Time and Fairness in a Process Algebra with Non-Blocking Reading

F. Corradini, M.R. Di Berardini,
W. Vogler

Report 2008-13 July 2008

Time and Fairness in a Process Algebra with Non-Blocking Reading

F. Corradini, M.R. Di Berardini
Dipartimento di Matematica e Informatica
Università di Camerino
{flavio.corradini,mariarita.diberardini}@unicam.it

W. Vogler
Institut für Informatik
Universität Augsburg
vogler@informatik.uni-augsburg.de

July 23, 2008

Abstract

We introduce the first process algebra with non-blocking reading actions for modelling concurrent asynchronous systems, and we do it in two different ways: one is more flexible, the other is simpler since it needs only one type of transitions. We study the impact this new kind of actions have on fairness, liveness and the timing of systems, using Dekker's mutual exclusion algorithm we already considered in [4] as running example. Regarding some actions as reading, this algorithm satisfies MUTEX liveness already under the assumption of fairness of actions. We demonstrate an interesting correspondence between liveness and the catastrophic cycles that we introduced in [7] when studying the performance of pipelining. Finally, our previous result on the correspondence between timing and fairness [4] scales up to the extended language.

1 Introduction

Read arcs are an extension of classical Petri nets to model non-destructive reading operations; they allow multiple concurrent reading of the same resource, a quite frequent situation in many distributed systems. Read-arcs represent *positive context conditions*, i.e. elements which are needed for an event to occur, but are not affected by it. As argued in [15], the importance of such elements is twofold. Firstly, they allow a faithful representation of systems where the notion of "reading without consuming" is commonly used, like database systems, concurrent constraint programming, or any computation framework based on shared memory. Secondly, they allow to specify directly and naturally a level of concurrency greater than in classical nets: two transitions reading the same place may occur in any order and also simultaneously; in classical nets, the transitions would be connected to the place by loops such that they cannot occur simultaneously. Read arcs have been used to model a variety of applications such as transaction serialisability in databases [18], concurrent constraint programming [16], asynchronous systems [19], and cryptographic protocols [11].

Semantics and expressivity of read arcs have been studied e.g. in the following: [2] discusses a step semantics. [1] discusses the expressiveness of timed Petri nets and timed automata and shows that timed Petri nets with read-arcs unify timed Petri nets and timed automata. Finally, [19] shows that read arcs add relevant expressivity; the MUTEX problem can be solved with nets having read arcs but not with ordinary nets having no read arcs.

In this paper, we introduce the first process algebra with non-blocking reading; we add reading in the form of a read-action prefix to PAFAS [3], a process algebra for modelling timed concurrent asynchronous systems. We present two different ways to enhance our process algebra with such non-blocking actions: one is more flexible, the other is simpler since it needs only one type of transition relation; we provide a translation of the latter into the former in Section 4. Non-blocking actions have a direct impact on the timed behaviour of systems. To show this with a simple example, consider a system composed of two processes that need to read the same variable to prepare an output they produce together. Such a system can be described in PAFAS as $(r.o.\mathsf{nil} \parallel_{\{o\}} r.o.\mathsf{nil}) \parallel_{\{r,w\}} \mathsf{rec}\ x.\,(r.x + w.x)$, where $r.o.\mathsf{nil} \parallel_{\{o\}} r.o.\mathsf{nil}$ models the two processes that read from the same variable and produce the output o together, while $\mathsf{rec}\ x.\,(r.x + w.x)$ models the variable that can repeatedly be read with r or written with w. According to the PAFAS semantics, enabled actions are performed immediately or become urgent after one time unit and must occur then; thus, after at most one time unit, the first r occurs. With the given model of the variable, every time an action r is performed, a new instance of r is generated that can let one time unit pass; thus, a second time unit might pass before the second r, and the output is produced within three time units in the worst-case. If the read action r were modelled as a non-blocking action, written $\mathsf{rec}\ x.\, r \triangleright w.x$, then the worst-case efficiency for producing the output would be two time units. In this case, after one time unit and the occurrence of the first r, the variable offers the same non-blocking action r such that the second r is still urgent and has to be performed before the second time step.

In previous work, we have shown that our notion of time [8] is strongly related to (weak) fairness, which requires that an action has to be performed, a component has to act resp., whenever it is enabled continuously in a run. We have proven that each everlasting (or non-Zeno) timed process execution is fair and vice versa, where fairness is defined in an intuitive but complicated way in the spirit of [10, 9]. In fact, we have shown this correspondence for fairness of actions [4] and with a modified notion of timing for fairness of components [5]. We used these characterisations in [6] to study the liveness property for Dekker's mutual exclusion algorithm, and proved that Dekker is *live under* the assumption of *fairness of components but not under* the assumption of *fairness of actions*.

Here, we show that non-blocking actions preserve the above correspondence result between fairness of actions and timing. Another main result is that, in the new setting, Dekker's algorithm is live when assuming fairness of actions, provided we regard as non-blocking the reading of a variable (as r in $\mathsf{rec}\ x.\,(r.x + w.x)$ above) as well as its writing in the case that the written value equals the current value. This kind of re-write does not change the state of the variable and, hence, can be thought of as a non-destructive or non-consuming operation (allowing potential concurrent behaviour). This way of accessing a variable is not new; in Remark 1.1 below, we describe how it is implemented in the area of databases.

Finally, we develop an interesting connection between liveness of MUTEX algorithms and catastrophic cycles: we considered the latter in [7] studying the very different problem of asymptotic performance for the specific, but often occurring class of request-response processes (having only actions *in* and *out*). A cycle in a transition system is catastrophic if it only contains time steps and internal actions, and we showed that a process can refuse to serve some request within finite time if and only if a reduced transition system of the process contains a catastrophic cycle. We also pointed out that the existence of catastrophic cycles in a reduced transition system can be determined in polynomial time. In the present paper, we show how to modify the process *Dekker* such that *Dekker* satisfies liveness under the assumption of fairness of actions if and only if the modified process does not have a catastrophic cycle. This opens the way to check automatically the liveness property for MUTEX algorithms.

Remark 1.1 Our idea of re-write has been implemented with the so-called *two-phase locking protocol*, that is used for preventing transaction conflicts (i.e. two transactions working on the same data item with at least one of them writing). Such a protocol implements a lock system where each transaction may only access a data item if it holds a lock on that item. There are two possible modes of locks: *shared* and *exclusive.* If a transaction T holds a *shared mode* lock (an S-lock, for short) on data item q, then T may read – but not write – q. On the other hand, a transaction with an *exclusive mode* lock (an X-lock) on q can both read and write it. Multiple S-locks are allowed on a single data item, but only one X-lock can be acquired for it. This allows multiple reads (which do not create serialisability conflicts) as in our modelling of variables, but writing prevents reading or another writing interaction (which would create conflicts).

In the two-phase locking protocol, a transaction can acquire new locks only during the so-called *growing phase*. All the locks acquired in the growing phase can be released only during a subsequent phase, called *shrinking phase*. Furthermore, an S-lock can be upgraded to X during the growing phase and, similarly, an X-lock can be downgraded to S during the shrinking phase. The idea here is that – during the growing phase – a transaction, instead of holding an X-lock on an item that it does not need to write yet, can hold an S-lock until the point where modifications to the old value begin, in order to allow other transactions to read the old value for longer. This can be used for a "reading first" implementation of writing: each write operation first reads the old value (this read requires an S-lock for the variable and can be done concurrently with other read operations) and then only writes a new one if it is really different (in this latter case, the S-lock has to be upgraded to an X-lock). This way, a re-writing of the same value is indeed non-blocking.

The rest of the paper is organised as follows. The next section introduces PAFAS with read-prefixes, and its functional and temporal operational semantics. Section 3 presents an alternative way of introducing non-blocking actions in process algebras. Section 4 contrasts our two alternative approaches of introducing non-blocking actions in process algebras. Section 5 introduces fairness, while Section 6 relates it to timing. Finally, Section 8 investigates the liveness of Dekker's algorithm under the assumption of fairness of actions and presents the interesting connection between liveness and catastrophic cycles.

2 A process algebra for describing read behaviours

In this section we introduce a new process description language – called PAFAS$_r$ – suitable to model non-blocking reading in concurrent asynchronous systems, a feature with an impact on fair and on timed behaviour as already discussed in [19] in a Petri net setting. PAFAS$_r$ is an extension of the timed process algebra PAFAS, introduced in [8] for evaluating the worst-case efficiency of asynchronous systems and used in [4, 5] for studying fairness of actions and components in system computations.

PAFAS is a CCS-like process description language [14] (with a *TCSP*-like parallel composition), where basic actions are atomic and instantaneous but have associated an upper time bound (either 0 or 1, for simplicity) interpreted as a maximal time delay for their execution. As explained in [8], these upper time bounds can be used for evaluating the performance of asynchronous systems, but do not influence *functionality* (which actions are performed); so compared to CCS, also PAFAS treats the full functionality of asynchronous systems. With respect to the original language in [8], here we introduce the new operator $\triangleright$ to represent non-blocking behaviour of processes. Intuitively, the term $\alpha \triangleright P$ models a process like a variable or a more complex data structure that behaves as P but can additionally be read with α: since being read does not change the state, α can be performed repeatedly until the execution of some ordinary action of P, and it does not block a synchronisation partner as described below.

We use the following notation. $\mathbb{A}$ is an infinite set of *basic actions*. An additional action τ is used to represent internal activity, which is unobservable for other components. We define $\mathbb{A}_\tau = \mathbb{A} \cup \{\tau\}$. Elements of $\mathbb{A}$ are denoted by $a, b, c, \ldots$ and those of $\mathbb{A}_\tau$ are denoted by $\alpha, \beta, \ldots$. Actions in $\mathbb{A}_\tau$ can let time 1 pass before their execution, i.e. 1 is their maximal delay. After that time, they become *urgent* actions written $\underline{a}$ or $\underline{\tau}$; these have maximal delay 0. The set of urgent actions is denoted by $\underline{\mathbb{A}}_\tau = \{\underline{a} \mid a \in \mathbb{A}\} \cup \{\underline{\tau}\}$ and is ranged over by $\underline{\alpha}, \underline{\beta}, \ldots$. Elements of $\mathbb{A}_\tau \cup \underline{\mathbb{A}}_\tau$ are ranged over by μ.

$\mathcal{X}$ is the set of process variables, used for recursive definitions. Elements of $\mathcal{X}$ are denoted by $x, y, z, \ldots$ $\Phi : \mathbb{A}_\tau \rightarrow \mathbb{A}_\tau$ is a *general relabelling function* if the set $\{\alpha \in \mathbb{A}_\tau \mid \emptyset \neq \Phi^{-1}(\alpha) \neq \{\alpha\}\}$ is finite and $\Phi(\tau) = \tau$. Such a function can also be used to define *hiding*: P/A, where the actions in A are made internal, is the same as $P[\Phi_A]$, where the relabeling function Φ_A is defined by $\Phi_A(\alpha) = \tau$ if $\alpha \in A$ and $\Phi_A(\alpha) = \alpha$ if $\alpha \notin A$. We assume that time elapses in a discrete way[1]. Thus, an action prefixed process $a.P$ can either do action a and become process P (as usual in CCS) or can let one time step pass and become $\underline{a}.P$; $\underline{a}$ is called *urgent* a, and $\underline{a}.P$ as a stand-alone process cannot let time pass, but can only do a to become P.

In the following definition, initial processes are just processes of a standard process algebra extended with $\triangleright$. General processes are defined here such that they include all processes reachable from the initial ones according to the operational semantics to be defined below.

Definition 2.1 (*timed process terms*) The set $\tilde{\mathbb{P}}_1$ of *initial (timed) process terms* is generated by the following grammar

$$P ::= \mathsf{nil} \mid x \mid \alpha.P \mid \alpha \triangleright P \mid P + P \mid P \parallel_A P \mid P[\Phi] \mid \mathsf{rec}\, x.P$$

where nil is a constant, $x \in \mathcal{X}$, $\alpha \in \mathbb{A}_\tau$, Φ is a general relabelling function and $A \subseteq \mathbb{A}$ possibly infinite. We assume that recursion is action-guarded (see below). The set $\tilde{\mathbb{P}}$ of (general) *(timed) process terms* is generated by the following grammar:

$$Q ::= P \mid \underline{\alpha}.P \mid \mu \triangleright Q \mid Q + Q \mid Q \parallel_A Q \mid Q[\Phi] \mid \mathsf{rec}\, x.Q$$

where $P \in \tilde{\mathbb{P}}_1$, $x \in \mathcal{X}$, $\alpha \in \mathbb{A}_\tau$, $\mu \in \mathbb{A}_\tau \cup \underline{\mathbb{A}}_\tau$; again Φ is a general relabelling function and $A \subseteq \mathbb{A}$ possibly infinite. We assume that recursion is *action-guarded*, i.e. for $\mathsf{rec}\, x.Q$ variable x only appears in Q within the scope of a prefix $\mu.()$ with $\mu \in \mathbb{A}_\tau \cup \underline{\mathbb{A}}_\tau$. A term Q is *action-guarded* if each occurrence of a variable is guarded in this sense. A process term is *closed* if every variable x is bound by the corresponding $\mathsf{rec}\, x$-operator; the set of closed timed process terms in $\tilde{\mathbb{P}}$ and $\tilde{\mathbb{P}}_1$, simply called *processes* and *initial processes* resp., is denoted by $\mathbb{P}$ and $\mathbb{P}_1$ resp.

A brief description of the (PAFAS$_r$) operators now follows. The nil-process cannot perform any action, but may let time pass without limit. A trailing nil will often be omitted, so e.g. $a.b + c$ abbreviates $a.b.\mathsf{nil} + c.\mathsf{nil}$. $\mu.Q$ is (action-)prefixing known from CCS. Terms like $\alpha \triangleright Q$ and $\underline{\alpha} \triangleright Q$ model processes that behave like Q except for the (lazy and urgent, respectively) non-blocking action α. In both cases the action α is always enabled until component Q evolves via some ordinary action, and if urgent, $\underline{\alpha}$ stays urgent even if it is performed. $Q_1 + Q_2$ models the choice between two conflicting processes Q_1 and Q_2. $Q_1 \| _A Q_2$ is the parallel composition of two processes Q_1 and Q_2 that run in parallel and have to synchronise on all actions from A; this synchronisation discipline is inspired from TCSP. $Q[\Phi]$ behaves as Q but with the actions changed according to Φ. $\mathsf{rec}\, x.Q$ models a recursive definition; we also use equations to define recursive processes.

For our operational semantics we need some preliminary definitions. Given a process term Q, $\mathcal{U}(Q, A)$ denotes the set of the *urgent* actions of Q when the environment prevents the actions in A. For technical convenience, we allow A to be a subset of $\mathbb{A}_\tau$.

Definition 2.2 (*urgent basic actions*) Let $Q \in \tilde{\mathbb{P}}$ and $A \subseteq \mathbb{A}_\tau$. The set $\mathcal{U}(Q, A)$ is defined by induction on Q. The *urgent actions* of Q are defined as $\mathcal{U}(Q, \emptyset)$ which we abbreviate to $\mathcal{U}(Q)$.

Nil, Var: $\mathcal{U}(\mathsf{nil}, A) = \mathcal{U}(x, A) = \emptyset$

Pref: $$\mathcal{U}(\mu.P, A) = \begin{cases} \{\alpha\} & \text{if } \mu = \underline{\alpha} \text{ and } \alpha \notin A \\ \emptyset & \text{otherwise} \end{cases}$$

[1]PAFAS is not time domain dependent, meaning that the choice of discrete or continuous time makes no difference for the testing-based semantics of asynchronous systems, see [8] for more details.

