rho[\overline{a}.[\![P_1]\!]_{\rho'}] \mid [\![D]\!]_\rho[a.[\![P_2]\!]_{\rho''}]\big] \\ &\longrightarrow\; [\![E]\!]_\varepsilon\big[[\![C]\!]_\rho[[\![P_1]\!]_{\rho'}] \mid [\![D]\!]_\rho[[\![P_2]\!]_{\rho''}]\big] \\ &=\; [\![E]\!]_\varepsilon\big[[\![C[P_1] \mid D[P_2]]\!]_\rho\big] \\ &=\; [\![E\big[C[P_1] \mid D[P_2]\big]]\!]_\varepsilon \\ &\equiv\; [\![P']\!]_\varepsilon \end{aligned} \tag{B.26}$$

  Therefore, the thesis holds with $k = 1$.
  b) This case concerns a synchronization due to an external error notification for a transaction scope. We consider $P \equiv E[C[t[P_1, Q]] \mid D[\overline{t}.P_2]]$, with $m = \mathtt{pb}(P_1)$, and $P' \equiv E[C[\mathsf{extr}(P_1) \mid \langle Q\rangle] \mid D[P_2]]$. We have the following derivation:

$$\begin{array}{lll}
\llbracket P\rrbracket_\varepsilon & \equiv & \llbracket E[C[t[P_1,Q]] \mid D[\bar{t}.P_2]]\rrbracket_\varepsilon \\
& = & \llbracket E\rrbracket_\varepsilon\Big[\llbracket C[t[P_1,Q]]\rrbracket_\rho \mid \llbracket D[\bar{t}.P_2]\rrbracket_\rho\Big] \\
& = & \llbracket E\rrbracket_\varepsilon\Big[\llbracket C\rrbracket_\rho[\llbracket t[P_1,Q]\rrbracket_{\rho'}] \mid \llbracket D\rrbracket_\rho[\llbracket \bar{t}.P_2\rrbracket_{\rho''}]\Big] \\
& = & \llbracket E\rrbracket_\varepsilon\Big[\llbracket C\rrbracket_\rho\Big[t\big[\llbracket P_1\rrbracket_{t,\rho'}\big] \mid t.\big(\mathtt{extr}\langle\!\langle t, p_{t,\rho'}, p_{\rho'}\rangle\!\rangle \mid p_{\rho'}[\llbracket Q\rrbracket_\varepsilon]\big)\Big] \mid \llbracket D\rrbracket_\rho[\bar{t}.h_t.\llbracket P_2\rrbracket_{\rho''}]\Big] \\
& \longrightarrow & \llbracket E\rrbracket_\varepsilon\Big[\llbracket C\rrbracket_\rho\big[I_t^{(1)}(\llbracket P_1\rrbracket_{t,\rho'}, \llbracket Q\rrbracket_\varepsilon)\big] \mid \llbracket D\rrbracket_\rho[h_t.\llbracket P_2\rrbracket_{\rho''}]\Big] \\
& \longrightarrow^{m+2} & \llbracket E\rrbracket_\varepsilon\Big[\llbracket C\rrbracket_\rho\big[[I_t^{(m+3)}(\llbracket P_1\rrbracket_{t,\rho'}, \llbracket Q\rrbracket_\varepsilon)\big] \mid \llbracket D\rrbracket_\rho\big[h_t.\llbracket P_2\rrbracket_{\rho''}\big]\Big] \\
& \longrightarrow & \llbracket E\rrbracket_\varepsilon\Big[\llbracket C\rrbracket_\rho\big[\llbracket \mathsf{extr}(P_1) \mid \langle Q\rangle\rrbracket_{\rho'}\big] \mid \llbracket D\rrbracket_\rho\big[\llbracket P_2\rrbracket_{\rho''}\big]\Big] \\
& = & \llbracket E\rrbracket_\varepsilon\Big[\llbracket C[\mathsf{extr}(P_1) \mid \langle Q\rangle]\rrbracket_\rho \mid \llbracket D[P_2]\rrbracket_\rho\Big] \\
& = & \llbracket E[C[\mathsf{extr}(P_1) \mid \langle Q\rangle] \mid D[P_2]]\rrbracket_\varepsilon \\
& \equiv & \llbracket P'\rrbracket_\varepsilon
\end{array}$$

Since we have that the error notification is external, in $\mathtt{extr}\langle\!\langle t, p_{t,\rho'}, p_{\rho'}\rangle\!\rangle$ (cf. Eq. (13)) we get that $\mathsf{ch}(t, \llbracket P_1\rrbracket_\rho) = \mathbf{0}$. (cf. Lemma B.6 for more details). The order/nature of these reduction steps is as follows:

i) The first synchronization concerns t and $\bar{t}$.

ii) The following $m+2$ synchronizations can be explained as follows:

- First, we have a process relocation through the update of location t, as enforced by the definition of process $\mathtt{extr}$. Process $I_t^{(1)}(\llbracket P_1\rrbracket_{t,\rho'}, \llbracket Q\rrbracket_\varepsilon)$ is as in Definition B.2 (Fig. B.17); as shown in Fig. B.16, there are two possibilities for reduction, depending on m.
- Subsequently, thanks to process

$$\mathtt{out}^{\mathtt{s}}(p_{t,\rho'}, p_{\rho'}\,,\, \mathtt{nl}(p_{t,\rho'}, \llbracket P_1\rrbracket_{t,\rho'})\,,\, t\langle\!\langle\dagger\rangle\!\rangle.\overline{h_t})$$

  we have m reduction steps that relocate processes on location $p_{t,\rho'}$ to location $p_{\rho'}$, as also shown in Fig. B.16.
- The final reduction corresponds to the erasure of the location t with all its contents, obtained by updating prefix $t\langle\!\langle\dagger\rangle\!\rangle$.

iii) Finally, we have a synchronization between h_t and $\overline{h_t}$, which serves to signal that all synchronizations related to location t have been completed.

Therefore, we can conclude that for $\llbracket P\rrbracket_\varepsilon \longrightarrow^k \llbracket P'\rrbracket_\varepsilon$ such that $k = 4+m$.

c) This case concerns a synchronization due to an internal error notification (i.e., the error comes from the default activity of transaction). Here we have $P \equiv C[t[D[\overline{u}.P_1], Q]]$, with $m = \mathtt{pb}(D[P_1])$, and $P' \equiv C[\mathsf{extr}(D[P_1]) \mid \langle Q\rangle]$. Then we have the following derivation:

$$\begin{array}{lll}
\llbracket P\rrbracket_\varepsilon & \equiv & \llbracket C\big[u[D[\overline{u}.P_1], Q]\big]\rrbracket_\varepsilon \\
& = & \llbracket C\rrbracket_\varepsilon\big[\llbracket u[D[\overline{u}.P_1], Q]\rrbracket_\rho\big] \\
& = & \llbracket C\rrbracket_\varepsilon\Big[u\big[\llbracket D[\overline{u}.P_1]\rrbracket_{u,\rho}\big] \mid u.\big(\mathtt{extr}\langle\!\langle t, p_{u,\rho}, p_\rho\rangle\!\rangle \mid p_\rho[\llbracket Q\rrbracket_\varepsilon]\big)\Big]\Big] \\
& = & \llbracket C\rrbracket_\varepsilon\Big[u\big[\llbracket D\rrbracket_{u,\rho}[\overline{u}.h_u.\llbracket P_1\rrbracket_{\rho'}]\big] \mid u.(\mathtt{extr}\langle\!\langle u, p_{u,\rho}, p_\rho\rangle\!\rangle \mid p_\rho[\llbracket Q\rrbracket_\varepsilon])\Big] \\
& \longrightarrow & \llbracket C\rrbracket_\varepsilon\big[O_u^{(1)}(\llbracket D\rrbracket_{u,\rho}[h_u.\llbracket P_1\rrbracket_{\rho'}], \llbracket Q\rrbracket_\varepsilon)\big] \\
& \longrightarrow^{m+2} & \llbracket C\rrbracket_\varepsilon\big[O_u^{(m+3)}(\llbracket D\rrbracket_{u,\rho}[h_u.\llbracket P_1\rrbracket_{\rho'}], \llbracket Q\rrbracket_\varepsilon)\big] \\
& \longrightarrow & \llbracket C\rrbracket_\varepsilon\big[O_u^{(m+4)}(\llbracket D\rrbracket_{u,\rho}[h_u.\llbracket P_1\rrbracket_{\rho'}], \llbracket Q\rrbracket_\varepsilon)\big] \\
& = & \llbracket C\rrbracket_\varepsilon\big[\llbracket \mathsf{extr}(D[P_1])\rrbracket_\rho \mid p_\rho[\llbracket Q\rrbracket_\varepsilon]\big] \\
& = & \llbracket C\big[\mathsf{extr}(D[P_1]) \mid \langle Q\rangle\big]\rrbracket_\varepsilon \\
& \equiv & \llbracket P'\rrbracket_\varepsilon
\end{array}$$

Process $O_u^{(q)}(\llbracket D\rrbracket_{u,\rho}[h_u.\llbracket P_1\rrbracket_{\rho'}], \llbracket Q\rrbracket_\varepsilon)$, where $q \in \{1, \dots, m+4\}$, is as in Definition B.3. It should be noted that the location on name u and its content will be erased before interaction on name h_u and $\overline{h_u}$ (cf. Fig. B.18 for $q = (m+2)$ and $q = (m+3)$). Therefore, in this case, the role of function $\mathsf{ch}(u, \cdot)$ is central: indeed, $\mathsf{ch}(u, \llbracket D\rrbracket_{u,\rho}\big[h_u.\llbracket P_1\rrbracket_{\rho'}\big])$ provides the input h_u which is necessary to achieve operational correspondence.

The order/nature/number of reduction steps can be explained as in Case b) above. We can then conclude that $\llbracket P\rrbracket_\varepsilon \longrightarrow^k \llbracket P'\rrbracket_\varepsilon$ such that $k = 4+m$.

(2) **Part (2) – Soundness:** Given $\llbracket P\rrbracket_\varepsilon \longrightarrow^n R$, by Lemma B.10, process R has the following form:

$$R \equiv \prod_{w=1}^{z} \llbracket E_w\rrbracket_\varepsilon\Big[\prod_{k=1}^{s_w} \llbracket G_{k,w}\rrbracket_{\rho_w}\Big[\prod_{i=1}^{l_k} \llbracket C_{i,k,w}\rrbracket_{\rho'_{k,w}}\big[I_{t_{i,k,w}}^{(p)}\big] \mid \prod_{j=1}^{r_k} \llbracket D_{j,k,w}\rrbracket_{\rho'_{k,w}}\big[h_{t_{j,k,w}}.\llbracket S_{t_{j,k,w}}\rrbracket_{\rho''_{k,w}}\big] \\ \mid \prod_{c=1}^{m_k} \llbracket L_{c,k,w}\rrbracket_{\rho'_{k,w}}\big[O_{u_{c,k,w}}^{(q)}\big]\Big]\Big].$$

$$\begin{tikzcd}
\llbracket P\rrbracket_{\varepsilon} \arrow[dr, "*"] \arrow[dd, "n"'] & \\
& \llbracket P''\rrbracket_{\varepsilon} \arrow[dd, "*"] \\
R \arrow[dr, "*"'] & \\
& \llbracket P'\rrbracket_{\varepsilon}
\end{tikzcd}$$

Fig. B.20. Diagram of the proof of soundness for $\llbracket\cdot\rrbracket_{\varepsilon}$.

Also by Lemma B.10, we have $P \longrightarrow^{*} P''$ where

$$\begin{aligned}
P'' \equiv \prod_{w=1}^{z} E_w\Big[\prod_{k=1}^{s_w} G_{k,w}\big[\prod_{i=1}^{l_k} C_{i,k,w}\big[t_{i,k,w}[P_{t_{i,k,w}}, Q_{t_{i,k,w}}]\big] \mid \prod_{j=1}^{r_k} D_{j,k,w}\big[\overline{t_{j,k,w}}.S_{t_{j,k,w}}\big] \\
\mid \prod_{c=1}^{m_k} L_{c,k,w}\big[u_{c,k,w}[F_{c,k,w}[\overline{u_{c,k,w}}.P_{u_{c,k,w}}], Q_{u_{c,k,w}}]\big]\big]\Big],
\end{aligned}$$

where by successive application of completeness it follows that $\llbracket P\rrbracket_{\varepsilon} \longrightarrow^{*} \llbracket P''\rrbracket_{\varepsilon}$.
By Lemma B.11, i.e., by l_k successive applications of (B.24) and m_k successive applications of (B.25) on process R, it follows that:

$$\begin{aligned}
R \longrightarrow^{*} & \prod_{w=1}^{z} \llbracket E_w\rrbracket_{\varepsilon}\Big[\prod_{k=1}^{s_w} \llbracket G_{k,w}\rrbracket_{\rho_w}\big[\prod_{i=1}^{l_k} \llbracket C_{i,k,w}\rrbracket_{\rho'_{k,w}}\big[\llbracket \mathsf{extr}(P'_{t_{i,k,w}})\rrbracket_{\rho''_{k,w}} \mid \llbracket\langle Q'_{t_{i,k,w}}\rangle\rrbracket_{\rho''_{k,w}}\big] \\
& \mid \prod_{j=1}^{r_k} \llbracket D_{j,k,w}\rrbracket_{\rho'_{k,w}}\big[\llbracket S_{t_{j,k,w}}\rrbracket_{\rho''_{k,w}}\big] \mid \prod_{c=1}^{m_k} \llbracket L_{c,k,w}\rrbracket_{\rho'_{k,w}}\big[\llbracket \mathsf{extr}(F_{c,k,w}[P_{u_{c,k,w}}])\rrbracket_{\rho''_{k,w}} \mid \llbracket\langle Q'_{u_{c,k,w}}\rangle\rrbracket_{\rho''_{k,w}}\big]\big]\Big] \\
& \equiv \Big\llbracket \prod_{w=1}^{z} E_w\Big[\prod_{k=1}^{s_w} G_{k,w}\big[\prod_{i=1}^{l_k} C_{i,k,w}\big[\mathsf{extr}(P'_{t_{i,k,w}}) \mid \langle Q'_{t_{i,k,w}}\rangle\big] \mid \prod_{j=1}^{r_k} D_{j,k,w}\big[S_{t_{j,k,w}}\big] \\
& \mid \prod_{c=1}^{m_k} L_{c,k,w}\big[\mathsf{extr}(F_{c,k,w}[P_{u_{c,k,w}}]) \mid \langle Q'_{u_{c,k,w}}\rangle\big]\big]\Big]\Big\rrbracket_{\varepsilon} \\
& \equiv \llbracket P'\rrbracket_{\varepsilon}.
\end{aligned}$$

Therefore, it follows that

$$\begin{aligned}
P' \equiv \prod_{w=1}^{z} E_w\Big[\prod_{k=1}^{s_w} G_{k,w}\big[\prod_{i=1}^{l_k} C_{i,k,w}\big[\mathsf{extr}(P'_{t_{i,k,w}}) \mid \langle Q'_{t_{i,k,w}}\rangle\big] \mid \prod_{j=1}^{r_k} D_{j,k,w}\big[S_{t_{j,k,w}}\big] \\
\mid \prod_{c=1}^{m_k} L_{c,k,w}\big[\mathsf{extr}(F_{c,k,w}[P_{u_{c,k,w}}]) \mid \langle Q'_{u_{c,k,w}}\rangle\big]\big]\Big].
\end{aligned}$$

Also, by Proposition 3.1, i.e., by l_k successive applications of case b) and m_k successive applications of case c) on process P'', it follows that $P'' \longrightarrow^{*} P'$.
By successive application of **(B.3.4) – Completeness** on the derivation $P'' \longrightarrow^{*} P'$ it follows that $\llbracket P''\rrbracket_{\varepsilon} \longrightarrow^{*} \llbracket P'\rrbracket_{\varepsilon}$. The proof scheme is shown in Fig. B.20. □

Example B.12. The example presented in Fig. B.21 illustrates the proof of soundness (Fig. B.20).

$$\begin{tikzcd}
\begin{array}{c}[\![P]\!]_\varepsilon = [\![t[\langle a \mid \overline{a}\rangle,\mathbf{0}] \mid \overline{t} \mid u[\overline{u},\mathbf{0}]]\!]_\varepsilon \\ = t\big[p_{t,\varepsilon}[a\mid\overline{a}]\big] \mid t.\big(\mathtt{extr}\langle\!\langle t, p_{t,\varepsilon}\,,\, p_\varepsilon\rangle\!\rangle \mid p_\varepsilon[\mathbf{0}]\big) \mid \overline{t}.h_t \\ \mid u\big[\overline{u}.h_u\big] \mid u.\big(\mathtt{extr}\langle\!\langle u, p_{u,\varepsilon}\,,\, p_\varepsilon\rangle\!\rangle \mid p_\varepsilon[\mathbf{0}]\big)\end{array} \arrow[d] \arrow[r, "*"] & {[\![\langle a \mid \overline{a}\rangle \mid \langle\mathbf{0}\rangle \mid u[\overline{u},\mathbf{0}]]\!]_\varepsilon} \arrow[d, "*"] \\
\begin{array}{c} I_t^{(1)} \mid h_t \mid u\big[\overline{u}.h_u\big] \mid u.\big(\mathtt{extr}\langle\!\langle u, p_{u,\varepsilon}\,,\, p_\varepsilon\rangle\!\rangle \mid p_\varepsilon[\mathbf{0}]\big) \\ \Big(I_t^{(1)} \equiv t\big[p_{t,\varepsilon}[a \mid \overline{a}]\big] \mid \mathtt{extr}\langle\!\langle t, p_{t,\varepsilon}\,,\, p_\varepsilon\rangle\!\rangle \mid p_\varepsilon[\mathbf{0}]\Big)\end{array} \arrow[d] & {[\![P'']\!]_\varepsilon = [\![\langle a \mid \overline{a}\rangle \mid \langle\mathbf{0}\rangle \mid \langle\mathbf{0}\rangle]\!]_\varepsilon} \arrow[ddd, "*"] \\
\begin{array}{c} I_t^{(1)} \mid h_t \mid O_u^{(1)} \\ \Big(O_u^{(1)} \equiv u\big[h_u\big] \mid \mathtt{extr}\langle\!\langle u, p_{u,\varepsilon}\; p_\varepsilon\rangle\!\rangle \mid p_\varepsilon[\mathbf{0}]\Big)\end{array} \arrow[d] & \\
\begin{array}{c} I_t'^{(1)} \mid h_t \mid O_u^{(1)} \\ \Big(I_t'^{(1)} \equiv t\big[p_{t,\varepsilon}[\mathbf{0} \mid \mathbf{0}]\big] \mid \mathtt{extr}\langle\!\langle t, p_{t,\varepsilon}\,,\, p_\varepsilon\rangle\!\rangle \mid p_\varepsilon[\mathbf{0}]\Big)\end{array} \arrow[d, "3"] & \\
R = p_\varepsilon[\mathbf{0} \mid \mathbf{0}] \mid p_\varepsilon[\mathbf{0}] \mid O_u^{(1)} \arrow[r, "*"] & {[\![P']\!]_\varepsilon = [\![\langle\mathbf{0} \mid \mathbf{0}\rangle \mid \langle\mathbf{0}\rangle \mid \langle\mathbf{0}\rangle]\!]_\varepsilon}
\end{tikzcd}$$

Fig. B.21. Example for operational soundness.

B.4. Proof of divergence reflection results: Theorem 5.12

In this section we shall prove that divergence reflection holds for the translation $[\![\cdot]\!]_\rho$. The proof relies on the following lemma, which was given in page 20:

Lemma 5.11. *Let $\{R_i\}_{i\geq 0}$ be a sequence of adaptable processes such that $R_i \longrightarrow R_{i+1}$, with $R_0 = [\![P_0]\!]_\rho$, for some compensable process P_0 and path ρ. Then for every $i \geq 1$ there is P_i such that*

(i) *$R_i \longrightarrow^* [\![P_i]\!]_\rho$,*
(ii) *$P_{i-1} = P_i$ or $P_{i-1} \longrightarrow P_i$, and*
(iii) *$R_i \not\equiv R_{i+1} \not\equiv \ldots \not\equiv R_{i+m}$ and $P_i = P_{i+1} = \ldots = P_{i+m}$ imply $m \leq 4 + \mathtt{npb}(P_0)$.*

Proof. The proof for (i) and (ii) proceeds by induction on i.
Base case: Assume that $i = 1$. By the proof of Lemma B.10, i.e. its ***Base case***, we have three cases:

a) $P_0 \equiv E[C[a.P_1'] \mid D[\overline{a}.P_2]]$ and $R_1 \equiv [\![E]\!]_\varepsilon\Big[[\![C]\!]_\rho[[\![P_1']\!]_{\rho'}] \mid [\![D]\!]_\rho[[\![P_2]\!]_{\rho''}]\Big] = [\![P_1]\!]_\rho$, it follows $P_0 \longrightarrow P_1$ (cf. Proposition 3.1 (a)).
b) $P_0 \equiv E\Big[C[t[P_2, Q]] \mid D[\overline{t}.P_1']]$ and
$R_1 \equiv [\![E]\!]_\varepsilon\Big[[\![C]\!]_\rho[t\big[[\![P_2']\!]_{t,\rho'}\big] \mid \mathtt{extr}\langle\!\langle t, p_{t,\rho'}, p_{\rho'}\rangle\!\rangle \mid p_{\rho'}[[\![Q']\!]_\varepsilon]] \mid [\![D]\!]_\rho[h_t.[\![P_1']\!]_{\rho''}]\Big]$. There is P_1 such that by Lemma B.11 (B.24) it follows $R_1 \longrightarrow^* [\![P_1]\!]_\rho$. Also, it follows $P_0 \longrightarrow P_1$ (cf. Proposition 3.1 (b)).
c) $P_0 \equiv C[u[D[\overline{u}.P_1'], Q]]$ and $R_1 \equiv [\![C]\!]_\varepsilon\Big[u\big[[\![D]\!]_{u,\rho}[h_u.[\![P_1']\!]_{\rho'}]\big] \mid \mathtt{extr}\langle\!\langle u, p_{u,\rho'}, p_{\rho'}\rangle\!\rangle \mid p_{\rho'}[[\![Q']\!]_\varepsilon]\Big]$. There is P_1 such that by Lemma B.11 (B.25) it follows $R_1 \longrightarrow^* [\![P_1]\!]_\rho$. Also, it follows $P_0 \longrightarrow P_1$ (cf. Proposition 3.1 (c)).