Read: $\mathcal{U}(\mu \triangleright Q, A) = \begin{cases} \{\alpha\} \cup \mathcal{U}(Q, A) & \text{if } \mu = \underline{\alpha} \text{ and } \alpha \notin A \\ \mathcal{U}(Q, A) & \text{otherwise} \end{cases}$

Sum: $\mathcal{U}(Q_1 + Q_2, A) = \mathcal{U}(Q_1, A) \cup \mathcal{U}(Q_2, A)$

Par: $\mathcal{U}(Q_1 \parallel_B Q_2, A) = \bigcup_{i=1,2} \mathcal{U}(Q_i, A \cup B) \cup (\mathcal{U}(Q_1, A) \cap \mathcal{U}(Q_2, A) \cap B)$

Rel: $\mathcal{U}(Q[\Phi], A) = \Phi(\mathcal{U}(Q, \Phi^{-1}(A)))$

Rec: $\mathcal{U}(\mathsf{rec}\, x.Q, A) = \mathcal{U}(Q, A)$

The set A represents the actions restricted upon. Thus, $\mathcal{U}(\mu.P, A) = \{\alpha\}$ only if $\mu = \underline{\alpha}$ and $\alpha \notin A$; otherwise $\mathcal{U}(\mu.P, A) = \emptyset$; observe that an initial process P cannot have any urgent actions. The urgent actions of a nondeterministic process are the urgent actions of its alternative components. The essential idea for parallel composition is that a synchronised action can be delayed if at least one component can delay it: $\mathcal{U}(Q_1 \parallel_B Q_2, A)$ includes the actions that are urgent in Q_1 or Q_2 when the actions in A and in B (the synchronising ones) are prevented, and the actions in B, but not in A, that are urgent both in Q_1 and in Q_2. The other rules are as expected. Observe that $\mathcal{U}(Q, A) = \mathcal{U}(Q) \backslash A$.

The operational semantics exploits two functions on process terms: $\mathsf{clean}(\_)$ and $\mathsf{unmark}(\_)$. Function $\mathsf{clean}(\_)$ removes *all inactive urgencies* in a process term $Q \in \tilde{\mathbb{P}}$; when a process evolves and a synchronised action is no longer urgent or enabled in some synchronisation partner, then it should also lose its urgency in the others; the corresponding change of markings is performed by clean, where again set A in $\mathsf{clean}(Q, A)$ denotes the set of actions that are not enabled or urgent due to restrictions of the environment. Function $\mathsf{unmark}(\_)$ simply removes all urgencies (inactive or not) in a process term $Q \in \tilde{\mathbb{P}}$.

Definition 2.3 (*cleaning inactive urgencies*) For any $Q \in \tilde{\mathbb{P}}$ we define $\mathsf{clean}(Q)$ as $\mathsf{clean}(Q, \emptyset)$ where, for $A \subseteq \mathbb{A}$, $\mathsf{clean}(Q, A)$ is defined below.

Nil, Var: $\mathsf{clean}(\mathsf{nil}, A) = \mathsf{nil}, \qquad \mathsf{clean}(x, A) = x$

Pref: $\mathsf{clean}(\mu.P, A) = \begin{cases} \alpha.P & \text{if } \mu = \underline{\alpha} \text{ and } \alpha \in A \\ \mu.P & \text{otherwise} \end{cases}$

Read: $\mathsf{clean}(\mu \triangleright Q, A) = \begin{cases} \alpha \triangleright \mathsf{clean}(Q, A) & \text{if } \mu = \underline{\alpha} \text{ and } \alpha \in A \\ \mu \triangleright \mathsf{clean}(Q, A) & \text{otherwise} \end{cases}$

Sum: $\mathsf{clean}(Q_1 + Q_2, A) = \mathsf{clean}(Q_1, A) + \mathsf{clean}(Q_2, A)$

Par: $\mathsf{clean}(Q_1 \parallel_B Q_2, A) = \mathsf{clean}(Q_1, (B \backslash \mathcal{U}(Q_2)) \cup A) \parallel_B \mathsf{clean}(Q_2, (B \backslash \mathcal{U}(Q_1)) \cup A)$

Rel: $\mathsf{clean}(Q[\Phi], A) = \mathsf{clean}(Q, \Phi^{-1}(A))[\Phi]$

Rec: $\mathsf{clean}(\mathsf{rec}\, x.Q, A) = \mathsf{rec}\, x.\, \mathsf{clean}(Q, A)$

We say that a given $Q \in \tilde{\mathbb{P}}$ is *clean* if $Q = \mathsf{clean}(Q)$. Since, obviously, for any $P \in \tilde{\mathbb{P}}_1$ it is $P = \mathsf{clean}(P)$, we have that all initial terms are clean.

Definition 2.4 *(cleaning all urgencies)*
Let Q be a $\tilde{\mathbb{P}}$ term. Then $\mathsf{unmark}(Q)$ is defined by induction on Q as follows:

Nil, Var: $\mathsf{unmark}(\mathsf{nil}) = \mathsf{nil}, \qquad \mathsf{unmark}(x) = x$

Pref: $\mathsf{unmark}(\alpha.P) = \mathsf{unmark}(\underline{\alpha}.P) = \alpha.P$

Read: $\mathsf{unmark}(\alpha \triangleright Q) = \mathsf{unmark}(\underline{\alpha} \triangleright Q) = \alpha \triangleright \mathsf{unmark}(Q)$

Sum: $\mathsf{unmark}(Q_1 + Q_2) = \mathsf{unmark}(Q_1) + \mathsf{unmark}(Q_2)$

Par: $\mathsf{unmark}(Q_1 \parallel_A Q_2) = \mathsf{unmark}(Q_1) \parallel_A \mathsf{unmark}(Q_2)$

Rel: $\mathsf{unmark}(Q[\Phi]) = \mathsf{unmark}(Q)[\Phi]$

Rec: $\mathsf{unmark}(\mathsf{rec}\, x.Q) = \mathsf{rec}\, x.\, \mathsf{unmark}(Q)$

2.1 The functional behaviour of PAFAS$_r$ processes

The transitional semantics describing the functional behaviour of PAFAS$_r$ processes indicates which basic actions they can perform. We distinguish two different transition relations $\overset{\alpha}{\mapsto}$ and $\overset{\alpha}{\rightsquigarrow}$ to describe, resp., the ordinary and the reading behaviour of PAFAS$_r$ processes. The functional behaviour is defined as the union of these two kinds of behaviour.

Definition 2.5 (*Functional operational semantics*) Let $Q \in \tilde{\mathbb{P}}$ and $\alpha \in \mathbb{A}_\tau$. We say that $Q \xrightarrow{\alpha} Q'$ if $Q \overset{\alpha}{\mapsto} Q'$ or $Q \overset{\alpha}{\rightsquigarrow} Q'$, where the SOS-rules defining the transition relations $\overset{\alpha}{\mapsto} \subseteq (\tilde{\mathbb{P}} \times \tilde{\mathbb{P}})$ (the *ordinary action transitions*) and $\overset{\alpha}{\rightsquigarrow} \subseteq (\tilde{\mathbb{P}} \times \tilde{\mathbb{P}})$ (the *read action transitions*) for $\alpha \in \mathbb{A}_\tau$, are given in Tables 1 and 2, respectively. As usual, we write $Q \xrightarrow{\alpha} Q'$ if $(Q, Q') \in \xrightarrow{\alpha}$ and $Q \xrightarrow{\alpha}$ if there exists a $Q' \in \tilde{\mathbb{P}}$ such that $(Q, Q') \in \xrightarrow{\alpha}$, and similar conventions will apply later on.

$$\textsc{Pref}_o\ \frac{\mu \in \{\alpha, \underline{\alpha}\}}{\mu.P \overset{\alpha}{\mapsto} P} \qquad \textsc{Read}_o\ \frac{Q \overset{\alpha}{\mapsto} Q'}{\mu \triangleright Q \overset{\alpha}{\mapsto} Q'} \qquad \textsc{Sum}_o\ \frac{Q_1 \overset{\alpha}{\mapsto} Q'}{Q_1 + Q_2 \overset{\alpha}{\mapsto} Q'}$$

$$\textsc{Par}_{o1}\ \frac{\alpha \notin A,\ Q_1 \overset{\alpha}{\mapsto} Q_1'}{Q_1 \|_A Q_2 \overset{\alpha}{\mapsto} \mathsf{clean}(Q_1' \|_A Q_2)} \qquad \textsc{Par}_{o2}\ \frac{\alpha \in A,\ Q_1 \overset{\alpha}{\mapsto} Q_1',\ Q_2 \xrightarrow{\alpha} Q_2'}{Q_1 \|_A Q_2 \overset{\alpha}{\mapsto} \mathsf{clean}(Q_1' \|_A Q_2')}$$

$$\textsc{Rel}_o\ \frac{Q \overset{\alpha}{\mapsto} Q'}{Q[\Phi] \overset{\Phi(\alpha)}{\mapsto} Q'[\Phi]} \qquad \textsc{Rec}_o\ \frac{Q\{\mathsf{rec}\, x.\mathsf{unmark}(Q)/x\} \overset{\alpha}{\mapsto} Q'}{\mathsf{rec}\, x.Q \overset{\alpha}{\mapsto} Q'}$$

Table 1: Ordinary behaviour of PAFAS$_r$ processes

Rules in Table 1 are quite standard (apart from using clean in the Par-rules). Rule $\textsc{Pref}_o$ describes the behaviour of an action-prefixed process as usual in CCS. Notice that timing can be disregarded: when an action is performed, one cannot see whether it was urgent or not, and thus $\underline{\alpha}.P \overset{\alpha}{\mapsto} P$; furthermore, component $\alpha.P$ has to act *within* time 1, i.e. it can also act immediately, giving $\alpha.P \overset{\alpha}{\mapsto} P$. Rule $\textsc{Read}_o$ says that $\mu \triangleright Q$ performs the same ordinary actions as Q removing the read-prefix at the same time. The use of the unmark in rule $\textsc{Rec}_o$ has no effects for an initial process, where $\textsc{Rec}_o$ is the standard SOS rule. For non-initial Q we will explain this rule in Example 2.8. We use the clean function as explained before its definition. Observe that in rule

$\textsc{Par}_{o_2}$, an ordinary action transition can synchronise with both an ordinary and a read action transition. The other rules are as expected. Furthermore, symmetric rules have been omitted.

$$\textsc{Read}_{r1}\ \frac{\mu \in \{\alpha, \underline{\alpha}\}}{\mu \triangleright Q \overset{\alpha}{\rightsquigarrow} \mu \triangleright Q} \qquad \textsc{Read}_{r2}\ \frac{Q \overset{\alpha}{\rightsquigarrow} Q'}{\mu \triangleright Q \overset{\alpha}{\rightsquigarrow} \mu \triangleright Q'} \qquad \textsc{Sum}_{r}\ \frac{Q_1 \overset{\alpha}{\rightsquigarrow} Q_1'}{Q_1 + Q_2 \overset{\alpha}{\rightsquigarrow} Q_1' + Q_2}$$

$$\textsc{Par}_{r1}\frac{\alpha \notin A,\ Q_1 \overset{\alpha}{\rightsquigarrow} Q_1'}{Q_1\|_A Q_2 \overset{\alpha}{\rightsquigarrow} Q_1'\|_A Q_2} \qquad \textsc{Par}_{r2}\frac{\alpha \in A,\ Q_1 \overset{\alpha}{\rightsquigarrow} Q_1',\ Q_2 \overset{\alpha}{\rightsquigarrow} Q_2'}{Q_1\|_A Q_2 \overset{\alpha}{\rightsquigarrow} Q_1'\|_A Q_2'}$$

$$\textsc{Rel}_{r}\frac{Q \overset{\alpha}{\rightsquigarrow} Q'}{Q[\Phi] \overset{\Phi(\alpha)}{\rightsquigarrow} Q'[\Phi]} \qquad \textsc{Rec}_{r}\frac{Q \overset{\alpha}{\rightsquigarrow} Q'}{\mathsf{rec}\, x.Q \overset{\alpha}{\rightsquigarrow} \mathsf{rec}\, x.Q'}$$

Table 2: Reading Behaviour of PAFAS$_r$ processes

As expected, rules in Table 2 say that the execution of reading actions does not change the state of a term Q (i.e. for any $Q \in \tilde{\mathbb{P}}$, $Q \overset{\alpha}{\rightsquigarrow} Q'$ implies $Q = Q'$—this is an easy induction proof). Again, symmetric rules are omitted.

Most important is Rule $\textsc{Read}_{r_2}$. This is needed in the case of nested reading actions, and it is here that we need to distinguish read from ordinary action transitions. As an example, consider an array with two Boolean values; if the values are t and f, the behaviour of the array can be defined as $B_{tf} = r_{0t} \triangleright r_{1f} \triangleright \sum_{i\in\{t,f\}} (w_{0i}.B_{if} + w_{1i}.B_{ti})$. By r_{0t}, we mean the action of reading the value t from entry 0 and similarly for the action r_{1f}; furthermore the action w_{ji} writes value i into entry j. Now $B_{tf} \overset{r_{1f}}{\rightsquigarrow} B_{tf}$ by Rules $\textsc{Read}_{r1}$ and $\textsc{Read}_{r2}$, allowing this non-blocking action to be repeated indefinitely. At this stage, only the execution of an ordinary action can change the current state of the array. Indeed, by Rule $\textsc{Read}_{o}$, $B_{tf} \overset{w_{1t}}{\mapsto} B_{tt}$.

Definition 2.6 (*activated basic actions*) The set of *activated* (or enabled) actions of $Q \in \tilde{\mathbb{P}}$ is defined as $\mathcal{A}(Q) = \{\alpha \mid Q \xrightarrow{\alpha}\}$. The set of activated actions of Q when the environment prevents the actions in $A \subseteq \mathbb{A}_\tau$ is defined as $\mathcal{A}(Q, A) = \mathcal{A}(Q)\backslash A$.

2.2 The temporal behaviour of PAFAS$_r$ processes

We are now ready to define the refusal traces of a term $Q \in \tilde{\mathbb{P}}$. Intuitively a refusal trace records, along a computation, which actions process Q can perform ($Q \xrightarrow{\alpha} Q'$, $\alpha \in \mathbb{A}_\tau$) and which actions Q can refuse to perform when time elapses ($Q \xrightarrow{X}_r Q'$, $X \subseteq \mathbb{A}$).

A transition like $Q \xrightarrow{X}_r Q'$ is called a (partial) *time-step*. The actions listed in X are not urgent; hence Q is justified in not performing them, but performing a time step instead. This time step is partial because it can occur only in contexts that can refuse the actions not in X. If $X = \mathbb{A}$ then Q is fully justified in performing this time-step; i.e., Q can perform it independently of the environment. If $Q \xrightarrow{\mathbb{A}}_r Q'$, we write $Q \xrightarrow{1} Q'$; we say that Q performs a *full time-step* and often write $\underline{Q}$ for Q'. In [8], it is shown that inclusion of refusal traces characterises an efficiency preorder which is intuitively justified by a testing scenario. In the present paper, we need partial time steps only to set up the following SOS-semantics; our real interest is in runs where all time steps are full. We let λ range over $\mathbb{A}_\tau \cup \{1\}$.