Inductive step: By inductive hypothesis, there are processes $P_1, \ldots, P_{i-1}, P_i$ such that $R_{i-1} \longrightarrow^* [\![P_{i-1}]\!]_\rho$ and either $P_{i-1} = P_i$ or $P_{i-1} \longrightarrow P_i$. Let us now consider $R_i \longrightarrow R_{i+1}$ (i.e., $[\![P_0]\!]_\rho \longrightarrow^i R_i \longrightarrow R_{i+1}$). By the proof of Lemma B.10, i.e., its ***Inductive step***, we get that there is P_{i+1} such that either $P_i = P_{i+1}$ or $P_i \longrightarrow P_{i+1}$ (cf. for example (B.16) and (B.18)). By Lemma B.11 it follows $R_{i+1} \longrightarrow^* [\![P_{i+1}]\!]_\rho$.

Now, we are going to prove the last assertion in the statement. In the following, we give guidelines on how to obtain the proof since it follows from (the proof) of Lemma B.10:

(1) The form of process R_i is given with (B.14), and process P_i has a form given with (B.15).
(2) In the proof, its ***Inductive step***, we consider only cases such that $R_i \not\equiv R_{i+1}$ and $P_i = P_{i+1}$. Therefore, we consider the cases in which intermediate processes $I^{(p)}_{t_{i,k,w}}$ and $O^{(q)}_{u_{c,k,w}}$ inside process R_i (cf. Definition B.2 and Definition B.3, respectively) have been changed.
(3) From Fig. B.17 and Fig. B.18 we obtain the form and the number of all intermediate processes. We remind the reader that the number of intermediate processes directly depends on the number of protected blocks in the observed transaction, more precisely in its compensation activity (cf. for example Fig. B.16).
(4) We conclude, for each $l \in \{1,\dots,m\}$ it follows that m is at most $4+\mathtt{npb}(P_0)$, i.e., $m = 4+\mathtt{pb}(Q') \le 4+\mathtt{npb}(P_0)$, for some Q' that appears in P_0. $\square$

B.5. *Proof of success sensitiveness results: Theorem 5.13*

Here we shall prove that success sensitiveness holds for the translation $[\![\cdot]\!]_\rho$. The first part of the statement

$$P \Downarrow \text{ implies } [\![P]\!]_\rho \Downarrow$$

follows directly from operational completeness (Theorem 5.8(1)) and Lemma B.1. The proof for the opposite direction

$$[\![P]\!]_\rho \Downarrow \text{ implies } P \Downarrow$$

is derived through the following steps:

- By Definition 5.10, if $[\![P]\!]_\rho \Downarrow$ then $[\![P]\!]_\rho \longrightarrow^k R$ and $R = C[\checkmark]$ for some context $C[\bullet]$.
- By Lemma B.10, we conclude that process R has the form given in (B.14).
- Assuming that $R = C[\checkmark]$, we identify all possible positions of $\checkmark$ in the form (B.14). For that purpose, we introduce some auxiliary lemmas:
  - By Lemma B.7, either $\checkmark$ appears at top level of some context (in parallel), in a form $[\![C'[\checkmark]]\!]_\rho$, or it is nested inside some locations. There are four additional nested places that we consider separately and list in the following items.
  - Lemma B.13 considers the case with $I^{(p)}_t([\![P_1]\!]_{t,\rho}, [\![Q_1]\!]_\varepsilon) = C''[\checkmark]$ and $\mathtt{nl}(p_{t,\rho}, [\![P_1]\!]_{t,\rho}) = 0$ and $p \in \{1,2,3\}$, where $I^{(p)}_t([\![P_1]\!]_{t,\rho}, [\![Q_1]\!]_\varepsilon)$ is given in Fig. B.17.
  - Lemma B.14 considers the case with $I^{(p)}_t([\![P_1]\!]_{t,\rho}, [\![Q_1]\!]_\varepsilon) = C''[\checkmark]$ and $\mathtt{nl}(p_{t,\rho}, [\![P_1]\!]_{t,\rho}) = m > 0$ and $p \in \{1,2,\dots,m+3\}$, where $I^{(p)}_t([\![P_1]\!]_{t,\rho}, [\![Q]\!]_\varepsilon)$ is given in Fig. B.17.
  - Lemma B.15 considers the case with $O^{(q)}_u([\![F]\!]_\rho[h_u.[\![P_1]\!]_{\rho'}], [\![Q_1]\!]_\varepsilon) = C''[\checkmark]$ and $\mathtt{nl}(p_{u,\rho}, [\![F]\!]_\rho[h_u.[\![P_1]\!]_{\rho'}]) = 0$ and $p \in \{1,2,3,4\}$, where $O^{(q)}_u([\![F]\!]_\rho[h_u.[\![P_2]\!]_{\rho'}], [\![Q_1]\!]_\varepsilon)$ is given in Fig. B.18.
  - Lemma B.16 considers the case with $O^{(q)}_u([\![F]\!]_\rho[h_u.[\![P_1]\!]_{\rho'}], [\![Q_1]\!]_\varepsilon) = C''[\checkmark]$ and $\mathtt{nl}(p_{u,\rho}, [\![F]\!]_\rho[h_u.[\![P_1]\!]_{\rho'}]) = m > 0$ and $p \in \{1,\dots,m+4\}$, where $O^{(q)}_u([\![F]\!]_\rho[h_u.[\![P_2]\!]_{\rho'}], [\![Q_1]\!]_\varepsilon)$ is given in Fig. B.18.
- Finally, after identifying the place of $\checkmark$, using (B.15) of Lemma B.10, we get the proof.

We proceed to introduce the auxiliary lemmas that consider nested appearances of $\checkmark$.

Lemma B.13. *Let t be a name, ρ a path, and P, Q well-formed compensable processes such that $\mathtt{nl}(p_{t,\rho}, [\![P]\!]_{t,\rho}) = 0$. If $I^{(p)}_t([\![P]\!]_{t,\rho}, [\![Q]\!]_\varepsilon) = C[\checkmark]$, for $p \in \{1,2,3\}$ and some context $C[\bullet]$, then*

(i) *either $[\![P]\!]_{t,\rho} = C_1[\checkmark]$,*
(ii) *or $[\![Q]\!]_\varepsilon = C_1[\checkmark]$*

for some context $C_1[\bullet]$.

Proof. There are three possible forms of $I^{(p)}_t([\![P]\!]_{t,\rho}, [\![Q]\!]_\varepsilon)$, given in the first three rows of Fig. B.17.

- $p \in \{1,2\}$: If $t\big[[\![P]\!]_{t,\rho}\big] \mid R \mid p_\rho[[\![Q]\!]_\varepsilon] = C[\checkmark]$ and $(R \equiv t\{(Y).t[Y] \mid \mathsf{ch}(t,Y) \mid t\{\dagger\}.\overline{h_t}\}$ or $R = t\{\dagger\}.\overline{h_t})$, by Definition 3.6, we have the following two possibilities:
  (i) $C[\bullet] = t\big[C_1[\bullet]\big] \mid R \mid p_\rho[[\![Q]\!]_\varepsilon]$ and $C_1[\checkmark] = [\![P]\!]_{t,\rho}$, or
  (ii) $C[\bullet] = t\big[[\![P]\!]_{t,\rho}\big] \mid R \mid p_\rho[C_1[\bullet]]$ and $C_1[\checkmark] = [\![Q]\!]_\varepsilon$.
- $p = 3$: If $\overline{h_t} \mid p_\rho[[\![Q]\!]_\varepsilon] = C[\checkmark]$, by Definition 3.6, $C[\bullet] = \overline{h_t} \mid p_\rho[C_1[\bullet]]$ and therefore $C_1[\checkmark] = [\![Q]\!]_\varepsilon$. $\square$

Lemma B.14. *Let t be a name, ρ a path, and P, Q well-formed compensable processes such that $[\![P]\!]_{t,\rho} = \prod_{k=1}^{m} p_{t,\rho}[[\![P'_k]\!]_\varepsilon] \mid S$ with $\mathtt{nl}(p_{t,\rho}, [\![P]\!]_{t,\rho}) = m$. If $I^{(p)}_t([\![P]\!]_{t,\rho}, [\![Q]\!]_\varepsilon) = C[\checkmark]$, for $p \in \{1,\dots,m+3\}$ and some context $C[\bullet]$, then*

(i) $[\![P'_k]\!]_\varepsilon = C_1[\checkmark]$, *or*
(ii) $[\![Q]\!]_\varepsilon = C_1[\checkmark]$, *or*
(iii) $S = C_1[\checkmark]$

for some context $C_1[\bullet]$ *and* $k \in \{1,\dots,m\}$.

Proof. The proof is similar to the proof of Lemma B.13 and follows directly from Definition 3.6, and Fig. B.17. □

Lemma B.15. *Let u be a name, ρ a path, and P, Q well-formed compensable processes such that* $\mathtt{nl}(p_{u,\rho}, [\![F]\!]_\rho[h_u.[\![P]\!]_{\rho'}]) = 0$. *If* $O_u^{(q)}([\![F]\!]_\rho[h_u.[\![P]\!]_{\rho'}], [\![Q]\!]_\varepsilon) = C[\checkmark]$, *for* $p \in \{1,2,3,4\}$ *and some context* $C[\bullet]$, *then*

(i) *either* $[\![F]\!]_\rho[h_u.[\![P]\!]_{\rho'}] = C_1[\checkmark]$,
(ii) *or* $[\![Q]\!]_\varepsilon = C_1[\checkmark]$

for some context $C_1[\bullet]$.