Definition 2.7 (*refusal transitional semantics*) The inference rules in Table 3 define $\xrightarrow{X}_r \subseteq \tilde{\mathbb{P}} \times \tilde{\mathbb{P}}$ where $X \subseteq \mathbb{A}$.

$$\textsc{Nil}_t\ \frac{}{\mathsf{nil} \xrightarrow{X}_r \mathsf{nil}} \qquad \textsc{Pref}_{t1}\ \frac{}{\alpha.P \xrightarrow{X}_r \underline{\alpha}.P} \qquad \textsc{Pref}_{t2}\ \frac{\alpha \notin X \cup \{\tau\}}{\underline{\alpha}.P \xrightarrow{X}_r \underline{\alpha}.P}$$

$$\textsc{Read}_{t1}\ \frac{Q \xrightarrow{X}_r Q'}{\alpha \triangleright Q \xrightarrow{X}_r \underline{\alpha} \triangleright Q'} \qquad \textsc{Read}_{t2}\ \frac{Q \xrightarrow{X}_r Q',\ \alpha \notin X \cup \{\tau\}}{\underline{\alpha} \triangleright Q \xrightarrow{X}_r \underline{\alpha} \triangleright Q'}$$

$$\textsc{Sum}_t\ \frac{Q_i \xrightarrow{X}_r Q_i' \text{ for } i = 1,2}{Q_1 + Q_2 \xrightarrow{X}_r Q_1' + Q_2'} \qquad \textsc{Rel}_t \frac{Q \xrightarrow{\Phi^{-1}(X\cup\{\tau\})\backslash\{\tau\}}_r Q'}{Q[\Phi] \xrightarrow{X}_r Q'[\Phi]} \qquad \textsc{Rec}_t \frac{Q \xrightarrow{X}_r Q'}{\mathsf{rec}\, x.Q \xrightarrow{X}_r \mathsf{rec}\, x.Q'}$$

$$\textsc{Par}_t \frac{Q_i \xrightarrow{X_i}_r Q_i' \text{ for } i = 1,2, X \subseteq (A \cap (X_1 \cup X_2)) \cup ((X_1 \cap X_2)\backslash A)}{Q_1 \|_A Q_2 \xrightarrow{X}_r \mathsf{clean}(Q_1' \|_A Q_2')}$$

Table 3: Refusal transitional semantics of PAFAS$_r$ processes

Rule $\textsc{Pref}_{t_1}$ says that a process $\alpha.P$ can let time pass and refuse to perform any action while rule $\textsc{Pref}_{t_2}$ says that a process $\underline{\alpha}.P$ can let time pass but cannot refuse the action α. Process $\underline{\tau}.P$ cannot let time pass and cannot refuse any action; in any context, $\underline{\tau}.P$ has to perform τ before time can pass further. Rule $\textsc{Par}_t$ defines which actions a parallel composition can refuse during a time-step. The intuition is that $Q_1 \|_A Q_2$ can refuse an action α if either $\alpha \notin A$ (Q_1 and Q_2 can do α independently) and both Q_1 and Q_2 can refuse α, or $\alpha \in A$ (Q_1 and Q_2 are forced to synchronise on α) and at least one of Q_1 and Q_2 can refuse α, i.e. can delay it. Thus, an action in a parallel composition is urgent (cannot be further delayed) only when all synchronising 'local' actions are urgent (also in this case we unmark the inactive urgencies). The other rules are as expected.

Example 2.8 As an example for the definitions given so far, consider $P = (R \|_\emptyset W) \|_{\{r,w\}} V$, where $V = \mathsf{rec}\, x.(r.x + w.x)$, $R = \mathsf{rec}\, x.\, r.x$ and $W = \mathsf{rec}\, x.w.x$ model a variable (with values abstracted away), and the activities of repeatedly reading and writing such a variable, respectively. According to our operational semantics, $V \xrightarrow{1} \underline{V} = \mathsf{rec}\, x.\ (\underline{r}.x + \underline{w}.x) \overset{r}{\mapsto} V$ (each occurrence of x is replaced by V before the second transition by using unmark); hence $\textsc{Rec}_o$ meets the usual intuition. Furthermore:

$$P \xrightarrow{1} (\underline{R} \|_\emptyset \underline{W}) \|_{\{r,w\}} \underline{V} = ((\mathsf{rec}\, x.\ \underline{r}.x) \|_\emptyset (\mathsf{rec}\, x.\ \underline{w}.x)) \|_{\{r,w\}} \mathsf{rec}\, x.\ (\underline{r}.x + \underline{w}.x) \overset{r}{\mapsto} P.$$

To explain the first transition, process P can synchronise either on action r or on action w. Both actions are activated so that they become urgent after the first time-step. The second transition models the execution of action r by synchronising $\underline{R}$ and $\underline{V}$. These processes evolve into R and V, respectively. As a side effect, the urgent w in $\underline{W}$ loses its urgency since its synchronisation partner V offers a new, non-urgent synchronisation. At this stage, it is function clean in Definition 2.5 that effects this change. The above behaviour can be repeated, demonstrating that repeated reading can repeatedly delay and thus block w indefinitely.

Alternatively, we can represent the variable in a such way that an occurrence of r cannot disable the action w (i.e. a process reading V cannot block another process trying to write it). This can be done by modeling r as non-blocking, i.e. $V' = \mathsf{rec}\, x.(r \triangleright (w.x))$. According to our operational rules, $V' \xrightarrow{1} \underline{V'} = \mathsf{rec}\, x.(\underline{r} \triangleright (\underline{w}.x)) \overset{r}{\rightsquigarrow} \mathsf{rec}\, x.(\underline{r} \triangleright (\underline{w}.x)) \overset{w}{\mapsto} V'$. Hence:

$$\begin{aligned} P' &\xrightarrow{1} Q = (\underline{R} \|_\emptyset \underline{W}) \|_{\{r,w\}} \underline{V'} \xrightarrow{r} Q' = (\mathsf{rec}\, x.r.x \|_\emptyset \mathsf{rec}\, x.\underline{w}.x) \|_{\{r,w\}} \mathsf{rec}\, x.(r \triangleright (\underline{w}.x)) \xrightarrow{r} Q' \\ &\xrightarrow{w} (R \|_\emptyset W) \|_{\{r,w\}} \mathsf{rec}\, x.(r \triangleright (w.x)) = P' \end{aligned}$$

After the first occurrence of the action r (corresponding to a synchronisation between $\underline{R}$ and $\underline{V'}$),

$\underline{R}$ becomes R and offers a new, non-urgent, instance of r to its partner; this causes the unmarking of the urgent r in $\underline{V}'$. Once in Q', we can either perform an r-action, evolving again into Q', or perform an action w and come back to P'. But we cannot perform 1, i.e. w is not delayed by r in contrast to P above. Moreover, from Q we can also perform w evolving directly to P'. In this step, the urgent r in $\underline{R}$ loses its urgency as above due to a new, non-urgent, synchronisation offered by V'. Therefore, repeated writes can delay the action r arbitrarily long, i.e. writing can still block reading.

Remark 2.9 To understand the impact of read-prefixes on the timed behaviour of concurrent systems more deeply, let us consider an example where two processes need to read the same value from a variable to prepare an output they produce together. As noted in the introduction, if we use non-blocking actions to model the variable, then such an output can be produced in at most two units of time. Indeed, by our operational semantics,

$(r.o.\mathsf{nil} \parallel_{\{o\}} r.o.\mathsf{nil}) \parallel_{\{r,w\}} \mathsf{rec}\, x.r \rhd w.x \xrightarrow{1} (\underline{r}.o.\mathsf{nil} \parallel_{\{o\}} \underline{r}.o.\mathsf{nil}) \parallel_{\{r,w\}} \mathsf{rec}\, x.\underline{r} \rhd w.x \xrightarrow{r}$
$(o.\mathsf{nil} \parallel_{\{o\}} \underline{r}.o.\mathsf{nil}) \parallel_{\{r,w\}} \mathsf{rec}\, x.\underline{r} \rhd w.x \xrightarrow{r} (o.\mathsf{nil} \parallel_{\{o\}} o.\mathsf{nil}) \parallel_{\{r,w\}} \mathsf{rec}\, x.r \rhd w.x \xrightarrow{1}$
$(\underline{o}.\mathsf{nil} \parallel_{\{o\}} \underline{o}.\mathsf{nil}) \parallel_{\{r,w\}} \mathsf{rec}\, x.r \rhd w.x \xrightarrow{o}$

If we use standard ordinary actions, then producing the output takes in the worst-case three units of time. Indeed:

$(r.o.\mathsf{nil} \parallel_{\{o\}} r.o.\mathsf{nil}) \parallel_{\{r,w\}} \mathsf{rec}\, x.(r.x + w.x) \xrightarrow{1} (\underline{r}.o.\mathsf{nil} \parallel_{\{o\}} \underline{r}.o.\mathsf{nil}) \parallel_{\{r,w\}} \mathsf{rec}\, x.(\underline{r}.x + w.x) \xrightarrow{r}$
$(o.\mathsf{nil} \parallel_{\{o\}} r.o.\mathsf{nil}) \parallel_{\{r,w\}} \mathsf{rec}\, x.(r.x + w.x) \xrightarrow{1} (o.\mathsf{nil} \parallel_{\{o\}} \underline{r}.o.\mathsf{nil}) \parallel_{\{r,w\}} \mathsf{rec}\, x.(\underline{r}.x + w.x) \xrightarrow{r}$
$(o.\mathsf{nil} \parallel_{\{o\}} o.\mathsf{nil}) \parallel_{\{r,w\}} \mathsf{rec}\, x.(r.x + w.x) \xrightarrow{1} (\underline{o}.\mathsf{nil} \parallel_{\{o\}} \underline{o}.\mathsf{nil}) \parallel_{\{r,w\}} \mathsf{rec}\, x.(r.x + w.x) \xrightarrow{o}$

The following proposition (proven in Appendix B) shows that all processes reachable from an initial one (which is obviously clean) are clean. Hence, all processes of interest to us are clean. The second item shows that a clean process that does not have to perform an action urgently (i.e. $\mathcal{U}(Q) = \emptyset$) does not contain urgent actions syntactically.

Proposition 2.10 Let $Q \in \tilde{\mathbb{P}}$ be a clean process. Then:

1. $Q \xrightarrow{\alpha} Q'$ or $Q \xrightarrow{X}_r Q'$ implies Q' clean;
2. $\mathcal{U}(Q) = \emptyset$ implies $Q \in \tilde{\mathbb{P}}_1$.

3 A read operator with a simpler semantics

In this section, we present a variation of the language PAFAS$_r$ as it has been defined in Section 2. For this, we define a new kind of read operator $\{\mu_1, \ldots, \mu_n\} \rhd Q$ with a slightly different syntax. The intuition is that in a term like $\{\mu_1, \ldots, \mu_n\} \rhd Q$, the read-set $\{\mu_1, \ldots, \mu_n\}$ consists of *all* the reading actions currently enabled. In this way we try to avoid terms with nested reading actions and, as a consequence, we can describe the behaviour of these new PAFAS$_s$ processes by means of a simpler timed operational semantics with just one type of action transitions. In particular, this makes it easier to implement the algorithm (described in Section 8.2) that detects violations of the liveness property of MUTEX algorithms by showing the existence of a special kind of cycles. A price to pay is that not all PAFAS$_s$ processes have a reasonable semantics; but the subset with a reasonable semantics is practically expressive enough due to the *set* of reading actions.

Definition 3.1 (*timed process terms*) The set $\tilde{\mathbb{S}}_1$ of *initial (timed) process terms* is generated by the following grammar

$$P ::= \mathsf{nil} \mid x \mid \alpha.P \mid \{\alpha_1, \ldots, \alpha_n\} \rhd P \mid P + P \mid P \parallel_A P \mid P[\Phi] \mid \mathsf{rec}\, x.P$$

where nil is a constant, $x \in \mathcal{X}$, $\alpha \in \mathbb{A}_\tau$, $\{\alpha_1, \dots, \alpha_n\} \subseteq \mathbb{A}_\tau$ finite, Φ is a general relabelling function and $A \subseteq \mathbb{A}$ possibly infinite. The set $\tilde{\mathbb{S}}$ of (general) *(timed) process terms* is generated by the following grammar:

$$Q ::= P \mid \underline{\alpha}.P \mid \{\mu_1, \dots, \mu_n\} \rhd Q \mid Q + Q \mid Q \parallel_B Q \mid Q[\Phi] \mid \mathsf{rec}\, x.Q$$

where $P \in \tilde{\mathbb{S}}_1$, $\alpha \in \mathbb{A}_\tau$ and $\{\mu_1, \dots, \mu_n\}$ is a finite subset of $\mathbb{A}_\tau \cup \underline{\mathbb{A}}_\tau$ such that for each $\alpha \in \mathbb{A}_\tau$, $\big|\{\alpha, \underline{\alpha}\} \cap \{\mu_1, \dots, \mu_n\}\big| \leq 1$, i.e. $\{\mu_1, \dots, \mu_n\}$ cannot contain two copies (one lazy and the other urgent) of the same action α. Notice that terms not satisfying such a property are not reachable from initial ones anyway (see, Section 3.1). Again, Φ is a general relabelling function and $A \subseteq \mathbb{A}$ possibly infinite. We say that a variable $x \in \mathcal{X}$ is *read-guarded* in Q if, for each subterm of Q of the form $\{\mu_1, \dots, \mu_n\} \rhd Q_1$, x is action-guarded in Q_1. In this section, we assume that recursion is action-guarded (as in Section 2) and read-guarded, i.e. for each term $\mathsf{rec}\, x.Q$, the variable x is read-guarded in Q. The set of closed timed process terms in $\tilde{\mathbb{S}}$ and $\tilde{\mathbb{S}}_1$, simply called *processes* and *initial processes* resp., is denoted by $\mathbb{S}$ and $\mathbb{S}_1$ resp.

In the following, we will focus on the so-called read-proper terms, which do have a reasonable semantics; we will prove this by relating them to PAFAS$_r$ processes with the same behaviour. Read-proper terms only have “properly” nested reading behaviour, i.e. we want to avoid terms like $\{a\} \rhd \{b\} \rhd Q$ or $\{a\} \rhd Q' + \{b\} \rhd Q$; these terms violate the intuition given above for the read-set operator. More formally, we say that $Q \in \tilde{\mathbb{S}}$ is *read-guarded* if every subterm of Q of the form $\{\mu_1, \dots, \mu_n\} \rhd Q'$ is in the scope of some μ (i.e. in some subterm $\mu.()$). A term $Q \in \tilde{\mathbb{S}}$ is *read-proper* if each subterm $Q_1 + Q_2$ is read-guarded and, for each subterm $\{\mu_1, \dots, \mu_n\} \rhd Q_1$, Q_1 is read-guarded. With this definition, neither $\{a\} \rhd \{b\} \rhd Q$ nor $\{a\} \rhd Q' + \{b\} \rhd Q$ are read-proper, since the subterm $\{b\} \rhd Q$ is not in the scope of some μ and, thus, also not read-guarded.

Definition 3.2 (*urgent basic actions*) Let $Q \in \tilde{\mathbb{S}}$ and $A \subseteq \mathbb{A}_\tau$. The set $\mathcal{U}(Q, A)$ is defined as in Definition 2.2 where the Read-rule is replaced by:

$$\textsc{Read}\text{: } \mathcal{U}(\{\mu_1, \mu_2, \dots, \mu_n\} \rhd Q, A) = (\mathcal{U}(\{\mu_1, \mu_2, \dots, \mu_n\}) \backslash A) \cup \mathcal{U}(Q, A)$$

where $\mathcal{U}(\{\mu_1, \mu_2, \dots, \mu_n\}) = \{\alpha \in \mathbb{A}_\tau \mid \underline{\alpha} \in \{\mu_1, \mu_2, \dots, \mu_n\}\}$

Definition 3.3 (*cleaning inactive urgencies*) For $Q \in \tilde{\mathbb{S}}$, we define $\mathsf{clean}(Q, A)$ as in Definition 2.3, where the Read-rule is replaced by the following one. Again, we define $\mathsf{clean}(Q)$ as $\mathsf{clean}(Q, \emptyset)$.

$$\text{Read: } \quad \mathsf{clean}(\{\mu_1, \mu_2, \dots, \mu_n\} \rhd Q, A) = \{\nu_1, \nu_2, \dots, \nu_n\} \rhd \mathsf{clean}(Q, A)$$

where $\{\nu_1, \nu_2, \dots, \nu_n\}$ is the read-set we obtain from $\{\mu_1, \mu_2, \dots, \mu_n\}$ by unmarking all urgencies in A; in more detail, for each $i \in [1, n]$, $\mu_i \in \{\alpha, \underline{\alpha}\}$ implies either (i) $\alpha \in A$ and $\nu_i = \alpha$, or (ii) $\alpha \notin A$ and $\nu_i = \mu_i$.

The definition for the function $\mathsf{unmark}(Q)$ for $Q \in \tilde{\mathbb{S}}$ is omitted.

3.1 The timed operational semantics of PAFAS$_s$ processes

Now, we describe the transitional semantics describing the functional behaviour and the temporal behaviour of PAFAS$_s$ terms.

Definition 3.4 (*Functional operational semantics*) The SOS-rules defining the transition relations $\xrightarrow{\alpha} \subseteq (\tilde{\mathbb{S}} \times \tilde{\mathbb{S}})$ (the *action transitions*) are those in Table 1[2] where we replace the rule $\textsc{Read}_o$ with:

$$\textsc{Read}_{s1}\ \frac{\mu_i \in \{\alpha, \underline{\alpha}\}}{\{\mu_1, \dots, \mu_n\} \rhd Q \xrightarrow{\alpha} \{\mu_1, \dots, \mu_n\} \rhd Q} \qquad \textsc{Read}_{s2}\ \frac{Q \xrightarrow{\alpha} Q'}{\{\mu_1, \dots, \mu_n\} \rhd Q \xrightarrow{\alpha} Q'}$$

[2]To be formally precise: we have to replace all arrows $\mapsto$, in Table 1 by $\rightarrow$.