Proof. The proof is similar to the proof of Lemma B.13 and follows directly from Definition 3.6, and Fig. B.18. □

Lemma B.16. *Let u be a name, ρ a path, and P, Q well-formed compensable processes such that* $[\![F]\!]_\rho[h_u.[\![P]\!]_{\rho'}] = \prod_{k=1}^{m} p_{u,\rho}[[\![P'_k]\!]_\varepsilon] \mid S$ *with* $\mathtt{nl}(p_{u,\rho}, [\![F]\!]_\rho[h_u.[\![P]\!]_{\rho'}]) = m$. *If* $O_u^{(q)}([\![F]\!]_\rho[h_u.[\![P]\!]_{\rho'}], [\![Q]\!]_\varepsilon) = C[\checkmark]$, *for* $p \in \{1,\dots,m+4\}$ *and some context* $C[\bullet]$, *then*

(i) $[\![F]\!]_\rho[h_u.[\![P]\!]_{\rho'}] = C_1[\checkmark]$, *or*
(ii) $[\![Q]\!]_\varepsilon = C_1[\checkmark]$, *or*
(iii) $[\![P'_k]\!]_\varepsilon = C_1[\checkmark]$,

for some context $C_1[\bullet]$ *and* $k \in \{1,\dots,m+4\}$.

Proof. Similar to the proof of Lemma B.13 and follows directly from Definition 3.6 and Fig. B.18. □

Now we repeat the statement at page 20:

Theorem 5.13 *(Success sensitiveness for $[\![\cdot]\!]_\rho$). Let P be a well-formed compensable process and ρ an arbitrary path. Then $P \Downarrow$ if and only if $[\![P]\!]_\rho \Downarrow$.*

Proof. ($\Rightarrow$) Let $P \Downarrow$, i.e., $P \longrightarrow^* P'$ and $P' = C[\checkmark]$. By Theorem 5.8 (B.3.4) – Completeness we have that $[\![P]\!]_\rho \longrightarrow^k [\![P']\!]_\rho = [\![C[\checkmark]]\!]_\rho$. By Convention 5.7 and Lemma B.1 it follows:

$$[\![C]\!]_\rho[\checkmark] = [\![C[\bullet]]\!]_\rho[[\![\checkmark]\!]_{\rho'}],$$

where ρ' is a path to hole in context $C[\bullet]$. By $[\![\checkmark]\!]_\rho = \checkmark$ we have that $[\![P]\!]_\rho \longrightarrow^k [\![C[\bullet]]\!]_\rho[\checkmark]$. This implies that $[\![P]\!]_\rho \Downarrow$.
($\Leftarrow$) Conversely, let $[\![P]\!]_\rho \Downarrow$, i.e., $[\![P]\!]_\rho \longrightarrow^k R$ and $R \equiv C[\checkmark]$. By Lemma B.10 it follows:

$$\begin{aligned} C[\checkmark] \equiv \prod_{w=1}^{z} [\![E_w]\!]_\varepsilon \Big[\prod_{k=1}^{s_w} [\![G_{k,w}]\!]_{\rho_w} \big[\prod_{i=1}^{l_k} [\![C_{i,k,w}]\!]_{\rho'_{k,w}} [I^{(p)}_{t_{i,k,w}}] \mid \prod_{j=1}^{r_k} [\![D_{j,k,w}]\!]_{\rho'_{k,w}} \big[h_{t_{j,k,w}}.[\![S_{t_{j,k,w}}]\!]_{\rho''_{k,w}}\big] \\ \mid \prod_{c=1}^{m_k} [\![L_{c,k,w}]\!]_{\rho'_{k,w}} \big[O^{(q)}_{u_{c,k,w}}\big]\big]\Big] \end{aligned} \tag{B.27}$$

By Lemma B.7, Lemma B.13, Lemma B.14, Lemma B.15, and Lemma B.16 we analyze all possible places where $\checkmark$ occurs in (B.27). By Lemma B.7,

(1) either

$$\begin{aligned} C[\checkmark] \equiv [\![E'']\!]_\varepsilon[\checkmark] \mid \prod_{w=1}^{z} [\![E'_w]\!]_\varepsilon \Big[\prod_{k=1}^{s_w} [\![G_{k,w}]\!]_{\rho_w} \big[\prod_{i=1}^{l_k} [\![C_{i,k,w}]\!]_{\rho'_{k,w}} [I^{(p)}_{t_{i,k,w}}] \mid \prod_{j=1}^{r_k} [\![D_{j,k,w}]\!]_{\rho'_{k,w}} \big[h_{t_{j,k,w}}.[\![S_{t_{j,k,w}}]\!]_{\rho''_{k,w}}\big] \\ \mid \prod_{c=1}^{m_k} [\![L_{c,k,w}]\!]_{\rho'_{k,w}} \big[O^{(q)}_{u_{c,k,w}}\big]\big] \end{aligned} \tag{B.28}$$

(2) or, there are $\omega \in \{1, \dots, z\}$ and $C_1[\bullet]$ such that

$$C_1[\checkmark] \equiv \prod_{k=1}^{s_w} [\![G_{k,w}]\!]_{\rho_w} \Big[\prod_{i=1}^{l_k} [\![C_{i,k,w}]\!]_{\rho'_{k,w}} \big[I^{(p)}_{t_{i,k,w}}\big] \mid \prod_{j=1}^{r_k} [\![D_{j,k,w}]\!]_{\rho'_{k,w}} \big[h_{t_{j,k,w}} \cdot [\![S_{t_{j,k,w}}]\!]_{\rho''_{k,w}}\big] \mid \prod_{c=1}^{m_k} [\![L_{c,k,w}]\!]_{\rho'_{k,w}} \big[O^{(q)}_{u_{c,k,w}}\big]\Big]\Big] \tag{B.29}$$

By Lemma B.7,
(2.1) either

$$C_1[\checkmark] \equiv [\![G''_w]\!]_{\rho_w}[\checkmark] \mid \prod_{k=1}^{s_w} [\![G'_{k,w}]\!]_{\rho_w} \Big[\prod_{i=1}^{l_k} [\![C_{i,k,w}]\!]_{\rho'_{k,w}} \big[I^{(p)}_{t_{i,k,w}}\big] \mid \prod_{j=1}^{r_k} [\![D_{j,k,w}]\!]_{\rho'_{k,w}} \big[h_{t_{j,k,w}} \cdot [\![S_{t_{j,k,w}}]\!]_{\rho''_{k,w}}\big] \mid \prod_{c=1}^{m_k} [\![L_{c,k,w}]\!]_{\rho'_{k,w}} \big[O^{(q)}_{u_{c,k,w}}\big]\Big]\Big] \tag{B.30}$$

(2.2) or, there are $C_2[\bullet]$ and $k \in \{1, \dots, s_\omega\}$ such that one of the following three cases holds:
(2.2.1) $C_2[\checkmark] \equiv [\![C_{i,k,w}]\!]_{\rho'_{k,w}} [I^{(p)}_{t_{i,k,w}}]$:
(2.2.1.1) either $C_2[\checkmark] \equiv [\![C''_{i,k,w}]\!]_{\rho'_{k,w}}[\checkmark] \mid \prod_{i=1}^{l_k} [\![C'_{i,k,w}]\!]_{\rho'_{k,w}} [I^{(p)}_{t_{i,k,w}}]$
(2.2.1.2) or, there are $C_3[\bullet]$ and $i \in \{1, \dots, l_k\}$ such that
$C_3[\checkmark] = I^{(p)}_{t_{i,k,w}}([\![P'_{t_{i,k,w}}]\!]_{t,\rho''}, [\![Q'_{t_{i,k,w}}]\!]_{\varepsilon})$.
Assume that $\mathtt{nl}(p_{t,\rho''}, [\![P'_{t_{i,k,w}}]\!]_{t,\rho''}) = 0$ and $p \in \{1, 2, 3\}$ (other cases are similar). By Lemma B.14,
(2.2.1.2.1) $[\![P'_{t_{i,k,w}}]\!]_{t,\rho''} = C_4[\checkmark]$, or
(2.2.1.2.2) $[\![Q]\!]_{\varepsilon} = C_4[\checkmark]$
for some $C_4[\bullet]$.
(2.2.2) $C_2[\checkmark] \equiv [\![D_{j,k,w}]\!]_{\rho'_{k,w}} \Big[h_{t_{j,k,w}} \cdot [\![S_{t_{j,k,w}}]\!]_{\rho''_{k,w}}\Big]$: By Lemma B.7,

$$[\![D_{j,k,w}]\!]_{\rho'_{k,w}} \big[h_{t_{j,k,w}} \cdot [\![S_{t_{j,k,w}}]\!]_{\rho''_{k,w}}\big] = [\![D'_{j,k,w}]\!]_{\rho'_{k,w}} [\checkmark] \mid [\![D_{j,k,w}]\!]_{\rho'_{k,w}} \big[h_{t_{j,k,w}} \cdot [\![S_{t_{j,k,w}}]\!]_{\rho''_{k,w}}\big].$$

(2.2.3) $C_2[\checkmark] \equiv [\![L_{c,k,w}]\!]_{\rho'_{k,w}} [O^{(q)}_{u_{c,k,w}}]$:
(2.2.3.1) either $C_2[\checkmark] \equiv [\![L'_{c,k,w}]\!]_{\rho'_{k,w}} [\checkmark] \mid [\![L''_{c,k,w}]\!]_{\rho'_{k,w}} [O^{(q)}_{u_{c,k,w}}]$
(2.2.3.2) or, there are $C_4[\bullet]$ and $c \in \{1, \dots, m_k\}$ such that
$C_4[\checkmark] = O^{(q)}_{u_{c,k,w}}([\![F_{c,k,w}]\!]_{\rho''}[h_{u_{c,k,w}} \cdot [\![P_{u_{c,k,w}}]\!]_{\rho'''}], [\![Q'_{u_{c,k,w}}]\!]_{\varepsilon})$.
Assume that $\mathtt{nl}(u_{c,k,w}, \rho''', [\![F_{c,k,w}]\!]_{\rho''}[h_{u_{c,k,w}} \cdot [\![P_{u_{c,k,w}}]\!]_{\rho'''}])) = 0$ and $q \in \{1, 2, 3, 4\}$ (other cases are similar). By Lemma B.15,
(2.2.3.2.1) either $[\![F_{c,k,w}]\!]_{\rho''}[h_{u_{c,k,w}} \cdot [\![P_{u_{c,k,w}}]\!]_{\rho'''}] = C_5[\checkmark]$
(2.2.3.2.2) or, $[\![Q'_{u_{c,k,w}}]\!]_{\varepsilon}) = C_5[\checkmark]$
for some $C_5[\bullet]$.

In all the cases listed above, it follows directly by Lemma B.10 that $P \Downarrow$ since

$$P \equiv \prod_{w=1}^{z} E_w \Big[\prod_{k=1}^{s_w} G_{k,w} \Big[\prod_{i=1}^{l_k} C_{i,k,w} \big[t_{i,k,w}[P_{t_{i,k,w}}, Q_{t_{i,k,w}}]\big] \mid \prod_{j=1}^{r_k} D_{j,k,w} \big[\overline{t_{j,k,w}} . S_{t_{j,k,w}}\big] \mid \prod_{c=1}^{m_k} L_{c,k,w} \big[u_{c,k,w}[F_{c,k,w}[\overline{u_{c,k,w}} . P_{u_{c,k,w}}], Q_{u_{c,k,w}}]\big]\Big]\Big]. \tag{B.31}$$

Other cases are similar. □

Appendix C. Results related to encoding of $\mathcal{C}$ into $\mathcal{O}$

C.1. Proof of operational correspondence results: Theorem 6.5

In this section we shall prove that operational correspondence (completeness and soundness) holds for the translation $[\![\cdot]\!]^{\circ}_{\rho}$. Most of the lemmas, definitions, and theorems we have introduced to prove the operational correspondence for the

(p)	$I_t^{(p)}(\llbracket P \rrbracket^{\circ}_{t,\rho}, \llbracket Q \rrbracket^{\circ}_{\varepsilon})$ for $\texttt{nl}(p_{t,\rho}, \llbracket P \rrbracket^{\circ}_{t,\rho}) = 0$
(1)	$t\big[\llbracket P \rrbracket^{\circ}_{t,\rho}\big] \mid \texttt{extr}\{t, p_{t,\rho}, p_\rho\} \mid p_\rho[\llbracket Q \rrbracket^{\circ}_{\varepsilon}] \equiv t\big[\llbracket P \rrbracket^{\circ}_{t,\rho}\big] \mid t\{(Y).t[Y] \mid \mathsf{ch}(t, Y) \mid t\{\dagger\}.\overline{h_t}\} \mid p_\rho[\llbracket Q \rrbracket^{\circ}_{\varepsilon}]$
(2)	$t\big[\llbracket P \rrbracket^{\circ}_{t,\rho}\big] \mid t\{\dagger\}.\overline{h_t} \mid p_\rho[\llbracket Q \rrbracket^{\circ}_{\varepsilon}]$
(3)	$\overline{h_t} \mid p_\rho[\llbracket Q \rrbracket^{\circ}_{\varepsilon}]$

(p)	$I_t^{(p)}(\llbracket P \rrbracket^{\circ}_{t,\rho}, \llbracket Q \rrbracket^{\circ}_{\varepsilon})$ for $\texttt{nl}(p_{t,\rho}, \llbracket P \rrbracket^{\circ}_{t,\rho}) > 0$
(1)	$t\big[\llbracket P \rrbracket^{\circ}_{t,\rho}\big] \mid \texttt{extr}\{t, p_{t,\rho}, p_\rho\} \mid p_\rho[\llbracket Q \rrbracket^{\circ}_{\varepsilon}]$ $\equiv t\big[\llbracket P \rrbracket^{\circ}_{t,\rho}\big] \mid t\{(Y).t[Y] \mid \mathsf{ch}(t, Y) \mid \texttt{out}^{\circ}(t, p_{t,\rho}\,,\ p_\rho\,,\ \texttt{nl}(p_{t,\rho}, Y)\,,\ t\{\dagger\}.\overline{h_t})\} \mid p_\rho[\llbracket Q \rrbracket^{\circ}_{\varepsilon}]$
(2)	$t\big[\llbracket P \rrbracket^{\circ}_{t,\rho}\big] \mid p_{t,\rho}\{(X_1, \ldots, X_m).z_t\{(Z).\big(\prod_{k=1}^{m} p_\rho[X_k] \mid t\{\dagger\}.\overline{h_t}\big)\}\}.z_t[\mathbf{0}] \mid p_\rho[\llbracket Q_t \rrbracket^{\circ}_{\varepsilon}]$
$(j+2)$ $1 \le j \le m-1$	$t\big[\llbracket P \rrbracket^{\circ}_{t,\rho} \mid p_{t,\rho}\{(X_1, \ldots, X_{m-j}).z_t\{(Z).\big(\prod_{k=1}^{m-j} p_\rho[X_k] \mid t\{\dagger\}.\overline{h_t}\big) \mid \prod_{k=1}^{j} p_\rho[\llbracket P'_k \rrbracket^{\circ}_{\varepsilon}]\}\}\big] \mid z_t[\mathbf{0}] \mid p_\rho[\llbracket Q_t \rrbracket^{\circ}_{\varepsilon}]$
$(m+2)$	$t\big[\llbracket P' \rrbracket^{\circ}_{t,\rho} \mid z_t\{(Z).\prod_{k=1}^{m} p_\rho[\llbracket P'_k \rrbracket^{\circ}_{\varepsilon}] \mid t\{\dagger\}.\overline{h_t}\}\big] \mid z_t[\mathbf{0}] \mid p_\rho[\llbracket Q \rrbracket^{\circ}_{\varepsilon}]$
$(m+3)$	$t\big[\llbracket P' \rrbracket^{\circ}_{t,\rho}\big] \mid \prod_{k=1}^{m} p_\rho[\llbracket P'_k \rrbracket^{\circ}_{\varepsilon}] \mid t\{\dagger\}.\overline{h_t} \mid p_\rho[\llbracket Q \rrbracket^{\circ}_{\varepsilon}]$
$(m+4)$	$\prod_{k=1}^{m} p_\rho[\llbracket P'_k \rrbracket^{\circ}_{\varepsilon}] \mid \overline{h_t} \mid p_\rho[\llbracket Q \rrbracket^{\circ}_{\varepsilon}]$

Fig. C.22. Process $I_t^{(p)}(\llbracket P \rrbracket^{\circ}_{t,\rho}, \llbracket Q \rrbracket^{\circ}_{\varepsilon})$ with $p \ge 1$.

translation with subjective update (Theorem 5.8), can be easily adapted for the translation with objective update. Therefore, we will re-use the following statements for $\llbracket\cdot\rrbracket^{\circ}_{\rho}$, assuming the expected modifications:

- Definition B.1 (page 43) and Lemma B.1 (page 43), that are related with a mapping of evaluation contexts for $\mathcal{C}$ into evaluation contexts of $\mathcal{S}$;
- Lemma B.2 (page 44) and Corollary B.3 (page 44), are the converse of Lemma B.1,
- Lemma B.6 (page 45), shows that $\mathsf{ch}(t, \llbracket P \rrbracket^{\circ}_{\rho}) = \mathbf{0}$ for all $\llbracket P \rrbracket^{\circ}_{\rho}$ and use Lemma B.5 (page 45), for the proof.
- Lemma B.7 (page 45), identifies processes that are created before a synchronization on h_t.

We first present an overview of the auxiliary results (and proofs) that are different from those presented in B.3. The following definition formalizes all possible forms for the process $I_t^{(p)}(\llbracket P \rrbracket^{\circ}_{t,\rho}, \llbracket Q \rrbracket^{\circ}_{\varepsilon})$.

Definition C.1. Let P, Q be well-formed compensable processes. Given a name t, a path ρ, and $p \ge 1$, we define the intermediate processes $I_t^{(p)}(\llbracket P \rrbracket^{\circ}_{t,\rho}, \llbracket Q \rrbracket^{\circ}_{\varepsilon})$ (Fig. C.22) depending on $m = \texttt{nl}(p_{t,\rho}, \llbracket P \rrbracket^{\circ}_{t,\rho})$:

1. if $m = 0$ then $p \in \{1, 2, 3\}$;
2. otherwise, if $m > 0$ then $\llbracket P \rrbracket^{\circ}_{t,\rho} = \prod_{k=1}^{m} p_{t,\rho}[\llbracket P'_k \rrbracket^{\circ}_{\varepsilon}] \mid S$ and $p \in \{1, \ldots, m+4\}$.

The following lemma formalizes all possible forms for the process $O_u^{(q)}(\llbracket F \rrbracket^{\circ}_{\rho''}[h_u.\llbracket P_u \rrbracket^{\circ}_{\rho'''}], \llbracket Q'_u \rrbracket_{\varepsilon})$.

Definition C.2. Let P, Q be well-formed compensable processes. Given a name u, paths ρ, ρ', and $q \ge 1$, we define the intermediate processes $O_u^{(q)}(\llbracket F \rrbracket^{\circ}_{\rho}[h_u.\llbracket P \rrbracket^{\circ}_{\rho'}], \llbracket Q \rrbracket^{\circ}_{\varepsilon})$ (Fig. C.23) depending on $m = \texttt{nl}(p_{u,\rho}, \llbracket F \rrbracket^{\circ}_{\rho}[h_u.\llbracket P \rrbracket^{\circ}_{\rho'}])$:

1. for $m = 0$ we have $q \in \{1, 2, 3, 4\}$, and
2. for $m > 0$ and $\llbracket F \rrbracket^{\circ}_{\rho}[h_u.\llbracket P \rrbracket^{\circ}_{\rho'}] = \prod_{k=1}^{m} p_{u,\rho}[\llbracket P'_k \rrbracket^{\circ}_{\varepsilon}] \mid S$ we have $q \in \{1, \ldots, m+5\}$.