Definition 3.5 (*refusal transitional semantics*) The inference rules defining the transition relation $\xrightarrow{X}_r \subseteq \tilde{\mathbb{S}} \times \tilde{\mathbb{S}}$ where $X \subseteq \mathbb{A}$ are those in Table 3 where we replace the rules $\textsc{Read}_{t1}$ and $\textsc{Read}_{t2}$ with

$$\textsc{Read}_t \; \frac{Q \xrightarrow{X}_r Q',\; \mathcal{U}(\{\mu_1,\ldots,\mu_n\}) \cap (X \cup \{\tau\}) = \emptyset}{\{\mu_1,\ldots,\mu_n\} \triangleright Q \xrightarrow{X}_r \underline{\{\mu_1,\ldots,\mu_n\}} \triangleright Q'}$$

where $\underline{\{\mu_1,\ldots,\mu_n\}}$ is obtained from $\{\mu_1,\ldots,\mu_n\}$ by replacing each α by $\underline{\alpha}$.

An essential idea of reading is that it does not change the state of a process and therefore does not block other actions. With the above operational semantics, we have $\{a\} \triangleright \{b\} \triangleright Q \xrightarrow{b} \{b\} \triangleright Q$ as well as $\{a\} \triangleright Q' + \{b\} \triangleright Q \xrightarrow{b} \{b\} \triangleright Q$, violating this idea; therefore, we exclude such processes.

4 Mapping PAFAS$_s$ into PAFAS$_r$

This section studies the expressiveness of the process description languages we have introduced above. As a first result, Theorem 4.3 shows that for each read-proper $Q \in \tilde{\mathbb{S}}$ there exists a corresponding term in $\tilde{\mathbb{P}}$ whose behaviour is bisimilar and even isomorphic to that of Q; see Appendix C for the proof of Theorem 4.3 and some other related results. On the inverse claim (i.e. each PAFAS$_r$ term can be translated into a bisimilar read-proper PAFAS$_s$ term) we will comment below.

We start by providing a translation function $[\![\_]\!]$ that maps terms in $\tilde{\mathbb{S}}$ to corresponding terms in $\tilde{\mathbb{P}}$; to regard $[\![\_]\!]$ as a function in the read-case, we have to assume that actions are totally ordered, and that the actions of a read-set are listed according to this order.

Definition 4.1 (*a translation function*) For $Q \in \tilde{\mathbb{S}}$, $[\![Q]\!]$ is defined by induction on Q as follows:

Nil, Var, Pref :	$[\![\mathsf{nil}]\!] = \mathsf{nil}, \qquad [\![x]\!] = x, \qquad [\![\mu.P]\!] = \mu.[\![P]\!]$
Read:	$[\![\{\mu_1,\ldots,\mu_n\} \triangleright Q]\!] = \mu_1 \triangleright \ldots \triangleright \mu_n \triangleright [\![Q]\!]$
Sum:	$[\![Q_1 + Q_2]\!] = [\![Q_1]\!] + [\![Q_2]\!]$
Par:	$[\![Q_1 \parallel_A Q_2]\!] = [\![Q_1]\!] \parallel_A [\![Q_2]\!]$
Rel:	$[\![Q[\Phi]]\!] = [\![Q]\!][\Phi]$
Rec:	$[\![\mathsf{rec}\, x.Q]\!] = \mathsf{rec}\, x.[\![Q]\!]$

Observe that $[\![]\!]$ is injective on read-proper terms; except for the read-case, this is easy since $[\![]\!]$ preserves all other operators. In the read case, Q is read-guarded, i.e. the top-operator of Q and $[\![Q]\!]$ is not $\triangleright$; hence, the read-set can be read off from $[\![\{\mu_1,\ldots,\mu_n\} \triangleright Q]\!]$ as the maximal sequence of $\triangleright$-prefixes the term starts with. With this observation, the following two results show that $[\![]\!]$ is an isomorphism between labelled transition systems, if we restrict it on the one hand to read-proper terms and their transitions and on the other to the images of read-proper terms and the transitions of these images. As a prerequisite for this, the first result states that the set of read-proper terms is closed w.r.t. our timed operational semantics.

Proposition 4.2 Let $Q \in \tilde{\mathbb{S}}$ be read-proper. Then: $Q \xrightarrow{\alpha} Q'$ or $Q \xrightarrow{X}_r Q'$ implies Q' is read-proper.

Theorem 4.3 For all read-proper $Q \in \tilde{\mathbb{S}}$:

1. $Q \xrightarrow{\alpha} Q'$ ($Q \xrightarrow{X}_r Q'$) implies $[\![Q]\!] \xrightarrow{\alpha} [\![Q']\!]$ ($[\![Q]\!] \xrightarrow{X}_r [\![Q']\!]$)
2. if $[\![Q]\!] \xrightarrow{\alpha} Q''$ ($[\![Q]\!] \xrightarrow{X}_r Q''$) then $Q \xrightarrow{\alpha} Q'$ ($Q \xrightarrow{X}_r Q'$) with $[\![Q']\!] = Q''$

For our application, it would be sufficient to prove that the full time steps of Q are matched; but it is hard to imagine a proof for this that does not also give the matches for all time steps.

It is not clear yet whether the inverse claim (i.e. that each PAFAS$_r$ term can be translated into a bisimilar read-proper PAFAS$_s$ term) holds. Just to give an idea of the difficulties, consider the PAFAS$_r$ process $a \triangleright (b \triangleright c.\mathsf{nil} \,\|_\emptyset\, d.\mathsf{nil})$; here, reading is not properly nested as in the images of $[\![\,]\!]$. Surprisingly, this process has the same timed behaviour as $(a \triangleright b \triangleright c.\mathsf{nil}) \,\|_{\{a\}}\, (a \triangleright d.\mathsf{nil})$, which is the translation of the read-proper $\tilde{\mathbb{S}}$ process $\{a, b\} \triangleright c.\mathsf{nil} \,\|_{\{a\}}\, \{a\} \triangleright d.\mathsf{nil}$. Hence, the proof idea for the inverse claim could be to find a transformation of arbitrary PAFAS$_r$ terms to 'properly nested' ones. Unfortunately, this seems hard for the term $a \triangleright (b \triangleright c.\mathsf{nil} \,\|_\emptyset\, a.\mathsf{nil})$; probably, we should exclude terms from consideration that are ill-formed in the sense that, in some subterm $a \triangleright Q$, a appears as an ordinary prefix in Q. So the inverse claim is still under investigation.

5 Fairness and PAFAS$_r$

In this section we briefly describe our theory of fairness. It closely follows Costa and Stirling's theory of (weak) fairness. The main ingredients of the theory are:

- *A labelling for process terms.* This allows to detect during a transition which action is actually performed; e.g., for process $P = \mathsf{rec}\, x.\alpha.x$, we need additional information to detect whether the left-hand side instance of action α or the right-hand one is performed in the transition $P\|_\emptyset\, P \xrightarrow{\alpha} P\|_\emptyset\, P$. When an action is performed, we speak of an *event*, which corresponds to a label – or actually, different from [10, 9], a duple of labels as we will see.

- *Live events.* An action of a process term is live if it can currently be performed. In a term like $a.b.\mathsf{nil}\|_{\{b\}}\, b.\mathsf{nil}$ only action a can be performed while b cannot, momentarily. Such a live action corresponds to a possible event, i.e. to a label.

- *Fair sequences.* A maximal sequence is fair when no event in a process term becomes live and then remains live throughout.

These items sketch the general methodology used by Costa and Stirling to define and isolate fair computations in [10, 9]. We now describe the three items in more detail. The definitions in the rest of this section generalise those in [4], which were mostly taken from [9] with the obvious slight variations due to the different language we are using (the timed process algebra PAFAS$_r$ with TCSP parallel composition instead of CCS). We also take from [9] those results that are language independent. The others will be proven.

5.1 A labelling for process terms

In order to determine the fairness of a transition sequence, Costa and Stirling use a labelling method. Labels are associated with basic actions and operators inside a process. Along a computation, labels are unique and, once a label disappears, it will not reappear in the process anymore. The set of *labels* is $\mathsf{LAB} = \{1, 2\}^*$ with ε as the empty label and $u, v, w, \ldots$ as typical elements; $\leq$ is the *prefix preorder* on LAB. We have that $u \leq v$ if there is $u' \in \mathsf{LAB}$ such that $v = uu'$ (and $u < v$ if $u' \in \{1, 2\}^+$). We also use the following notation:

- (Set of tuples) $\mathcal{N} = \{\langle v_1, \ldots, v_n\rangle \mid n \geq 1,\ v_1, \ldots, v_n \in \mathsf{LAB}\}$;

- (Composition of tuples) $s_1 \times s_2 = \langle v_1, \ldots, v_n, w_1, \ldots, w_m\rangle$, where $s_1, s_2 \in \mathcal{N}$ and $s_1 = \langle v_1, \ldots, v_n\rangle$, $s_2 = \langle w_1, \ldots, w_m\rangle$;

- (Composition of sets of tuples) $N \times M = \{s_1 \times s_2 \mid s_1 \in N \text{ and } s_2 \in M\}$, where $N, M \subseteq \mathcal{N}$. Note that $N = \emptyset$ or $M = \emptyset$ implies $N \times M = \emptyset$.

All PAFAS$_r$ operators and variables will now be labelled in such a way that no label occurs more than once in an expression. We call this property *unicity of labels*. As indicated above, an action being performed might correspond to a pair or more generally to a tuple of labels, namely if it is a synchronisation; cf. the definition of live events below (5.7); therefore, we call tuples of labels *event labels*. Labels (i.e. elements of LAB) are assigned systematically following the structure of PAFAS$_r$ terms usually as indices and in case of parallel composition as upper indices. Due to recursion the labelling is dynamic: the rule for rec generates new labels.

Definition 5.1 (*labelled process algebra*) The labelled process algebra $\mathsf{L}(\tilde{\mathbb{P}})$ (and similarly $\mathsf{L}(\tilde{\mathbb{P}}_1)$ etc.) is defined as $\bigcup_{u \in \mathsf{LAB}} \mathsf{L}_u(\tilde{\mathbb{P}})$, where $\mathsf{L}_u(\tilde{\mathbb{P}}) = \bigcup_{Q \in \tilde{\mathbb{P}}} \mathsf{L}_u(Q)$ and $\mathsf{L}_u(Q)$ is defined as follows:

Nil, Var:	$\mathsf{L}_u(\mathsf{nil}) = \{\mathsf{nil}_u\}, \quad \mathsf{L}_u(x) = \{x_u\}$
	In examples, we will often write nil for nil_u, if the label u is not relevant.
Pref:	$\mathsf{L}_u(\mu.P) = \{\mu_u.P' \mid P' \in \mathsf{L}_{u1}(P)\}$
Read:	$\mathsf{L}_u(\mu \triangleright Q) = \{\mu_{u1} \triangleright_u Q' \mid Q' \in \mathsf{L}_{u2}(Q)\}$
Sum:	$\mathsf{L}_u(Q_1 + Q_2) = \{Q_1' +_u Q_2' \mid Q_1' \in \mathsf{L}_{u1}(Q_1),\ Q_2' \in \mathsf{L}_{u2}(Q_2)\}$
Par:	$\mathsf{L}_u(Q_1 \parallel_A Q_2) = \{Q_1' \parallel_A^u Q_2' \mid Q_1' \in \mathsf{L}_{u1v}(Q_1),\ Q_2' \in \mathsf{L}_{u2v'}(Q_2)$ where $v, v' \in \mathsf{LAB}\}$
Rel:	$\mathsf{L}_u(Q[\Phi]) = \{Q'[\Phi_u] \mid\ Q' \in \mathsf{L}_{u1v}(Q)$ where $v \in \mathsf{LAB}\}$
Rec:	$\mathsf{L}_u(\mathsf{rec}\, x.Q) = \{\mathsf{rec}\, x_u.Q' \mid Q' \in \mathsf{L}_{u1}(Q)\}$

We assume that, in $\mathsf{rec}\, x_u.Q$, $\mathsf{rec}\, x_u$ binds all free occurrences of a labelled x. We let $\mathsf{L}(Q) = \bigcup_{u \in \mathsf{LAB}} \mathsf{L}_u(Q)$ and $\mathsf{LAB}(Q)$ is the set of labels occurring in Q.

The unicity of labels must be preserved under derivation. For this reason in the rec rule the standard substitution must be replaced by a substitution operation which also changes the labels of the substituted expression.

Definition 5.2 (*a new substitution operator*) The new substitution operation, denoted by $\{\!|\ \_\ |\!\}$, is defined on $\mathsf{L}(\tilde{\mathbb{P}})$ using the following operators:

i. $()^{+v}$ If $Q \in \mathsf{L}_u(\tilde{\mathbb{P}})$, then $(Q)^{+v} \in \mathsf{L}_{vu}(\tilde{\mathbb{P}})$ obtained by prefixing v to all labels in Q.

ii. $()_\varepsilon$ If $Q \in \mathsf{L}_u(\tilde{\mathbb{P}})$, then $(Q)_\varepsilon$ is the term in $\mathsf{L}_\varepsilon(\tilde{\mathbb{P}})$ obtained by removing the prefix u from all labels in Q. (Note that u is the unique prefix-minimal label in Q.)

Suppose $Q, Q' \in \mathsf{L}(\tilde{\mathbb{P}})$ and $x_u, \ldots, x_v$ are all free occurrences of a labelled x in Q then $Q\{\!|\ Q'/x\ |\!\} = Q\{((Q')_\varepsilon)^{+u}/x_u, \ldots, ((Q')_\varepsilon)^{+v}/x_v\}$. The motivation of this definition is that in $Q\{\!|Q'/x|\!\}$ each substituted Q' inherits the label of the x it replaces.

The relationship between activated and urgent actions of PAFAS$_r$ and of labelled PAFAS$_r$ processes is easy. We can simply define $\mathcal{U}(Q, A)$ and $\mathcal{A}(Q, A)$ for a labelled PAFAS$_r$ process Q just as in Definitions 2.6 and 2.2, resp. Indeed, labels are just annotations used to distinguish different instances of the same basic action and they do not interfere with the notions of activated and urgent actions. Similarly, the operation of removing urgencies, inactive or not, does not depend on labels. They are performed in the same way both in the unlabelled and labelled setting and can be defined as in Section 2.

Finally, the behavioural operational semantics of the labelled PAFAS$_r$ is obtained by replacing the rule Rec$_o$ in Definition 2.5 with the rule:

$$\text{Rec}_o \frac{Q\{\!|\ \mathsf{rec}\, x_u.\mathsf{unmark}(Q)/x\ |\!\} \overset{\alpha}{\mapsto} Q'}{\mathsf{rec}\, x_u.Q \xrightarrow{\alpha} Q'}$$

In the other rules we ignore the labels, but keep the labels of a static operator where $\triangleright$ is "partly" static; e.g we have the rules *removed clean*:

$$\text{Pref}_o \frac{\mu \in \{\alpha, \underline{\alpha}\}}{\mu_u.P \overset{\alpha}{\mapsto} P} \qquad\qquad \text{Read}_{r_1} \frac{\mu \in \{\alpha, \underline{\alpha}\}}{\mu_{u1} \rhd_u Q \overset{\alpha}{\leadsto} \mu_{u1} \rhd_u Q}$$

because we assume that labels are not observable when actions are performed. As a consequence, a labelled term Q and its unlabelled version, that we denote with $\mathsf{R}(Q)$, can perform exactly the same transitions, as stated by the following proposition.

Proposition 5.3 Let $Q \in \mathsf{L}_u(\tilde{\mathbb{P}})$ and $A \subseteq \mathbb{A}_\tau$. Then:

i. $Q \xrightarrow{\alpha} Q'$ $(Q \xrightarrow{X}_r Q')$ implies $\mathsf{R}(Q) \xrightarrow{\alpha} \mathsf{R}(Q')$ $(\mathsf{R}(Q) \xrightarrow{X}_r \mathsf{R}(Q'))$ in unlabelled PAFAS$_r$ and $Q' \in \mathsf{L}(\tilde{\mathbb{P}})$;

ii. if $\mathsf{R}(Q) \xrightarrow{\alpha} R$ $(\mathsf{R}(Q) \xrightarrow{X}_r R)$ in unlabelled PAFAS$_r$ then for some Q' with $R = \mathsf{R}(Q')$, we have $Q \xrightarrow{\alpha} Q'$ $(Q \xrightarrow{X}_r Q')$;

iii. $\mathcal{A}(Q, A) = \mathcal{A}(\mathsf{R}(Q), A)$ and $\mathcal{U}(Q, A) = \mathcal{U}(\mathsf{R}(Q), A)$.

As an example for 5.3 (ii), observe that for $a.\mathsf{nil}\|_\emptyset \mathsf{nil} \xrightarrow{a} \mathsf{nil}\|_\emptyset \mathsf{nil}$ and $\mathsf{R}(a_{u1}.\mathsf{nil}_{u11}\|^u_\emptyset \mathsf{nil}_{u2}) = a.\mathsf{nil}\|_\emptyset \mathsf{nil}$ we indeed have $a_{u1}.\mathsf{nil}_{u11}\|^u_\emptyset \mathsf{nil}_{u2} \xrightarrow{a} \mathsf{nil}_{u11}\|^u_\emptyset \mathsf{nil}_{u2}$; the latter term is a labelled process since we allow $Q_1' \in \mathsf{L}_{u1v}(Q_1)$ in case Par of Definition 5.1, while for example in case Pref we require $P' \in \mathsf{L}_{u1}(P)$.
The next facts are an immediate consequence of the labelling. We omit the proofs since they are a trivial extension of similar results in [9] and can be easily proven by induction on terms in $\mathsf{L}(\tilde{\mathbb{P}})$.

Fact 5.4 Let $Q \in \mathsf{L}_u(\tilde{\mathbb{P}})$. Then:

1. no label occurs more than once in Q,
2. $w \in \mathsf{LAB}(Q)$ implies $u \leq w$.

Central to labelling is the persistence and disappearance of labels under derivation. In particular, once a label disappears it can never reappear. It is these features which allow us to recognise when a component contributes to the performance of an action.

Fact 5.5 Let $Q \in \mathsf{L}_u(\tilde{\mathbb{P}})$ and $\alpha, \alpha_1, \ldots, \alpha_n \in \mathbb{A}_\tau$. Then:

1. $Q \overset{\alpha}{\leadsto} Q'$ implies $Q' \in \mathsf{L}_u(\tilde{\mathbb{P}})$. Moreover $\mathsf{LAB}(Q') = \mathsf{LAB}(Q)$;
2. $Q \overset{\alpha}{\mapsto} Q'$ implies $Q' \in \mathsf{L}_v(\tilde{\mathbb{P}})$ with $u \leq v$;
3. $Q = Q_0 \xrightarrow{\alpha_1} Q_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} Q_n$ implies $Q_i \in \mathsf{L}_{v_i}(\tilde{\mathbb{P}})$ with $u \leq v_i$. Moreover, if $w \in \mathsf{LAB}$ such that $w < u$ then $w \notin \mathsf{LAB}(Q_i)$.

Fact 5.6 Let $Q_0 \in \mathsf{L}(\tilde{\mathbb{P}})$. If $Q_0 \xrightarrow{\alpha_1} Q_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_i} Q_i \xrightarrow{\alpha_{i+1}} \ldots \xrightarrow{\alpha_n} Q_n$ and $v \in \mathsf{LAB}(Q_0) \cap \mathsf{LAB}(Q_n)$ then $v \in \mathsf{LAB}(Q_i)$, for every $i \in [0, n]$.

Throughout the rest of this section we assume the labelled calculus. However, whenever possible, labels will be left implicit to keep the notation simple (as, for instance, in proofs of statements that do not explicitly deal with labels in processes) and the same applies for the treatment of labelled processes in the next section.

5.2 Live events

To capture the fairness constraint for execution sequences, we need to define the *live events*. For $\alpha_u.\mathsf{nil}\|_{\{\alpha\}}\ \alpha_v.\mathsf{nil}$ (with labels u and v), there is only one live action. This is action α; it can be performed in only one way, i.e. there is only one α-event, which we will identify with the tuple $\langle u, v\rangle$, i.e. with the tuple of labels of 'local' α's that synchronise when the process performs α; recall that we call such tuples *event labels*.[3] Similarly, there is only one live action in $\alpha_u.\beta_v.\mathsf{nil}\,\|_{\{\beta\}}\beta_y.\mathsf{nil}$ (action α corresponding to tuple $\langle u\rangle$) because the parallel composition prevents the instance of β labelled by $\langle y\rangle$ from contributing an action. However, $\langle v, y\rangle$ becomes live, once action α is performed.

We now define $\mathsf{LE}(Q, A)$ as the set of live events of Q (when the execution of actions in A are prevented by the environment). Again for technical reasons, we allow τ to be part of A.

Definition 5.7 (*live events*) Let $Q \in \mathsf{L}(\tilde{\mathbb{P}})$, $A \subseteq \mathbb{A}_\tau$. The set $\mathsf{LE}(Q, A)$ is defined below by induction on Q. The *set of live events* in Q is defined as $\mathsf{LE}(Q, \emptyset)$ which we abbreviate to $\mathsf{LE}(Q)$.

Nil, Var: $\mathsf{LE}(\mathsf{nil}_u, A) = \mathsf{LE}(x_u, A) = \emptyset$

Pref: $$\mathsf{LE}(\mu_u.P, A) = \begin{cases} \{\langle u\rangle\} & \text{if } \mu \in \{\alpha, \underline{\alpha}\} \text{ and } \alpha \notin A \\ \emptyset & \text{otherwise} \end{cases}$$

Read: $$\mathsf{LE}(\mu_v \triangleright_u Q, A) = \begin{cases} \{\langle v\rangle\} \cup \mathsf{LE}(Q, A) & \text{if } \mu \in \{\alpha, \underline{\alpha}\} \text{ and } \alpha \notin A \\ \mathsf{LE}(Q, A) & \text{otherwise} \end{cases}$$

Sum: $\mathsf{LE}(Q_1 +_u Q_2, A) = \mathsf{LE}(Q_1, A) \cup \mathsf{LE}(Q_2, A)$

Par: $\mathsf{LE}(Q_1 \|^u_B Q_2, A) = \bigcup\limits_{i=1,2} \mathsf{LE}(Q_i, A \cup B) \cup \bigcup_{\alpha \in B\setminus A} \mathsf{LE}(Q_1, \mathbb{A}_\tau\setminus\{\alpha\}) \times \mathsf{LE}(Q_2, \mathbb{A}_\tau\setminus\{\alpha\})$

Rel: $\mathsf{LE}(Q[\Phi_u], A) = \mathsf{LE}(Q, \Phi^{-1}(A))$

Rec: $\mathsf{LE}(\mathsf{rec}\, x_u.Q, A) = \mathsf{LE}(Q, A)$

Also in this case, the set A represents the restricted actions. Then, $\mathsf{LE}(a_u.P, \{a\})$ must be empty because the action a is prevented. Note that, in the Par-case, $\mathsf{LE}(Q_1, A \cup B) \cup \mathsf{LE}(Q_2, A \cup B)$ is the set of the labels of the live actions of Q_1 and Q_2, when the environment prevents actions from A and from B, corresponding to those actions that Q_1 and Q_2 can perform independently. To properly deal with synchronisation, for all $\alpha \in B\setminus A$ we combine each live event of Q_1 corresponding to α with each live event of Q_2 corresponding to α, getting tuples of labels.

An important subset of the live events of a process Q is the subset of urgent live events, that is, those that cannot be delayed anymore.

Definition 5.8 (*urgent live events*) Let $Q \in \mathsf{L}(\tilde{\mathbb{P}})$ and $A \subseteq \mathbb{A}_\tau$. The set $\mathsf{UE}(Q, A)$ is defined as in Definition 5.7 when $\mathsf{LE}(\_)$ is replaced by $\mathsf{UE}(\_)$ and rules Pref and Read are replaced as follows. The set set of *urgent event* of Q is defined as $\mathsf{UE}(Q) = \mathsf{UE}(Q, \emptyset)$

Pref: $$\mathsf{UE}(\mu_u.P, A) = \begin{cases} \{\langle u\rangle\} & \text{if } \mu = \underline{\alpha} \text{ and } \alpha \notin A \\ \emptyset & \text{otherwise} \end{cases}$$

Read: $$\mathsf{UE}(\mu_v \triangleright_u Q, A) = \begin{cases} \{\langle v\rangle\} \cup \mathsf{UE}(Q, A) & \text{if } \mu = \underline{\alpha} \text{ and } \alpha \notin A \\ \mathsf{UE}(Q, A) & \text{otherwise} \end{cases}$$

An easy observation is the following lemma.

Lemma 5.9 Let Q be a labelled process term. Then:

1. $\mathsf{UE}(Q, A) \subseteq \mathsf{LE}(Q, A)$, for every $A \subseteq \mathbb{A}_\tau$.
2. $\langle v_1, \ldots, v_n\rangle \in \mathsf{LE}(Q)$ implies $v_i \in \mathsf{LAB}(Q)$, for every $i \in [1, n]$.

[3]Since Costa and Stirling deal with fairness of components, they have no need for tuples.

3. $Q \in \mathsf{L}(\tilde{\mathbb{P}}_1)$ implies $\mathsf{UE}(Q, A) = \emptyset$, for every $A \subseteq \mathbb{A}_\tau$.

In the rest of this section we just state some properties useful to prove our main correspondence results. Detailed proofs have been moved into the appendixes. We start with a proposition relating labels and (functional and temporal) transitions which will be used to prove Proposition 6.1. In the case of functional transitions, if an event label is urgent and live in the source process, then either the label preserves its status in the target one or one of its constituents disappears (a similar statement would hold for live events in place of urgent ones). In the case of temporal transitions the set of live events of the source state coincides with the set of live events of the target one. In addition, after the temporal move all live events become urgent.

Proposition 5.10 Let $Q, Q' \in \mathsf{L}(\tilde{\mathbb{P}})$ and $A \subseteq \mathbb{A}_\tau$. Then:

1. $Q \xrightarrow{\alpha} Q'$ and $s = \langle v_1, \ldots, v_n \rangle \in \mathsf{UE}(Q, A)$ implies either $s \in \mathsf{UE}(Q', A)$ or there exists some $j \in [1, n]$ such that $v_j \notin \mathsf{LAB}(Q')$;

2. $Q \xrightarrow{X}_r Q'$ implies $\mathsf{LE}(Q, A) = \mathsf{LE}(Q', A) = \mathsf{UE}(Q', A)$.

The next proposition relates full time-steps and urgent activated actions. A process term can perform a full time-step only if it does not have any pending urgent actions, and vice versa for a action-guarded process term. Moreover, it shows how urgent activated actions and urgent live events are strictly related. This statement will be used to prove Proposition 6.2 and 6.5.

Proposition 5.11 Let $Q \in \mathsf{L}(\tilde{\mathbb{P}})$ and $A \subseteq \mathbb{A}_\tau$. Then:

1. $Q \xrightarrow{1}$ implies $\mathcal{U}(Q) = \emptyset$;

2. Q action-guarded and $\mathcal{U}(Q) = \emptyset$ implies $Q \xrightarrow{1}$;

3. $\mathcal{U}(Q, A) = \emptyset$ if and only if $\mathsf{UE}(Q, A) = \emptyset$.

The following proposition states that process terms that are able to perform two subsequent time steps cannot exhibit any functional behaviour. Moreover, if a term cannot make any functional move (and is action-guarded), then it can let two time steps pass. Intuitively, this captures the functional deadlock of terms. Terms that cannot exhibit any functional behaviour can let any amount of time pass. The following proposition formalises this intuition. It will be used to prove Proposition 6.6.

Proposition 5.12 Let $Q, Q', Q'' \in \mathsf{L}(\tilde{\mathbb{P}})$.

1. $Q \xrightarrow{1} Q' \xrightarrow{1} Q''$ implies $Q \not\xrightarrow{\alpha}$ and $Q' \not\xrightarrow{\alpha}$ for any $\alpha \in \mathbb{A}_\tau$. Moreover $Q' = Q''$;

2. Q action-guarded and $Q \not\xrightarrow{\alpha}$ for any $\alpha \in \mathbb{A}_\tau$ implies $Q \xrightarrow{1} Q' \xrightarrow{1} Q'$

5.3 Fair execution sequences

We can now define the (weak) fairness constraint. The following definitions and results are essentially borrowed from [9], and just adapted to our notions of fairness and labelling. First of all, for a *process* Q_0, we say that a sequence of transitions $\gamma = Q_0 \xrightarrow{\lambda_0} Q_1 \xrightarrow{\lambda_1} \ldots$ with $\lambda_i \in \mathbb{A}_\tau \cup \{1\}$ is a *timed execution sequence* if it is an infinite sequence of action transitions and full time-steps; note that a maximal sequence of such transitions/steps is never finite, since for $\gamma = Q_0 \xrightarrow{\lambda_0} Q_1 \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_{n-1}} Q_n$, we have either $Q_n \xrightarrow{\alpha}$ or $Q_n \xrightarrow{1}$ by Proposition 5.12.2. The second part of this proposition is applicable, since processes are always action-guarded. Note that a timed execution sequence is *everlasting* in the sense of having infinitely many time steps if and only if it is *non-Zeno*; a Zeno run would have infinitely many actions in a finite amount of time, which in a setting with discrete time means exactly that it ends with infinitely many action transitions without a time step.

For an *initial* process P_0, we say that a sequence of transitions $\gamma = P_0 \xrightarrow{\alpha_0} P_1 \xrightarrow{\alpha_1} \ldots$ with $\alpha_i \in \mathbb{A}_\tau$ is an *execution sequence* if it is a maximal sequence of action transitions; i.e. it is infinite or ends with a process P_n such that $P_n \stackrel{\alpha}{\not\to}$ for any action α. Now we formalise fairness by calling a (timed) execution sequence *fair*, if no event becomes live and then remains live throughout.

Definition 5.13 (*fair execution sequences*) Let $\gamma = Q_0 \xrightarrow{\lambda_0} Q_1 \xrightarrow{\lambda_1} \ldots$ be an execution sequence or a timed execution sequence from Q_0; we will write '(timed) execution sequence' for such a sequence. We say that γ is *fair* if

$$\neg(\exists\, s\, \exists\, i\, .\, \forall\, k \geq i\ : s \in \mathsf{LE}(Q_k))$$

Following [9], now we present an alternative, more local, definition of fair computations which will be useful to prove our main statements.

Definition 5.14 (*B-step*) For a process Q_0, we say that $Q_0 \xrightarrow{\lambda_0} Q_1 \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_{n-1}} Q_n$ with $n > 0$ is a *timed B-step* when (i) B is a finite set of event labels, and (ii) $B \cap \mathsf{LE}(Q_0) \cap \ldots \cap \mathsf{LE}(Q_n) = \emptyset$. If $\lambda_i \in \mathbb{A}_\tau$, $i = 0, \ldots, n-1$, then the sequence is a *B-step*. If $Q_0 \xrightarrow{\lambda_0} Q_1 \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_{n-1}} Q_n$ is a (timed) B-step and $v = \lambda_0 \ldots \lambda_{n-1}$ we write $Q_0 \xrightarrow{v}_B Q_{n+1}$.

In particular, a (timed) $\mathsf{LE}(Q)$-step from Q is "locally" fair: all live events of Q lose their liveness at some point in the step.

Definition 5.15 (*fair-step sequences*) A *(timed) fair-step sequence* from Q_0 is any maximal sequence of (timed) steps of the form $Q_0 \xrightarrow{v_0}_{\mathsf{LE}(Q_0)} Q_1 \xrightarrow{v_1}_{\mathsf{LE}(Q_1)} \cdots$

A fair-step sequence is simply a concatenation of locally fair steps. If δ is a (timed) fair-step sequence, then its *associated* (timed) execution sequence is the sequence which drops all references to the sets $\mathsf{LE}(Q_i)$.