The following lemmas, which we established for the translation with subjective update $\llbracket\cdot\rrbracket_{\rho}$, hold also for translation with objective update $\llbracket\cdot\rrbracket^{\circ}_{\rho}$; the difference is that they use Definition C.1 and Definition C.2 instead of Definition B.2 and Definition B.3, respectively:

(q)	$O_u^{(q)}(\llbracket F\rrbracket^{\circ}_{\rho}[h_u.\llbracket P\rrbracket^{\circ}_{\rho'}], \llbracket Q\rrbracket^{\circ}_{\varepsilon}),\ \mathtt{nl}(p_{u,\rho}, \llbracket F\rrbracket^{\circ}_{\rho}[h_u.\llbracket P\rrbracket^{\circ}_{\rho'}]) = 0$

(1)	$u\big[\llbracket F\rrbracket^{\circ}_{\rho}[h_u.\llbracket P\rrbracket^{\circ}_{\rho'}]\big] \mid \mathtt{extr}\{u, p_{u,\rho}, p_{\rho}\} \mid p_{\rho}[\llbracket Q\rrbracket^{\circ}_{\varepsilon}]$ $\equiv u\big[\llbracket F\rrbracket^{\circ}_{\rho}[h_u.\llbracket P\rrbracket^{\circ}_{\rho'}]\big] \mid u\{(Y).u[Y] \mid \mathsf{ch}(u, Y) \mid u\{\dagger\}.\overline{h_u}\} \mid p_{\rho}[\llbracket Q\rrbracket^{\circ}_{\varepsilon}]$
(2)	$u\big[\llbracket F\rrbracket^{\circ}_{\rho}[h_u.\llbracket P\rrbracket^{\circ}_{\rho'}]\big] \mid h_u \mid u\{\dagger\}.\overline{h_u} \mid p_{\rho}[\llbracket Q\rrbracket^{\circ}_{\varepsilon}]$
(3)	$h_u \mid \overline{h_u} \mid p_{\rho}[\llbracket Q\rrbracket^{\circ}_{\varepsilon}]$
(4)	$p_{\rho}[\llbracket Q\rrbracket^{\circ}_{\varepsilon}]$

(q)	$O_u^{(q)}(\llbracket F\rrbracket^{\circ}_{\rho}[h_u.\llbracket P\rrbracket^{\circ}_{\rho'}], \llbracket Q\rrbracket^{\circ}_{\varepsilon}),\ \mathtt{nl}(p_{u,\rho}, \llbracket F\rrbracket^{\circ}_{\rho}[h_u.\llbracket P\rrbracket^{\circ}_{\rho'}]) > 0$

(1)	$u\big[\llbracket F\rrbracket^{\circ}_{\rho}[h_u.\llbracket P\rrbracket^{\circ}_{\rho'}]\big] \mid \mathtt{extr}\{u, p_{u,\rho}, p_{\rho}\} \mid p_{\rho}[\llbracket Q\rrbracket^{\circ}_{\varepsilon}]$ $\equiv u\big[\llbracket F\rrbracket^{\circ}_{\rho}[h_u.\llbracket P\rrbracket^{\circ}_{\rho'}]\big] \mid u\{(Y).u[Y] \mid \mathsf{ch}(u, Y) \mid \mathtt{out}^{\circ}(u, p_{u,\rho}\ ,\ p_{\rho}\ ,\ \mathtt{nl}(p_{u,\rho}, Y)\ ,\ u\{\dagger\}.\overline{h_u})\} \mid p_{\rho}[\llbracket Q\rrbracket^{\circ}_{\varepsilon}]$
(2)	$u\big[\llbracket F\rrbracket^{\circ}_{\rho}[h_u.\llbracket P\rrbracket^{\circ}_{\rho'}]\big] \mid h_u \mid p_{u,\rho}\{(X_1, \ldots, X_m).z_u\{(Z).\big(\prod_{k=1}^{m} p_{\rho}[X_k] \mid u\langle\langle\dagger\rangle\rangle.\overline{h_u}\big)\}\}.z_u[\mathbf{0}] \mid p_{\rho}[\llbracket Q\rrbracket^{\circ}_{\varepsilon}]$
$(j+2)$ $1 \le j \le m-1$	$u\big[\llbracket F\rrbracket^{\circ}_{\rho}[h_u.\llbracket P\rrbracket^{\circ}_{\rho'}] \mid p_{u,\rho}\{(X_1, \ldots, X_{m-j}).z_u\{(Z).\big(\prod_{k=1}^{m-j} p_{\rho}[X_k] \mid u\langle\langle\dagger\rangle\rangle.\overline{h_u}\big) \mid \prod_{k=1}^{j} p_{\rho}[\llbracket P'_k\rrbracket^{\circ}_{\varepsilon}]\}\}\big]$ $\mid h_u \mid z_u[\mathbf{0}] \ \mid p_{\rho}[\llbracket Q\rrbracket^{\circ}_{\varepsilon}]$
$(m+2)$	$u\big[\llbracket F\rrbracket^{\circ}_{\rho}[h_u.\llbracket P\rrbracket^{\circ}_{\rho'}] \mid z_u\{(Z).\big(\prod_{k=1}^{m-j} p_{\rho}[X_k] \mid u\langle\langle\dagger\rangle\rangle.\overline{h_u}\big)\}\big] \mid h_u \mid z_u[\mathbf{0}] \mid \prod_{k=1}^{j} p_{\rho}[\llbracket P'_k\rrbracket^{\circ}_{\varepsilon}] \mid p_{\rho}[\llbracket Q\rrbracket^{\circ}_{\varepsilon}]$
$(m+3)$	$u\big[\llbracket F'\rrbracket^{\circ}_{\rho}[h_u.\llbracket P\rrbracket^{\circ}_{\rho'}]\big] \mid \prod_{k=1}^{m} p_{\rho}[\llbracket P'_k\rrbracket^{\circ}_{\varepsilon}] \mid h_u \mid p_{\rho}[\llbracket Q\rrbracket^{\circ}_{\varepsilon}] \mid u\{\dagger\}.\overline{h_u}$
$(m+4)$	$\prod_{k=1}^{m} p_{\rho}[\llbracket P'_k\rrbracket^{\circ}_{\varepsilon}] \mid h_u \mid p_{\rho}[\llbracket Q\rrbracket^{\circ}_{\varepsilon}] \mid \overline{h_u}$
$(m+5)$	$\prod_{k=1}^{m} p_{\rho}[\llbracket P'_k\rrbracket^{\circ}_{\varepsilon}] \mid p_{\rho}[\llbracket Q\rrbracket^{\circ}_{\varepsilon}]$

Fig. C.23. Process $O_u^{(q)}(\llbracket F\rrbracket^{\circ}_{\rho}[h_u.\llbracket P\rrbracket^{\circ}_{\rho'}], \llbracket Q\rrbracket^{\circ}_{\varepsilon})$ with $q \ge 1$.

- Lemma B.10 (page 50), is about the shape of process R in $\llbracket P\rrbracket_{\varepsilon} \longrightarrow^{n} R$, and also ensures that there is a process P' with an appropriate shape. The proof proceeds by induction on n.
- Lemma B.4 (page 44), is used as the base case in the proof of Lemma B.10;
- Lemma B.8 (page 46) and Lemma B.9 (page 49) are used in the inductive step of the proof of Lemma B.10.
- Lemma B.11 (page 54), ensures that the adaptable process obtained thanks to Lemma B.8 and Lemma B.9 can evolve until reaching a process that corresponds to the translation of a compensable process.

C.1.1. Operational correspondence

In the following we repeat statement at page 23.

Theorem 6.5 *(Operational correspondence for $\llbracket\cdot\rrbracket^{\circ}_{\varepsilon}$). Let P be a well-formed process in $\mathcal{C}$.*

(1) *If $P \to P'$ then $\llbracket P\rrbracket^{\circ}_{\varepsilon} \longrightarrow^{k} \llbracket P'\rrbracket^{\circ}_{\varepsilon}$ where for*
  a) *$P \equiv E[C[\overline{a}.P_1] \mid D[a.P_2]]$ and $P' \equiv E[C[P_1] \mid D[P_2]]$ it follows that $k = 1$.*
  b) *$P \equiv E[C[t[P_1, Q]] \mid D[\overline{t}.P_2]]$ and $P' \equiv E[C[\mathrm{extr}(P_1) \mid \langle Q\rangle] \mid D[P_2]]$ it follows $k = 4 + \mathrm{pb}(P_1) + \mathrm{z}(P_1)$,*
  c) *$P \equiv C[u[D[\overline{u}.P_1], Q]]$ and $P' \equiv C[\mathrm{extr}(D[P_1]) \mid \langle Q\rangle]$, it follows $k = 4 + \mathrm{pb}(D[P_1]) + \mathrm{z}(D[P_1])$.*

  for some contexts C, D, E, processes P_1, Q, P_2 and names t, u.

(2) *If $\llbracket P\rrbracket^{\circ}_{\varepsilon} \longrightarrow^{n} R$ with $n > 0$ then there is P' such that $P \longrightarrow^{*} P'$ and $R \longrightarrow^{*} \llbracket P'\rrbracket^{\circ}_{\varepsilon}$.*

Proof. We consider completeness and soundness (Parts (1) and (2)) separately.

(1) **Part (1) – Completeness:** The proof proceeds by induction on the derivation of $P \longrightarrow P'$. We have three base cases, corresponding to cases $a), b)$ and $c)$ of Proposition 3.1 (page 7). Also, we prove all cases by using Definition 6.2 and Lemma B.1.
  a) This case corresponds to an input-output synchronization, such that $a \in \mathcal{N}_s$. Therefore, we observe that $P \equiv E[C[\overline{a}.P_1] \mid D[a.P_2]]$ and $P' \equiv E[C[P_1] \mid D[P_2]]$. The derivation that corresponds to this case is as the derivation

presented in **Part (1) – Completeness** case (a) for translation with subjective update (cf. derivation (B.26)). Therefore, the thesis holds with $k = 1$.

b) This case corresponds to a synchronization due to an external error notification for a transaction scope. Therefore, for this case we consider that $P \equiv E[C[t[P_1, Q]] \mid D[\bar{t}.P_2]]$ and $P' \equiv E[C[\mathsf{extr}(P_1) \mid \langle Q \rangle] \mid D[P_2]]$. We will consider two sub-cases depending on whether process P_1 contains or not protected blocks. Below, we will use that $m = \mathtt{pb}(P_1)$.

(i) In this sub-case $m = 0$. Therefore, we have the following derivation:

$$\begin{aligned}
[\![P]\!]^{\circ}_{\varepsilon} &\equiv [\![E[C[t[P_1, Q]] \mid D[\bar{t}.P_2]]]\!]^{\circ}_{\varepsilon} \\
&= [\![E]\!]^{\circ}_{\varepsilon}\Big[[\![C[t[P_1, Q]]]\!]^{\circ}_{\rho} \mid [\![D[\bar{t}.P_2]]\!]^{\circ}_{\rho}\Big] \\
&= [\![E]\!]^{\circ}_{\varepsilon}\Big[[\![C]\!]^{\circ}_{\rho}[[\![t[P_1, Q]]\!]^{\circ}_{\rho'}] \mid [\![D]\!]^{\circ}_{\rho}[[\![\bar{t}.P_2]\!]^{\circ}_{\rho''}]\Big] \\
&= [\![E]\!]^{\circ}_{\varepsilon}\Big[[\![C]\!]^{\circ}_{\rho}[t\big[[\![P_1]\!]^{\circ}_{t,\rho'}\big] \mid t.\big(\mathtt{extr}\{t, p_{t,\rho'}, p_{\rho'}\} \mid p_{\rho'}[[\![Q]\!]^{\circ}_{\varepsilon}]\big)] \mid [\![D]\!]^{\circ}_{\rho}[\bar{t}.h_t.[\![P_2]\!]^{\circ}_{\rho''}]\Big] \\
\longrightarrow\ & [\![E]\!]^{\circ}_{\varepsilon}\Big[[\![C]\!]^{\circ}_{\rho}\Big[I^{(1)}_t([\![P_1]\!]^{\circ}_{t,\rho'}, [\![Q]\!]^{\circ}_{\varepsilon})\Big] \mid [\![D]\!]^{\circ}_{\rho}[h_t.[\![P_2]\!]^{\circ}_{\rho''}]\Big] \\
\longrightarrow^2\ & [\![E]\!]^{\circ}_{\varepsilon}\Big[[\![C]\!]^{\circ}_{\rho}\Big[I^{(3)}_t([\![P_1]\!]^{\circ}_{t,\rho'}, [\![Q]\!]^{\circ}_{\varepsilon})\Big] \mid [\![D]\!]^{\circ}_{\rho}\big[h_t.[\![P_2]\!]^{\circ}_{\rho''}\big]\Big] \\
\longrightarrow\ & [\![E]\!]^{\circ}_{\varepsilon}\Big[[\![C]\!]^{\circ}_{\rho}[[\![\langle Q \rangle]\!]^{\circ}_{\rho'}] \mid [\![D]\!]^{\circ}_{\rho}\big[[\![P_2]\!]^{\circ}_{\rho''}\big]\Big] \\
&= [\![E]\!]^{\circ}_{\varepsilon}\big[[\![C[\langle Q \rangle]]\!]^{\circ}_{\rho} \mid [\![D[P_2]]\!]^{\circ}_{\rho}\big] \\
&= [\![E[C[\langle Q \rangle] \mid D[P_2]]]\!]^{\circ}_{\varepsilon} \\
&\equiv [\![P']\!]^{\circ}_{\varepsilon}
\end{aligned}$$