Corollary 5.16 A (timed) execution sequence is fair if and only if it is the sequence associated with a (timed) fair-step sequence.

6 Fairness and Timing

This section relates fairness and timing in a process algebraic setting. It contains two main contributions. First, we prove that all everlasting (i.e. non-Zeno) sequences of PAFAS$_r$ processes are fair. Second, we provide a characterisation of fair execution sequences of *initial* PAFAS$_r$ processes (PAFAS$_r$ processes evolving only via functional operational semantics) in terms of *timed* execution sequences. For finite state processes, one can derive from this a finite representation of the fair runs with a transition system that has arcs labelled by regular expressions, as done in [4].

6.1 Fairness of everlasting sequences

The next two propositions are key statements for proving that everlasting timed execution sequences of PAFAS processes are fair. The former result relates time steps, urgent live events and live events; the latter one relate $\mathsf{LE}(P)$-steps ans sequences of actions between consecutive time-step.

Proposition 6.1 Let Q be a labelled process term. Then: $Q \xrightarrow{X}_r Q_1 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_{n-1}} Q_n$, where $X \subseteq \mathbb{A}$ and $\alpha_1, \ldots, \alpha_{n-1} \in \mathbb{A}_\tau$, implies $\mathsf{LE}(Q_1) \cap (\mathsf{LE}(Q_n) \backslash \mathsf{UE}(Q_n)) = \emptyset$ (and by Proposition 5.10 also $\mathsf{LE}(Q) \cap (\mathsf{LE}(Q_n) \backslash \mathsf{UE}(Q_n)) = \emptyset$).

Proof: Assume, toward a contradiction, that there exists a tuple of labels $s = \langle v_1, \ldots, v_m \rangle$ such that $s \in \mathsf{LE}(Q_1) = \mathsf{UE}(Q_1)$ (by Proposition 5.10-2) and $s \in \mathsf{LE}(Q_n) \backslash \mathsf{UE}(Q_n)$. Lemma 5.9-2 and $s \in \mathsf{LE}(Q_1)$ imply $v_1, \ldots, v_m \in \mathsf{LAB}(Q_1)$. On the other hand, since $s \in \mathsf{UE}(Q_1)$ but $s \notin \mathsf{UE}(Q_n)$, we can find j, $1 \leq j < n$, such that $s \in \mathsf{UE}(Q_j)$ and $s \notin \mathsf{UE}(Q_{j+1})$. Then, by Proposition 5.10-1, there exists some $k \in [1, m]$ such that $v_k \notin \mathsf{LAB}(Q_{j+1})$. By Fact 5.6, $v_k \in \mathsf{LAB}(Q_1)$ and $v_k \notin \mathsf{LAB}(Q_{j+1})$ implies $v_k \notin \mathsf{LAB}(Q_i)$, for every $i \in [j+1, n]$ and, again by Lemma 5.9, $s \notin \mathsf{LE}(Q_i)$, for every $i \in [j+1, n]$. In particular, $s \notin \mathsf{LE}(Q_n)$ which contradicts the assumption $s \in \mathsf{LE}(Q_n) \backslash \mathsf{UE}(Q_n)$. □

Proposition 6.2 Let $Q \in \mathsf{L}(\tilde{\mathbb{P}})$ and $v, w \in (\mathbb{A}_\tau)^*$.

1. If $Q \xrightarrow{1} Q_1 \xrightarrow{v} Q_2 \xrightarrow{1}$ then $Q \xrightarrow{1v}_{\mathsf{LE}(Q)} Q_2$;
2. If $Q \xrightarrow{v} Q' \xrightarrow{1} Q'_1 \xrightarrow{w} Q'_2 \xrightarrow{1}$ then $Q \xrightarrow{v1w}_{\mathsf{LE}(Q)} Q'_2$.

Proof: Item 2 follows immediately from 1. and the definition of a timed B-step. We only prove Item 1. Assume that $Q \xrightarrow{1} Q_1 \xrightarrow{v} Q_2$. Proposition 6.1 implies $\mathsf{LE}(Q) \cap (\mathsf{LE}(Q_2) \backslash \mathsf{UE}(Q_2)) = \emptyset$. Moreover $Q_2 \xrightarrow{1}$ and Proposition 5.11 imply that $\mathcal{U}(Q_2) = \emptyset$ and $\mathsf{UE}(Q_2) = \emptyset$. Thus $\mathsf{LE}(Q) \cap \mathsf{LE}(Q_1) \cap \ldots \cap \mathsf{LE}(Q_2) \subseteq \mathsf{LE}(Q) \cap \mathsf{LE}(Q_2) = \mathsf{LE}(Q) \cap (\mathsf{LE}(Q_2) \backslash \mathsf{UE}(Q_2)) = \emptyset$. By the definition of a timed B-step, $Q \xrightarrow{1v}_{\mathsf{LE}(P)} Q_2$. □

Theorem 6.3 Let $v_0, v_1, v_2 \ldots \in (\mathbb{A}_\tau)^*$. Each everlasting timed execution sequence, i.e. each timed execution sequence of the form $\gamma = P_0 \xrightarrow{v_0} P_1 \xrightarrow{1} Q_1 \xrightarrow{v_1} P_2 \xrightarrow{1} Q_2 \xrightarrow{v_2} P_3 \xrightarrow{1} \ldots$ is fair.

Proof: By Proposition 6.2 we have that $P_0 \xrightarrow{v_0 1 v_1}_{\mathsf{LE}(P_0)} P_2$, $P_2 \xrightarrow{1v_2}_{\mathsf{LE}(P_2)} P_3$ and so on. Then γ is a sequence associated with a timed fair-step sequence and is fair by Corollary 5.16. □

Observe that an everlasting timed execution sequence, by its definition, does not depend on the labelling, i.e. it is a notion of the *unlabelled* PAFAS calculus.

6.2 Relating Timed Executions and Fair Executions

While in the previous section we have shown that every everlasting timed execution is fair, here we show that everlasting timed execution sequences of initial PAFAS processes in fact *characterise* the fair untimed executions of these processes. Observe that the latter is a notion of a *labelled untimed* process algebra (like CCS or TCSP), while the former is a notion of our *unlabelled timed* process algebra. The key statement for proving this relates B-steps and action sequences performed between two full time-steps. In more detail, we prove that whenever an initial process P can perform a sequence v of basic actions and this execution turns out to be an $\mathsf{LE}(P)$-step, then P can alternatively let time pass (perform a 1-time step) and then perform the sequence of basic actions v, and vice versa. The following proposition relates live events and transitional properties of terms in $\tilde{\mathbb{P}}_1$ and their "marked" version. The proof and related results have been moved to Appendix G.

Proposition 6.4 Let $Q \in \mathsf{L}(\tilde{\mathbb{P}})$ and $P \in \mathsf{L}(\tilde{\mathbb{P}}_1)$ such that $P = \mathsf{unmark}(Q)$. Then:

1. $\mathsf{LE}(Q, A) = \mathsf{LE}(P, A)$ for every A;

2. $Q \xrightarrow{\alpha} Q'$ implies $P \xrightarrow{\alpha} P'$ and $P' = \mathsf{unmark}(Q')$ for some P'. Moreover $\mathsf{UE}(Q', A) \subseteq \mathsf{UE}(Q, A)$ and, whenever Q' is clean and $\mathsf{UE}(Q') = \emptyset$, we have $Q' = P'$;

3. $P \xrightarrow{\alpha} P'$ implies $Q \xrightarrow{\alpha} Q'$ and $P' = \mathsf{unmark}(Q')$ for some Q'.

Now we are ready to present our key proposition relating LE-step and temporal transitions.

Proposition 6.5 Let $P_0 \in \mathsf{L}(\mathbb{P}_1)$ and $v = \alpha_1 \dots \alpha_n \in (\mathbb{A}_\tau)^+$. Then:

1. $P_0 \xrightarrow{v}_{\mathsf{LE}(P_0)} P_n$ implies $P_0 \xrightarrow{1} Q_0 \xrightarrow{v} P_n$;

2. $P_0 \xrightarrow{1} Q_0 \xrightarrow{v} Q_n \xrightarrow{1}$ implies $Q_n = P_n \in \mathsf{L}(\mathbb{P}_1)$ and $P_0 \xrightarrow{v}_{\mathsf{LE}(P_0)} P_n$.

Proof: Let us first prove Item 1. $P_0 \xrightarrow{v}_{\mathsf{LE}(P_0)} P_n$ implies $P_0 \xrightarrow{v} P_n$ and $\mathsf{LE}(P_0) \cap \dots \cap \mathsf{LE}(P_n) = \emptyset$. Since $P_0 \in \mathsf{L}(\mathbb{P}_1)$, it is $P_0 \xrightarrow{1} Q_0$ with $P_0 = \mathsf{unmark}(Q_0)$ (easy induction proof) and, by Proposition 6.4-3 and 6.4-1, $Q_0 \xrightarrow{\alpha_0} \dots \xrightarrow{\alpha_n} Q_n$ with $P_i = \mathsf{unmark}(Q_i)$ and $\mathsf{LE}(P_i) = \mathsf{LE}(Q_i)$ for every $i \in [0, n]$. Then $\mathsf{LE}(Q_0) \cap \dots \cap \mathsf{LE}(Q_n) = \emptyset$ and, since $\mathsf{UE}(S) \subseteq \mathsf{LE}(S)$ for a generic S (Lemma 5.9), also $\mathsf{UE}(Q_0) \cap \dots \cap \mathsf{UE}(Q_n) = \emptyset$. Moreover, by Proposition 6.4-2, $\mathsf{UE}(Q_{i+1}) \subseteq \mathsf{UE}(Q_i)$ for $i \in [0, n-1]$. Thus $\mathsf{UE}(Q_n) = \emptyset$, Q_n clean (by Proposition 2.10) and Proposition 6.4-2 imply $Q_n = P_n$. We get $P_0 \xrightarrow{1} Q_0 \xrightarrow{v} P_n$.

Now, assume $P_0 \xrightarrow{1} Q_0 \xrightarrow{v} Q_n \xrightarrow{1}$. By Proposition 6.1 it is $\mathsf{LE}(Q_0) \cap (\mathsf{LE}(Q_n) \backslash \mathsf{UE}(Q_n)) = \emptyset$. Moreover $Q_n \xrightarrow{1}$ and Propositions 5.11-1 and 5.11-3 imply $\mathsf{UE}(Q_n) = \emptyset$ and, hence, $\mathsf{LE}(Q_0) \cap \mathsf{LE}(Q_n) = \emptyset = \mathsf{LE}(Q_0) \cap \dots \cap \mathsf{LE}(Q_n)$. Now, $P_0 = \mathsf{unmark}(Q_0)$ (as above) Propositions 6.4-2 and 6.4-1 implies $P_0 \xrightarrow{\alpha_0} \dots \xrightarrow{\alpha_n} P_n$ with $P_i = \mathsf{unmark}(Q_i)$, $\mathsf{LE}(P_i) = \mathsf{LE}(Q_i)$ for every $i \in [0, n]$. Again by Propositions 2.10 and 6.4-2, we also have $Q_n = P_n$. Thus, by definition, $P_0 \xrightarrow{v}_{\mathsf{LE}(P_0)} P_n$. $\square$

Iterative applications of Proposition 6.5 prove the main theorems of this section. To present our characterisation results we distinguish between finite and infinite sequences of untimed processes.

Proposition 6.6 Let $P \in \mathsf{L}(\mathbb{P}_1)$ and $v_0, v_1, \dots \in (\mathbb{A}_\tau)^+$. Then:

1. For any finite fair-step sequence from P

$$P = P_0 \xrightarrow{v_0}_{\mathsf{LE}(P_0)} P_1 \xrightarrow{v_1}_{\mathsf{LE}(P_1)} P_2 \dots P_{n-1} \xrightarrow{v_{n-1}}_{\mathsf{LE}(P_{n-1})} P_n$$

there exists a timed execution sequence

$$P = P_0 \xrightarrow{1} Q_0 \xrightarrow{v_0} P_1 \xrightarrow{1} Q_1 \xrightarrow{v_1} P_2 \dots P_{n-1} \xrightarrow{1} Q_{n-1} \xrightarrow{v_{n-1}} P_n \xrightarrow{1} Q_n \xrightarrow{1} Q_n \dots$$

2. For any timed execution sequence

$$P = P_0 \xrightarrow{1} Q_0 \xrightarrow{v_0} P_1 \xrightarrow{1} Q_1 \xrightarrow{v_1} P_2 \dots P_{n-1} \xrightarrow{1} Q_{n-1} \xrightarrow{v_{n-1}} P_n \xrightarrow{1} Q_n \xrightarrow{1} Q_n \dots$$

there exists a finite fair-step sequence from P

$$P_0 \xrightarrow{v_0}_{\mathsf{LE}(P_0)} P_1 \xrightarrow{v_1}_{\mathsf{LE}(P_1)} P_2 \dots P_{n-1} \xrightarrow{v_{n-1}}_{\mathsf{LE}(P_{n-1})} P_n$$

Proof: First we prove Item 1. Assume $P_0 \xrightarrow{v_0}_{\mathsf{LE}(P_0)} P_1 \xrightarrow{v_1}_{\mathsf{LE}(P_1)} P_2 \ldots P_{n-1} \xrightarrow{v_{n-1}}_{\mathsf{LE}(P_{n-1})} P_n$ and $P_n \overset{\alpha}{\not\rightarrow}$ for any α. By iterative applications of Proposition 6.5-1 we can prove that $P_i \xrightarrow{1} Q_i \xrightarrow{v_i} P_{i+1}$ for $i \in [0, n-1]$. Moreover $P_n \in \mathsf{L}(\mathbb{P}_1)$ (and hence P_n action-guarded) and $P_n \overset{\alpha}{\not\rightarrow}$ for any $\alpha \in \mathbb{A}_\tau$ imply, by Proposition 5.12-2, $P_n \xrightarrow{1} Q_n \xrightarrow{1} Q_n$. Now, we prove Item 2. $P = P_0 \xrightarrow{1} Q_0 \xrightarrow{v_0} P_1 \xrightarrow{1} Q_1 \xrightarrow{v_1} P_2 \ldots P_{n-1} \xrightarrow{1} Q_{n-1} \xrightarrow{v_{n-1}} P_n \xrightarrow{1} Q_n \xrightarrow{1} Q_n \ldots$ Then, by iterative applications of Proposition 6.5-2, we can prove that $P = P_0 \xrightarrow{v_0}_{\mathsf{LE}(P_0)} P_1 \xrightarrow{v_1}_{\mathsf{LE}(P_1)} P_2 \ldots P_{n-1} \xrightarrow{v_{n-1}}_{\mathsf{LE}(P_{n-1})} P_n$. Moreover $P_n \xrightarrow{1} Q_n \xrightarrow{1} Q_n$ and Proposition 5.12-1 imply $P_n \overset{\alpha}{\not\rightarrow}$ for any $\alpha \in \mathbb{A}_\tau$. □

Similarly we can prove an analogous result for infinite sequences.

Proposition 6.7 Let $P \in \mathsf{L}(\mathbb{P}_1)$ and $v_0, v_1, \ldots \in (\mathbb{A}_\tau)^+$. Then:

1. For any infinite fair-step sequence from P

$$P = P_0 \xrightarrow{v_0}_{\mathsf{LE}(P_0)} P_1 \xrightarrow{v_1}_{\mathsf{LE}(P_1)} P_2 \ldots P_i \xrightarrow{v_i}_{\mathsf{LE}(P_i)} P_{i+1} \ldots$$

   there exists a timed execution sequence

$$P = P_0 \xrightarrow{1} Q_0 \xrightarrow{v_0} P_1 \xrightarrow{1} Q_1 \xrightarrow{v_1} P_2 \ldots P_i \xrightarrow{1} Q_i \xrightarrow{v_i} P_{i+1} \ldots$$

2. For any timed execution sequence

$$P = P_0 \xrightarrow{1} Q_0 \xrightarrow{v_0} P_1 \xrightarrow{1} Q_1 \xrightarrow{v_1} P_2 \ldots P_i \xrightarrow{1} Q_i \xrightarrow{v_i} P_{i+1} \ldots$$

   there exists a finite fair-step sequence from P

$$P = P_0 \xrightarrow{v_0}_{\mathsf{LE}(P_0)} P_1 \xrightarrow{v_1}_{\mathsf{LE}(P_1)} P_2 \ldots P_i \xrightarrow{v_i}_{\mathsf{LE}(P_i)} P_{i+1} \ldots$$

By Proposition 5.3 we can also remove the labels from processes P_i, Q_i in the timed computation, and by Corollary 5.16 we can replace fair-step sequences by fair execution sequences. This way, we obtain a similar correspondence result between fair executions of labelled PAFAS$_r$ and timed executions of unlabelled PAFAS$_r$.