Thus, the number of reduction steps is $k = 4$. Notice that here $\longrightarrow^2$ tells us that there have been two reduction steps: the first one is an update on location name t; the second reduction step "kills" with $t\{\dagger\}$ both the location t and the process it hosts.

(ii) In this sub-case we consider $m > 0$, i.e., this is when there is at least one protected block in the default activity P_1. We have the following derivation:

$$\begin{aligned}
[\![P]\!]^{\circ}_{\varepsilon} \quad &\equiv \quad [\![E[C[t[P_1, Q]] \mid D[\bar{t}.P_2]]]\!]^{\circ}_{\varepsilon} \\
&= \quad [\![E]\!]^{\circ}_{\varepsilon}\Big[[\![C[t[P_1, Q]]]\!]^{\circ}_{\rho} \mid [\![D[\bar{t}.P_2]]\!]^{\circ}_{\rho}\Big] \\
&= \quad [\![E]\!]^{\circ}_{\varepsilon}\Big[[\![C]\!]^{\circ}_{\rho}[[\![t[P_1, Q]]\!]^{\circ}_{\rho'}] \mid [\![D]\!]^{\circ}_{\rho}[[\![\bar{t}.P_2]\!]^{\circ}_{\rho''}]\Big] \\
&= \quad [\![E]\!]^{\circ}_{\varepsilon}\Big[[\![C]\!]^{\circ}_{\rho}[t\big[[\![P_1]\!]^{\circ}_{t,\rho'}\big] \mid t.\big(\mathtt{extr}\{t, p_{t,\rho'}, p_{\rho'}\} \mid p_{\rho'}[[\![Q]\!]^{\circ}_{\varepsilon}]\big)] \mid [\![D]\!]^{\circ}_{\rho}[\bar{t}.h_t.[\![P_2]\!]^{\circ}_{\rho''}]\Big] \\
&\longrightarrow \quad [\![E]\!]^{\circ}_{\varepsilon}\Big[[\![C]\!]^{\circ}_{\rho}[I^{(1)}_t([\![P_1]\!]^{\circ}_{t,\rho'}, [\![Q]\!]^{\circ}_{\varepsilon})] \mid [\![D]\!]^{\circ}_{\rho}[h_t.[\![P_2]\!]^{\circ}_{\rho''}]\Big] \\
&\longrightarrow^{m+3} \quad [\![E]\!]^{\circ}_{\varepsilon}\Big[[\![C]\!]^{\circ}_{\rho}\Big[I^{(m+4)}_t([\![P_1]\!]^{\circ}_{t,\rho'}, [\![Q]\!]^{\circ}_{\varepsilon})\Big] \mid [\![D]\!]^{\circ}_{\rho}\big[h_t.[\![P_2]\!]^{\circ}_{\rho''}\big]\Big] \\
&\longrightarrow \quad [\![E]\!]^{\circ}_{\varepsilon}\Big[[\![C]\!]^{\circ}_{\rho}\Big[[\![\mathsf{extr}(P_1)]\!]^{\circ}_{\rho'} \mid p_{\rho'}[[\![Q]\!]^{\circ}_{\varepsilon}]\Big] \mid [\![D]\!]^{\circ}_{\rho}\big[[\![P_2]\!]^{\circ}_{\rho''}\big]\Big] \\
&= \quad [\![E]\!]^{\circ}_{\varepsilon}\big[[\![C[\mathsf{extr}(P_1) \mid \langle Q \rangle]]\!]^{\circ}_{\rho} \mid [\![D[P_2]]\!]^{\circ}_{\rho}\big] \\
&= \quad [\![E[C[\mathsf{extr}(P_1) \mid \langle Q \rangle] \mid D[P_2]]]\!]^{\circ}_{\varepsilon} \\
&\equiv \quad [\![P']\!]^{\circ}_{\varepsilon}
\end{aligned}$$

Therefore, $k = 4 + m + \mathtt{z}(P_1) = 5 + m$, where:

- 4 steps are as described in Section B.3 and under a semantics with objective update, after m updates, processes located at $p_{t,\rho'}$ will stay at location t, and
- $\mathtt{z}(P_1)$ gives 1 more step; as we explained in the main part of the paper, to avoid leaving such processes in the wrong location, the translation in [9] use an (objective) update on auxiliary location z_t, so to take them out of t once m updates on $p_{t,\rho'}$ have been executed. This additional synchronization step on name z_t is the key to the efficiency gains when moving from objective to subjective updates (cf. Definition 6.3, page 23).

c) In this case we consider that error notification arrives from the default activity of transaction; the error notification is internal. Again, according to Proposition 3.1 we consider the following case. Let $P \equiv C[u[D[\bar{u}.P_1], Q]]$ and $P' \equiv C[\mathsf{extr}(D[P_1]) \mid \langle Q \rangle]$. Letting $m = \mathtt{pb}(D[P_1])$, we consider two sub-cases: the first case is when $m = 0$ and the second is when $m > 0$:

(i) If $m = 0$ then there is the following derivation:

$$\begin{array}{rcl}
[\![P]\!]^\circ_\varepsilon & \equiv & [\![C[u[D[\overline{u}.P_1], Q]]\!]^\circ_\varepsilon \\
& = & [\![C]\!]^\circ_\varepsilon[[\![u[D[\overline{u}.P_1], Q]]\!]^\circ_\rho] \\
& = & [\![C]\!]^\circ_\varepsilon[u\big[[\![D[\overline{u}.P_1]]\!]^\circ_{u,\rho}\big] \mid u.\big(\mathtt{extr}\{u, p_{u,\rho}, p_\rho\} \mid p_\rho[[\![Q]\!]^\circ_\varepsilon]\big)]] \\
& = & [\![C]\!]^\circ_\varepsilon[u\big[[\![D]\!]^\circ_{u,\rho}[\overline{u}.h_u.[\![P_1]\!]^\circ_{\rho'}]\big] \mid u.(\mathtt{extr}\{u, p_{u,\rho}, p_\rho\} \mid p_\rho[[\![Q]\!]^\circ_\varepsilon])] \\
& \longrightarrow & [\![C]\!]^\circ_\varepsilon\big[O^{(1)}_u([\![D]\!]^\circ_\rho[h_u.[\![P_1]\!]^\circ_{\rho'}], [\![Q]\!]^\circ_\varepsilon)\big] \\
& \longrightarrow^3 & [\![C]\!]^\circ_\varepsilon\big[O^{(4)}_u([\![D]\!]^\circ_\rho[h_u.[\![P_1]\!]^\circ_{\rho'}], [\![Q]\!]^\circ_\varepsilon)\big] \\
& \equiv & [\![P']\!]^\circ_\varepsilon
\end{array}$$

Therefore, the number of reduction steps is $k = 4$.

(ii) If $m > 0$ then there is the following derivation:

$$\begin{array}{rcl}
[\![P]\!]^\circ_\varepsilon & \equiv & [\![C[u[D[\overline{u}.P_1], Q]]\!]^\circ_\varepsilon \\
& = & [\![C]\!]^\circ_\varepsilon[[\![u[D[\overline{u}.P_1], Q]]\!]^\circ_\rho] \\
& = & [\![C]\!]^\circ_\varepsilon[u\big[[\![D[\overline{u}.P_1]]\!]^\circ_{u,\rho}\big] \mid u.\big(\mathtt{extr}\{u, p_{u,\rho}, p_\rho\} \mid p_\rho[[\![Q]\!]^\circ_\varepsilon]\big)]] \\
& = & [\![C]\!]^\circ_\varepsilon[u\big[[\![D]\!]^\circ_{u,\rho}[\overline{u}.h_u.[\![P_1]\!]^\circ_{\rho'}]\big] \mid u.(\mathtt{extr}\{u, p_{u,\rho}, p_\rho\} \mid p_\rho[[\![Q]\!]^\circ_\varepsilon])] \\
& \longrightarrow & [\![C]\!]^\circ_\varepsilon\Big[O^{(1)}_u([\![D]\!]^\circ_\rho[h_u.[\![P_1]\!]^\circ_{\rho'}], [\![Q]\!]^\circ_\varepsilon)\Big] \\
& \longrightarrow^{m+3} & [\![C]\!]^\circ_\varepsilon\Big[O^{(m+4)}_u([\![D]\!]^\circ_\rho[h_u.[\![P_1]\!]^\circ_{\rho'}], [\![Q]\!]^\circ_\varepsilon)\Big] \\
& \longrightarrow & [\![C]\!]^\circ_\varepsilon\Big[O^{(m+5)}_u([\![D]\!]^\circ_\rho[h_u.[\![P_1]\!]^\circ_{\rho'}], [\![Q]\!]^\circ_\varepsilon)\Big] \\
& = & [\![C]\!]^\circ_\varepsilon\Big[[\![\mathsf{extr}(D[P_1])]\!]^\circ_\rho \mid p_\rho[[\![Q]\!]^\circ_\varepsilon]\Big] \\
& = & [\![C[\mathsf{extr}(D[P_1]) \mid \langle Q\rangle]\!]^\circ_\varepsilon \\
& \equiv & [\![P']\!]^\circ_\varepsilon
\end{array}$$

Therefore, the number of reduction steps is $k = 4 + m + \mathtt{Z}(D[P_1]) = 5 + m$.