Theorem 6.8 (*Characterisation of finite fair timed execution sequences*)
Let $P \in \mathsf{L}(\mathbb{P}_1)$ and $\alpha_0, \alpha_1, \alpha_2 \ldots \in \mathbb{A}_\tau$. Then:

1. For any finite fair execution sequence from P

$$P = P_0 \xrightarrow{\alpha_0} P_1 \xrightarrow{\alpha_1} P_2 \ldots P_{n-1} \xrightarrow{\alpha_{n-1}} P_n$$

   there exists a timed execution sequence in unlabelled PAFAS$_r$

$$\mathsf{R}(P) = R_{i_0} \xrightarrow{1} Q_{i_0} \xrightarrow{v_{i_0}} R_{i_1} \xrightarrow{1} Q_{i_1} \xrightarrow{v_{i_1}} R_{i_2} \ldots R_{i_m} \xrightarrow{1} Q_{i_m} \xrightarrow{1} Q_{i_m} \xrightarrow{1} \ldots$$

   where $i_0, i_1, \ldots, i_{m-1}, i_m \in [0, n]$ with $i_0 = 0$ and $i_m = n$, $v_{i_j} = \alpha_{i_j} \alpha_{i_j+1} \ldots \alpha_{i_{j+1}-1}$, and $R_{i_j} = \mathsf{R}(P_{i_j})$, for every $j \in [0, m]$.

2. For any timed execution sequence from $\mathsf{R}(P)$ in unlabelled PAFAS$_r$

$$\mathsf{R}(P) = R_{i_0} \xrightarrow{1} Q_{i_0} \xrightarrow{v_{i_0}} R_{i_1} \xrightarrow{1} Q_{i_1} \xrightarrow{v_{i_1}} R_{i_2} \dots R_{i_m} \xrightarrow{1} Q_{i_m} \xrightarrow{1} Q_{i_m} \xrightarrow{1} \dots$$

where $i_0 = 0$, $v_{i_j} = \alpha_{i_j}\alpha_{i_j+1} \dots \alpha_{i_{j+1}-1}$, for every $j \in [0, m]$, there exists a finite fair execution sequence

$$P = P_0 \xrightarrow{\alpha_0} P_1 \xrightarrow{\alpha_1} P_2 \dots P_{n-1} \xrightarrow{\alpha_{n-1}} P_n$$

where $i_m = n$ and $R_{i_j} = \mathsf{R}(P_{i_j})$, for every $j \in [0, m]$.

Theorem 6.9 (*Characterisation of infinite fair timed execution sequences*)
Let $P \in \mathsf{L}(\mathbb{P}_1)$ and $\alpha_0, \alpha_1, \alpha_2 \dots \in \mathbb{A}_\tau$. Then:

1. For any infinite fair execution sequence from P

$$P = P_0 \xrightarrow{\alpha_0} P_1 \xrightarrow{\alpha_1} P_2 \dots P_i \xrightarrow{\alpha_i} P_{i+1} \dots$$

there exists a timed execution sequence in unlabelled PAFAS$_r$

$$\mathsf{R}(P) = R_{i_0} \xrightarrow{1} Q_{i_0} \xrightarrow{v_{i_0}} R_{i_1} \xrightarrow{1} Q_{i_1} \xrightarrow{v_{i_1}} R_{i_2} \dots R_{i_j} \xrightarrow{1} Q_{i_j} \xrightarrow{v_{i_j}} R_{i_{j+1}} \dots$$

where $i_0 = 0$, $v_{i_j} = \alpha_{i_j}\alpha_{i_j+1} \dots \alpha_{i_{j+1}-1}$ and $R_{i_j} = \mathsf{R}(P_{i_j})$, for every $j \geq 0$.

2. For any timed execution sequence from $\mathsf{R}(P)$ in unlabelled PAFAS$_r$

$$\mathsf{R}(P) = R_{i_0} \xrightarrow{1} Q_{i_0} \xrightarrow{v_{i_0}} Q_{i_1} \xrightarrow{1} R_{i_1} \xrightarrow{v_{i_1}} R_{i_2} \dots R_{i_j} \xrightarrow{1} Q_{i_j} \xrightarrow{v_{i_j}} R_{i_{j+1}} \dots$$

where $i_0 = 0$, $v_{i_j} = \alpha_{i_j}\alpha_{i_j+1} \dots \alpha_{i_{j+1}-1}$, for every $j \geq 0$, there exists an infinite fair execution sequence

$$P = P_0 \xrightarrow{\alpha_0} P_1 \xrightarrow{\alpha_1} P_2 \dots P_i \xrightarrow{\alpha_i} P_{i+1} \dots$$

where $R_{i_j} = \mathsf{R}(P_{i_j})$, for every $j \geq 0$.

Example 6.10 Consider again $P = (R \parallel_\emptyset W) \parallel_{\{r,w\}} V$ from Example 2.8 and a run from P consisting of infinitely many r's. Such a run is fair w.r.t. actions; in particular, it is fair for w because, at each transition, process V offers a "fresh" action w for synchronisation – each time an action r is performed, a new instance of w is produced. We can use timing to see this fairness formally. After a time step, all activated actions become urgent and must be performed or get disabled before the next time step; this happens when r is performed, as we noted above: $P \xrightarrow{1} ((\mathsf{rec}\, x.\ \underline{r}.x) \parallel_\emptyset (\mathsf{rec}\, x.\ \underline{w}.x)) \parallel_{\{r,w\}} \mathsf{rec}\, x.\ (\underline{r}.x + \underline{w}.x) \xrightarrow{r} P$. If we repeat this infinitely often, we get a non-Zeno timed execution sequence related to the trace of infinitely many r's. Thus, fairness of actions allows computations along which repeated reading of a variable indefinitely blocks another process trying to write to it (and vice versa for repeated writing).

As stated in [6] this is the reason why some fair runs of Dekker's algorithm violate liveness (see below) when using standard PAFAS. Note that this problem is not specific to our setting or to our notion of fairness. In [17], Raynal writes about Dekker that possibly, " if P_i is a very fast repetitive process which ... keeps entering its critical section, ... P_j cannot set $flag[j]$ to true, being prevented from doing so by P_i's reading of the variable." He observes that liveness of the algorithm therefore depends on the liveness of the hardware. This is exactly the sort of consideration for which we have a formal treatment: read prefixes say that the hardware guarantees that at least infinite reading cannot block writing.

In the case of our example, we can prevent this kind of unwanted behaviour by modelling the action r as reading (see Example 2.8). Indeed, a run from $P' = (R \parallel_\emptyset W) \parallel_{\{r,w\}} V'$ consisting of infinitely many r is not fair, since we can have at most one time step along such a run: e.g. $P' \xrightarrow{1} (\underline{R} \parallel_\emptyset \underline{W}) \parallel_{\{r,w\}} \underline{V'} = Q \xrightarrow{r} Q' \xrightarrow{r} \ldots \xrightarrow{r} Q' \ldots$, where Q' does not allow a full time step. Now, fairness of actions ensures that a process trying to write the variable will eventually do so. Notice that, on the contrary, a run from P' consisting of infinitely many w only is still fair (again by Example 2.8, $P' \xrightarrow{1} Q \xrightarrow{w} P'$) and, hence, repeated writing of a variable can block another process trying to read it. We can prevent also this kind of behaviour if the process variable is represented as $V'' = r \triangleright (w \triangleright \mathsf{nil})$, which only makes sense if writing does not change the system state.

7 Fairness and PAFAS$_s$

In this section we briefly consider the problem of defining and characterizing fair computations of read-proper $\tilde{\mathbb{S}}$-terms. Due to our expressiveness result (Theorem 4.3) and due to the correspondence between the timed behaviour of a labelled term $Q \in \mathsf{L}(\tilde{\mathbb{P}})$ and its unlabelled version (see Proposition 5.3), fair execution sequences in the case of read-proper terms in $\tilde{\mathbb{S}}$ can be defined as follows.

Definition 7.1 (*fair execution sequences*) Let $Q_0 \in \tilde{\mathbb{S}}$ be read-proper. We say that a (timed) execution sequence $\gamma = Q_0 \xrightarrow{\lambda_0} Q_1 \xrightarrow{\lambda_1} \ldots$ (in unlabelled PAFAS$_s$) is *fair* if the (timed) execution sequence $\gamma' = Q'_0 \xrightarrow{\lambda_0} Q'_1 \xrightarrow{\lambda_1} \ldots$ (in the labelled PAFAS$_r$) is where $Q'_i \in \mathsf{L}([\![Q_i]\!])$ for each $i \geq 0$.

Again by Theorem 4.3 and Proposition 5.3, we can state the following theorem, providing a characterisation of fair execution sequences of initial read-proper PAFAS$_s$ processes in terms of timed execution sequences, as a trivial consequence of Theorems 6.8 and 6.9.

Theorem 7.2 (*Characterisation of fair timed execution sequences for PAFAS$_s$ processes*)
Let $S \in \mathbb{S}_1$ be read-proper and $\alpha_0, \alpha_1, \alpha_2 \ldots \in \mathbb{A}_\tau$. Then:

1. Any finite execution sequence from S

$$S = S_0 \xrightarrow{\alpha_0} S_1 \xrightarrow{\alpha_1} S_2 \ldots S_{n-1} \xrightarrow{\alpha_{n-1}} S_n$$

is fair iff there exists a timed execution sequence

$$R_{i_0} \xrightarrow{1} Q_{i_0} \xrightarrow{v_{i_0}} R_{i_1} \xrightarrow{1} Q_{i_1} \xrightarrow{v_{i_1}} R_{i_2} \ldots R_{i_m} \xrightarrow{1} Q_{i_m} \xrightarrow{1} Q_{i_m} \xrightarrow{1} \ldots$$

where $i_0, i_1, \ldots, i_{m-1}, i_m \in [0, n]$ with $i_0 = 0$ and $i_m = n$, $v_{i_j} = \alpha_{i_j}\alpha_{i_j+1} \ldots \alpha_{i_{j+1}-1}$, and $R_{i_j} = S_{i_j}$, for every $j \in [0, m]$.

2. Any infinite fair execution sequence from S

$$S = S_0 \xrightarrow{\alpha_0} S_1 \xrightarrow{\alpha_1} S_2 \ldots S_i \xrightarrow{\alpha_i} S_{i+1} \ldots$$

is fair iff there exists a timed execution sequence

$$R_{i_0} \xrightarrow{1} Q_{i_0} \xrightarrow{v_{i_0}} R_{i_1} \xrightarrow{1} Q_{i_1} \xrightarrow{v_{i_1}} R_{i_2} \ldots R_{i_j} \xrightarrow{1} Q_{i_j} \xrightarrow{v_{i_j}} R_{i_{j+1}} \ldots$$

where $i_0 = 0$, $v_{i_j} = \alpha_{i_j}\alpha_{i_j+1} \ldots \alpha_{i_{j+1}-1}$ and $R_{i_j} = S_{i_j}$, for every $j \geq 0$.

8 Dekker's algorithm and its liveness property

In this section we briefly describe Dekker's MUTEX algorithm. There are two processes $\mathtt{P}_1$ and $\mathtt{P}_2$, two Boolean-valued variables b_1 and b_2, whose initial values are *false*, and a variable k, which may take the values 1 and 2 and whose initial value is arbitrary. Informally, the b variables are "request" variables and k is a "turn" variable: b_i is *true* if $\mathtt{P}_i$ is requesting entry to its critical section and k is i if it is $\mathtt{P}_i$'s turn to enter its critical section. Only $\mathtt{P}_i$ writes b_i, but both processes read it. The ith process (with $i = 1, 2$) can be described as follows, where j is the index of the other process:

```
while true do begin
    ⟨noncritical section⟩;
    $b_i$ = true;
    while $b_j$ do if $k = j$ then begin
            $b_i$ := false;   while $k = j$ do skip;   $b_i$ := true;
    end;
    ⟨critical section⟩;
    $k := j$;   $b_i$ := false;
end;
```

8.1 Translating the Algorithm into PAFAS$_s$ Processes

In our translation of the algorithm into PAFAS$_s$, we use essentially the same coding as Walker in [20]. Each program variable is represented as a family of processes. For instance, the process $\mathsf{B}_1(\mathit{false})$ denotes the variable b_1 with value *false*. The *sort* of the process $\mathsf{B}_1(\mathit{false})$ (i.e. the set of actions it can ever perform) is the set $\{b_1\mathit{rf}, b_1\mathit{rt}, b_1\mathit{wf}, b_1\mathit{wt}\}$ where $b_1\mathit{rf}$ and $b_1\mathit{rt}$ represent the actions of reading the values *false* and *true* from b_1, $b_1\mathit{wf}$ and $b_1\mathit{wt}$ represent, respectively, the writing of the values *false* and *true* into b_1. Let $\mathbb{B} = \{\mathit{false}, \mathit{true}\}$ and $\mathbb{K} = \{1, 2\}$.

Definition 8.1 (*the algorithm*) Let $i \in \{1, 2\}$. We define the processes representing program variables as follows:

$$\begin{array}{lcl}
\mathsf{B}_i(\mathit{false}) & = & \{b_i\mathit{rf}, b_i\mathit{wf}\} \triangleright b_i\mathit{wt}.\mathsf{B}_i(\mathit{true}) \\
\mathsf{B}_i(\mathit{true}) & = & \{b_i\mathit{rt}, b_i\mathit{wt}\} \triangleright b_i\mathit{wf}.\mathsf{B}_i(\mathit{false}) \\
\mathsf{K}(1) & = & \{\mathit{kr1}, \mathit{kw1}\} \triangleright \mathit{kw2}.\mathsf{K}(2) \\
\mathsf{K}(2) & = & \{\mathit{kr2}, \mathit{kw2}\} \triangleright \mathit{kw1}.\mathsf{K}(1)
\end{array}$$

Let $B = \{b_i\mathit{rf}, b_i\mathit{rt}, b_i\mathit{wf}, b_i\mathit{wt} \mid i \in \{1, 2\}\} \cup \{\mathit{kr1}, \mathit{kr2}, \mathit{kw1}, \mathit{kw2}\}$ be the union of the sorts of all variables and Φ_B the relabelling function such that $\Phi_B(\alpha) = \tau$ if $\alpha \in B$ and $\Phi_B(\alpha) = \alpha$ if $\alpha \notin B$. Given $b_1, b_2 \in \mathbb{B}$, $k \in \mathbb{K}$, we define $\mathsf{PV}(b_1, b_2, k) = (\mathsf{B}_1(b_1) \parallel_\emptyset \mathsf{B}_2(b_2)) \parallel_\emptyset \mathsf{K}(k)$. Processes $\mathtt{P}_1$ and $\mathtt{P}_2$ are represented by the following PAFAS processes; the actions $\mathtt{req}_i$ and cs_i have been added to indicate the request to enter and the execution of the critical section by the process $\mathtt{P}_i$.

$$\begin{array}{ll}
P_1 = \mathtt{req}_1.b_1\mathit{wt}.P_{11} + \tau.P_1 & P_2 = \mathtt{req}_2.b_2\mathit{wt}.P_{21} + \tau.P_2 \\
P_{11} = b_2\mathit{rf}.P_{14} + b_2\mathit{rt}.P_{12} & P_{21} = b_1\mathit{rf}.P_{24} + b_1\mathit{rt}.P_{22} \\
P_{12} = \mathit{kr1}.P_{11} + \mathit{kr2}.b_1\mathit{wf}.P_{13} & P_{22} = \mathit{kr2}.P_{21} + \mathit{kr1}.b_2\mathit{wf}.P_{23} \\
P_{13} = \mathit{kr1}.b_1\mathit{wt}.P_{11} + \mathit{kr2}.P_{13} & P_{23} = \mathit{kr2}.b_2\mathit{wt}.P_{21} + \mathit{kr1}.P_{23} \\
P_{14} = \mathsf{cs}_1.\mathit{kw2}.b_1\mathit{wf}.P_1 & P_{24} = \mathsf{cs}_2.\mathit{kw1}.b_2\mathit{wf}.P_2
\end{array}$$

Now, we define the algorithm as $\mathit{Dekker} = ((P_1 \parallel P_2) \parallel_B \mathsf{PV}(\mathit{false}, \mathit{false}, 1))[\Phi_B]$. The sort of *Dekker* is the set $\mathbb{A}_d = \{\mathtt{req}_i, \mathsf{cs}_i \mid i = 1, 2\}$.