(2) **Part (2) – Soundness:** The proof for soundness follows the approach described in detail for encoding with subjective update (cf. proof for soundness B.20). Therefore, given $[\![P]\!]^\circ_\varepsilon \longrightarrow^n R$, by Lemma B.10 (which also applies to $[\![\cdot]\!]^\circ_\rho$), process R has the following form:

$$R \equiv \prod_{w=1}^{z}[\![E_w]\!]^\circ_\varepsilon\Big[\prod_{k=1}^{s_w}[\![G_{k,w}]\!]^\circ_{\rho_w}\big[\prod_{i=1}^{l_k}[\![C_{i,k,w}]\!]^\circ_{\rho'_{k,w}}\big[I^{(p)}_{t_{i,k,w}}\big]\mid\prod_{j=1}^{r_k}[\![D_{j,k,w}]\!]^\circ_{\rho'_{k,w}}\big[h_{t_{j,k,w}}.[\![S_{t_{j,k,w}}]\!]^\circ_{\rho''_{k,w}}\big]$$
$$\mid\prod_{c=1}^{m_k}[\![L_{c,k,w}]\!]^\circ_{\rho'_{k,w}}\big[O^{(q)}_{u_{c,k,w}}\big]\big]\Big],$$

where $I^{(p)}_{t_{i,k,w}}$ and $O^{(q)}_{u_{c,k,w}}$ are processes from Fig. C.22 and Fig. C.23, respectively.
By Lemma B.10, we have $P \longrightarrow^* P''$ such that

$$P'' \equiv \prod_{w=1}^{z}E_w\Big[\prod_{k=1}^{s_w}G_{k,w}\big[\prod_{i=1}^{l_k}C_{i,k,w}\big[t_{i,k,w}[P_{t_{i,k,w}}, Q_{t_{i,k,w}}]\big]\mid\prod_{j=1}^{r_k}D_{j,k,w}\big[\overline{t_{j,k,w}}.S_{t_{j,k,w}}\big]$$
$$\mid\prod_{c=1}^{m_k}L_{c,k,w}\big[u_{c,k,w}[F_{c,k,w}[\overline{u_{c,k,w}}.P_{u_{c,k,w}}], Q_{u_{c,k,w}}]\big]\big]\Big],$$

where by successive application of completeness it follows that $[\![P]\!]^\circ_\varepsilon \longrightarrow^* [\![P'']\!]^\circ_\varepsilon$.
By Lemma B.11 (which also applies to $[\![\cdot]\!]^\circ_\rho$), i.e., by l_k successive applications of (B.24) and m_k successive applications of (B.25) on process R, it follows that

$$R \longrightarrow^* \prod_{w=1}^{z}[\![E_w]\!]^\circ_\varepsilon\Big[\prod_{k=1}^{s_w}[\![G_{k,w}]\!]^\circ_{\rho_w}\big[\prod_{i=1}^{l_k}[\![C_{i,k,w}]\!]^\circ_{\rho'_{k,w}}\big[[\![\mathsf{extr}(P'_{t_{i,k,w}})]\!]^\circ_{\rho''_{k,w}} \mid [\![\langle Q'_{t_{i,k,w}}\rangle]\!]^\circ_{\rho''_{k,w}}\big]$$
$$\mid\prod_{j=1}^{r_k}[\![D_{j,k,w}]\!]^\circ_{\rho'_{k,w}}\big[[\![S_{t_{j,k,w}}]\!]^\circ_{\rho''_{k,w}}\big]\mid\prod_{c=1}^{m_k}[\![L_{c,k,w}]\!]^\circ_{\rho'_{k,w}}\big[[\![\mathsf{extr}(F_{c,k,w}[P_{u_{c,k,w}}])]\!]^\circ_{\rho''_{k,w}} \mid [\![\langle Q'_{u_{c,k,w}}\rangle]\!]^\circ_{\rho''_{k,w}}\big]\big]\Big]$$
$$\equiv [\![\prod_{w=1}^{z}E_w\Big[\prod_{k=1}^{s_w}G_{k,w}\big[\prod_{i=1}^{l_k}C_{i,k,w}\big[\mathsf{extr}(P'_{t_{i,k,w}}) \mid \langle Q'_{t_{i,k,w}}\rangle\big]\mid\prod_{j=1}^{r_k}D_{j,k,w}\big[S_{t_{j,k,w}}\big]$$

$$\mid \prod_{c=1}^{m_k} L_{c,k,w}\Big[\mathsf{extr}(F_{c,k,w}[P_{u_{c,k,w}}]) \mid \langle Q'_{u_{c,k,w}}\rangle\Big]\Big]\Big]\rrbracket^{\circ}_{\varepsilon}$$

$$\equiv \llbracket P' \rrbracket^{\circ}_{\varepsilon}.$$

Therefore, it follows that

$$P' \equiv \prod_{w=1}^{z} E_w\Big[\prod_{k=1}^{s_w} G_{k,w}\Big[\prod_{i=1}^{l_k} C_{i,k,w}\big[\mathsf{extr}(P'_{t_{i,k,w}}) \mid \langle Q'_{t_{i,k,w}}\rangle\big] \mid \prod_{j=1}^{r_k} D_{j,k,w}\big[S_{t_{j,k,w}}\big]$$

$$\mid \prod_{c=1}^{m_k} L_{c,k,w}\big[\mathsf{extr}(F_{c,k,w}[P_{u_{c,k,w}}]) \mid \langle Q'_{u_{c,k,w}}\rangle\big]\Big]\Big].$$

By Proposition 3.1, i.e., by l_k successive applications of case b) and m_k successive applications of case c) on process P'', it follows that $P'' \longrightarrow^* P'$.

By successive application of **(C.1.1) – Completeness** on the derivation $P'' \longrightarrow^* P'$ it follows that $\llbracket P'' \rrbracket^{\circ}_{\varepsilon} \longrightarrow^* \llbracket P' \rrbracket^{\circ}_{\varepsilon}$. □

References

[1] S. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M.G. Ford, Y. Goland, A. Guzar, N. Kartha, C.K. Liu, R. Khalaf, D. König, M. Marin, V. Mehta, S. Thatte, D. Rijn, P. Yendluri, A. Yiu, Web services business process execution language version 2.0 (oasis standard). 2007.
[2] L. Bocchi, C. Laneve, G. Zavattaro, A calculus for long-running transactions, in: Proc. of FMOODS 2003, in: LNCS, vol. 2884, Springer, 2003, pp. 124–138.
[3] M. Bravetti, C.D. Giusto, J.A. Pérez, G. Zavattaro, Adaptable processes, Log. Methods Comput. Sci. 8 (4) (2012).
[4] M. Bravetti, G. Zavattaro, On the expressive power of process interruption and compensation, Math. Struct. Comput. Sci. 19 (3) (2009) 565–599.
[5] M.J. Butler, C. Ferreira, M.Y. Ng, Precise modelling of compensating business transactions and its application to BPEL, J. Univers. Comput. Sci. 11 (5) (2005) 712–743.
[6] L. Caires, C. Ferreira, H.T. Vieira, A process calculus analysis of compensations, in: Proc. of TGC 2008, in: LNCS, vol. 5474, Springer, 2009, pp. 87–103.
[7] L. Cardelli, A.D. Gordon, Mobile ambients, Theor. Comput. Sci. 240 (1) (2000) 177–213.
[8] G. Castagna, J. Vitek, F.Z. Nardelli, The seal calculus, Inf. Comput. 201 (1) (2005) 1–54.
[9] J. Dedeic, J. Pantovic, J.A. Pérez, On compensation primitives as adaptable processes, in: EXPRESS/SOS 2015, in: EPTCS, vol. 190, 2015, pp. 16–30.
[10] J. Dedeić, J. Pantović, J.A. Pérez, Efficient compensation handling via subjective updates, in: Proceedings of the Symposium on Applied Computing, SAC '17, New York, NY, USA, ACM, 2017, pp. 51–58.
[11] C. Ferreira, I. Lanese, A. Ravara, H.T. Vieira, G. Zavattaro, Advanced mechanisms for service combination and transactions, in: Results of SENSORIA, in: LNCS, vol. 6582, Springer, 2011, pp. 302–325.
[12] D. Gorla, Towards a unified approach to encodability and separation results for process calculi, Inf. Comput. 208 (9) (2010) 1031–1053.
[13] T.T. Hildebrandt, J.C. Godskesen, M. Bundgaard, Bisimulation congruences for Homer - a calculus of higher-order mobile embedded resources, Technical Report TR-2004-52, IT University, 2004.
[14] C.A.R. Hoare, Communicating Sequential Processes, Prentice-Hall, 1985.
[15] I. Lanese, C. Vaz, C. Ferreira, On the expressive power of primitives for compensation handling, in: Proc. of ESOP 2010, in: LNCS, vol. 6012, Springer, 2010, pp. 366–386.
[16] I. Lanese, G. Zavattaro, Decidability results for dynamic installation of compensation handlers, in: Coordination, in: LNCS, vol. 7890, Springer, 2013, pp. 136–150.
[17] C. Laneve, G. Zavattaro, Foundations of web transactions, in: Proc. of FOSSACS 2005, in: LNCS, vol. 3441, Springer, 2005, pp. 282–298.
[18] R. Milner, Communication and Concurrency, PHI Series in Computer Science, Prentice-Hall, 1989.
[19] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes, I, Inf. Comput. 100 (1) (1992) 1–40.
[20] J. Parrow, Expressiveness of process algebras, in: Proceedings of the LIX Colloquium on Emerging Trends in Concurrency Theory, LIX 2006, in: Electronic Notes in Theoretical Computer Science, vol. 209, 2008, pp. 173–186.
[21] K. Peters, Comparing process calculi using encodings, in: J.A. Pérez, J. Rot (Eds.), Proceedings Combined 26th International Workshop on Expressiveness in Concurrency and 16th Workshop on Structural Operational Semantics, EXPRESS/SOS 2019, Amsterdam, The Netherlands, 26th August 2019, in: EPTCS, vol. 300, 2019, pp. 19–38.
[22] D. Sangiorgi, Expressing mobility in process algebras: first-order and higher order paradigms, PhD thesis, University of Edinburgh, 1992.
[23] A. Schmitt, J. Stefani, The Kell calculus: a family of higher-order distributed process calculi, in: C. Priami, P. Quaglia (Eds.), Global Computing, IST/FET International Workshop, GC 2004, Rovereto, Italy, March 9–12, 2004, Revised Selected Papers, in: Lecture Notes in Computer Science, vol. 3267, Springer, 2004, pp. 146–178.
[24] C. Vaz, C. Ferreira, On the analysis of compensation correctness, J. Log. Algebraic Program. 81 (5) (2012) 585–605.