A MUTEX algorithm like Dekker's satisfies *liveness* if, in every fair trace, each $\mathtt{req}_i$ is followed by the respective cs_i. Since no process should be forced to request by the fairness assumption, P_i has the alternative of an internal move, i.e. staying in its noncritical section.

8.2 Liveness violations and catastrophic cycles

Based on PAFAS, a testing-based faster-than relation has been defined in [8] that compares processes according to their worst-case efficiency. In [7], this testing-approach is adapted to a setting where user behaviour is known to belong to a very specific, but often occurring class of request-response behaviours: processes serving these users receive requests via an action *in* and provide a response *out* for each *in*-action; it is shown how to determine an asymptotic performance measure for finite-state processes of this kind. This result only holds for request-response processes that pass certain sanity checks: they must not produce more responses than requests, and they must allow requests and provide responses in finite time. While the first requirement can easily be checked by looking at the transition system, violation of the latter requirement is characterised as the existence of so-called *catastrophic cycles* in a reduced transition system denoted $\mathsf{rRTS}(P)$. The *refusal transition system* of P – denoted by $\mathsf{RTS}(P)$ – consists of all transitions $Q \xrightarrow{\alpha} Q'$ and $Q \xrightarrow{X}_r Q'$, where Q is reachable from P via such transitions. $\mathsf{rRTS}(P)$ is obtained by removing all time steps except those $Q \xrightarrow{X}_r Q'$ where either $X = \{out\}$ and Q has some *pending out*-action or $X = \{in, out\}$; then, all processes not reachable any more are deleted as well. In the case $X = \{out\}$, some *in* has not received a response and the user is waiting for an *out*, but the process can still delay this, while being willing to accept another request immediately. The case $X = \{in, out\}$ corresponds to a full time step). A cycle is catastrophic if it contains a time step but no *in*- or *out*-transition, such that time can pass without end without any useful actions.

A tool has been developed for automatically checking whether a process of (original) PAFAS has a catastrophic cycle with the algorithm described in [7], and only recently it has been adapted to a setting with reading actions. We will now give a result that allows us to decide whether *Dekker* is live using this tool. The tool cannot be applied directly: first, *Dekker* has more than two actions; second, it can perform a full time step followed by the two internal actions of P_1 and P_2 giving a catastrophic cycle, which is not relevant for the the liveness property.

Consequently, we modify *Dekker* to obtain a new process Dekker_{io} as follows: we change the actions $\mathtt{req}_1$ and $\mathtt{cs}_1$ into τ actions, we delete the τ-summand of P_2 (see Definition 8.1) and, finally, we change the actions $\mathtt{req}_2$ and $\mathtt{cs}_2$ in *in* and *out*, respectively. With this, we get the following result (proven in Appendix H) and corollary:

Theorem 8.2 *Dekker* is live iff Dekker_{io} does not have catastrophic cycles.

Corollary 8.3 *Dekker* is live.

8.3 An alternative representation of the Dekker's algorithm

To further stress the impact of introducing non-blocking actions in PAFAS (and in general in modelling concurrent systems), in this section we show that already a slight change in our representation of Dekker's algorithm (in particular, a slightly different representation of the program variables) may have a decisive impact on the liveness property. Let Dekker_ℓ be the new version of Dekker we obtain by replacing the program variables in Definition 8.1 by the following ones.

$$\begin{array}{lcl}
\mathsf{B}_i(\mathit{false}) & = & b_i\mathit{rf} \triangleright (b_1\mathit{wf}.\mathsf{B}_1(\mathit{false}) + b_1\mathit{wt}.\mathsf{B}_1(\mathit{true})) \\
\mathsf{B}_i(\mathit{true}) & = & b_i\mathit{rt} \triangleright ((b_1\mathit{wf}.\mathsf{B}_1(\mathit{false}) + b_1\mathit{wt}.\mathsf{B}_1(\mathit{true})) \\
\mathsf{K}(i) & = & \mathit{kri} \triangleright (\mathit{kw1}.\mathsf{K}(1) + \mathit{kw2}.\mathsf{K}(2))
\end{array}$$

In this case, all write actions (even those that write the old value again) are ordinary actions. Example 8.4 shows that Dekker_ℓ is *not live* under the assumption of fairness of actions, i.e. there exist non-Zeno execution sequences of Dekker_ℓ that demonstrate the violation of liveness.

Example 8.4 Consider the following sequence of steps from Dekker_ℓ:

$$\begin{array}{rll} \mathit{Dekker}_\ell \xrightarrow{1} Q_0 = & ((\underline{P_1} \parallel \underline{P_2}) \parallel_B \mathsf{PV}(\mathit{false}, \mathit{false}, 1))[\Phi_B] & \xrightarrow{\mathtt{req}_1 \tau\, \mathtt{req}_2 \tau} \\ & ((P_{11} \parallel P_{21}) \parallel_B \mathsf{PV}(\mathit{true}, \mathit{true}, 1))[\Phi_B] & \xrightarrow{\tau\tau\tau} \\ & ((P_{11} \parallel P_{23}) \parallel_B \mathsf{PV}(\mathit{true}, \mathit{false}, 1))[\Phi_B] & \xrightarrow{\tau \mathtt{cs}_1 \tau} \\ P = & ((b_1 \mathit{wf}.P_1 \parallel P_{23}) \parallel_B \mathsf{PV}(\mathit{true}, \mathit{false}, 2))[\Phi_B] & \end{array}$$

Now, let $\mathsf{B}'_1(\mathit{true}) = b_1 \mathit{rt} \rhd (\underline{b_1 \mathit{wf}}.\mathsf{B}_1(\mathit{false}) + b_1 \mathit{wt}.\mathsf{B}_1(\mathit{true}))$, $\mathsf{K}'(2) = \underline{\mathit{kr2}}. \rhd (\mathit{kw1}.\mathsf{K}(1) + \mathit{kw2}.\mathsf{K}(2))$ and $P'_{23} = \underline{\mathit{kr2}}.b_2 \mathit{wt}.P_{21} + \mathit{kr1}.P_{23}$ (notice that we have marked as urgent the only activated actions) then:

$$\begin{array}{rll} P \xrightarrow{1} Q = & ((\underline{b_1 \mathit{wf}}.P_1 \parallel P'_{23}) \parallel_B (\mathsf{B}'_1(\mathit{true}) \parallel \mathsf{B}_2(\mathit{false}) \parallel \mathsf{K}'(2)))[\Phi_B] & \xrightarrow{\tau\, \mathtt{req}_1 \tau\, \tau\, \mathtt{cs}_1} \\ & ((\mathit{kw2}.b_1 \mathit{wf}.P_1 \parallel P'_{23}) \parallel_B (\mathsf{B}_1(\mathit{true}) \parallel \mathsf{B}_2(\mathit{false}) \parallel \mathsf{K}'(2)))[\Phi_B] & \xrightarrow{\tau} \\ P = & ((b_1 \mathit{wf}.P_1 \parallel P_{23}) \parallel_B \mathsf{PV}(\mathit{true}, \mathit{false}, 2)))[\Phi_B] & \end{array}$$

As a consequence of the execution of the last internal action (i.e. after the rewriting of the value 2 to the variable k, $\mathsf{K}'(2)$ evolves into $\mathsf{K}(2)$ offering a new, non-urgent occurrence of the action *kr2* to its synchronisation partners. This causes the unmarking of the (at this stage inactive) urgency in P'_{23}. Thus, the execution sequence $\mathit{Dekker}_\ell \xrightarrow{\mathtt{req}_1 \tau\, \mathtt{req}_2 \tau^5 \mathtt{cs}_1 \tau} P_1 \xrightarrow{\tau\, \mathtt{req}_1 \tau^2 \mathtt{cs}_1 \tau} P_1 \ldots$ is fair but not live since the process $\mathtt{P}_2$ will never enter its critical section.

References

[1] P. Bouyer, S. Haddad, P.A. Reynier. Timed Petri Nets and Timed Automata: On the Discriminating Power of Zeno Sequences. Proc. of ICALP'06, LNCS 4052, pp. 420-431, 2006.

[2] S. Christensen, N. D. Hansen. Coloured Petri nets extended with place capacities, test arcs, and inhibitor arcs. In Applications of Theory of Petri Nets, LNCS 691, pp. 186-205, 1993.

[3] F. Corradini, M.R. Di Berardini and W. Vogler. PAFAS at work: Comparing the Worst-Case Efficiency of Three Buffer Implementations. Proc. of 2nd Asia-Pacific Conference on Quality Software, APAQS 2001, pp. 231-240, IEEE, 2001.

[4] F. Corradini, M.R. Di Berardini and W. Vogler. Fairness of Actions in System Computations. *Acta Informatica* **43**, pp. 73 130, 2006.

[5] F. Corradini, M.R. Di Berardini, and W. Vogler. Fairness of Components in System Computations *Theoretical Computer Science* **356**, pp. 291-324, 2006

[6] F. Corradini, M.R. Di Berardini, and W. Vogler. Checking a Mutex Algorithm in a Process Algebra with Fairness. Proc. of CONCUR '06, pp. 142-157, LNCS 4137, 2006.

[7] F. Corradini, W. Vogler. Measuring the Performance of Asynchronous Systems with PAFAS. *Theoretical Computer Science*, **335**, pp. 187-213, 2005.

[8] F. Corradini, W. Vogler, and L. Jenner. Comparing the Worst-Case Efficiency of Asynchronous Systems with PAFAS. *Acta Informatica* **38**, pp. 735-792, 2002.

[9] G. Costa, C. Stirling. Weak and Strong Fairness in CCS. *Information and Computation* **73**, pp. 207-244, 1987.

[10] G. Costa, C. Stirling. A Fair Calculus of Communicating Systems. *Acta Informatica* **21**, pp. 417-441, 1984.

[11] F. Crazzolara, G. Winskel. Events in security protocols. In Proc. of 8th ACM conference on Computer and Communication Security, CCS'01, pp. 96-105, 2001.

[12] N. Francez. *Fairness.* Springer, 1986.

[13] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall, 1985.

[14] R. Milner. *Communication and Concurrency.* International series in computer science, Prentice Hall International, 1989.

[15] U. Montanari, F. Rossi. Contextual net. *Acta Informatica* **32**, pp. 545-596, 1995.

[16] U. Montanari, F. Rossi. Contextual occurrence nets and concurrent constraints programming. In Proc. of Graph Transformation in Computer Science, LNCS 776, pp. 280-295, 1994

[17] M. Raynal. *Algorithms for Mutual Exclusion.* North Oxford Academic, 1986.

[18] G. Ristori. Modelling Systems with Shared Resources via Petri Nets. PhD thesis, Department of Computer Science, University of Pisa, 1994.

[19] W. Vogler. Efficiency of Asynchronous Systems, Read Arcs and the MUTEX-problem. *Theoretical Computer Science* 275(1-2), pp. 589-631, 2002.

[20] D.J. Walker. Automated Analysis of Mutual Exclusion algorithms using CCS. *Formal Aspects of Computing* **1**, pp. 273-292, 1989.

A Useful Properties

In this appendix section, we state and prove some useful properties concerning some of the notions in the main body of the paper. They are not related to each other, but just useful to prove the main statements.

Proposition A.1 Let $Q \in \tilde{\mathbb{P}}$, $A, A' \subseteq \mathbb{A}_\tau$ and $\alpha \in \mathbb{A}_\tau$.

1. $\alpha \in \mathcal{U}(Q, A)$ implies $\alpha \notin A$;
2. $\alpha \in \mathcal{U}(Q, A)$ and $\alpha \notin A'$ implies $\alpha \in \mathcal{U}(Q, A')$;
3. $A \subseteq A'$ implies $\mathcal{U}(Q, A') \subseteq \mathcal{U}(Q, A)$;
4. $\mathcal{U}(Q, A) = \mathcal{U}(Q) \backslash A$.

Proof: First we prove Items 1 and 2 by induction hypothesis on $Q \in \tilde{\mathbb{P}}$.

Nil, Var: $Q = \mathsf{nil}$, $Q = x$. These cases are not possible since $\mathcal{U}(Q, A) = \emptyset$.

Pref: $Q = \mu.P_1$ with $P_1 \in \tilde{\mathbb{P}}_1$. the latter case.

   1. $\alpha \in \mathcal{U}(Q, A) \neq \emptyset$ implies $\mu = \underline{\alpha}$ and $\alpha \notin A$.
   2. Again $\alpha \in \mathcal{U}(Q, A) \neq \emptyset$ implies $\mu = \underline{\alpha}$ and $\alpha \notin A$. Thus, $\alpha \notin A'$ implies $\mathcal{U}(Q, A') = \{\alpha\}$ and, hence, $\alpha \in \mathcal{U}(Q, A')$.

Read: $Q = \mu \triangleright Q_1$.

   1. $\alpha \in \mathcal{U}(Q, A) \neq \emptyset$ implies either $\mu = \underline{\alpha}$ and $\alpha \notin A$ or $\alpha \in \mathcal{U}(Q_1, A)$. Let us consider only the latter case (since in the former one there is noting to prove). Then, by induction hypothesis, it is $\alpha \notin A$.
   2. Again $\alpha \in \mathcal{U}(Q, A) \neq \emptyset$ implies either $\mu = \underline{\alpha}$ and $\alpha \notin A$ or $\alpha \in \mathcal{U}(Q_1, A)$. In the former case, $\mu = \underline{\alpha}$ and $\alpha \notin A'$ implies $\alpha \in \mathcal{U}(Q, A')$. In the latter one, $\alpha \in \mathcal{U}(Q_1, A)$ and $\alpha \notin A'$ implies, by induction hypothesis, $\alpha \in \mathcal{U}(Q_1, A') \subseteq \mathcal{U}(Q, A')$.

Sum: $Q = Q_1 + Q_2$

   1. $\alpha \in \mathcal{U}(Q, A)$ implies $\alpha \in \mathcal{U}(Q_1, A)$ or $\alpha \in \mathcal{U}(Q_2, A)$. In both cases, by induction hypothesis, $\alpha \notin A$.
   2. $\alpha \in \mathcal{U}(Q, A)$ implies (i) $\alpha \in \mathcal{U}(Q_1, A)$ or (ii) $\alpha \in \mathcal{U}(Q_2, A)$. Consider the case (i) (the other one is similar). $\alpha \in \mathcal{U}(Q_1, A)$ and $\alpha \notin A'$ implies, by induction hypothesis, $\alpha \in \mathcal{U}(Q_1, A') \subseteq \mathcal{U}(Q, A')$.

Par: $Q = Q_1 \parallel_B Q_2$.

   1. Assume $\alpha \in \mathcal{U}(Q, A)$ and consider the following possible subcases.
      - $\alpha \in \mathcal{U}(Q_1, A \cup B)$. By induction hypothesis $\mu \notin A \cup B$ and, hence, $\mu \notin A$.
      - $\alpha \in \mathcal{U}(Q_2, A \cup B)$ or $\alpha \in (\mathcal{U}(Q_1, A) \cap \mathcal{U}(Q_2, A)) \cap B$. Both these cases can be proven as the previous one.
   2. Assume $\alpha \in \mathcal{U}(Q, A)$, $\alpha \notin A'$ and consider the following possible subcases.
      - $\alpha \in \mathcal{U}(Q_1, A \cup B)$. By Item 1, we have that $\alpha \notin A \cup B$ and, hence, $\alpha \notin B$. By induction hypothesis $\alpha \in \mathcal{U}(Q_1, A \cup B)$ and $\alpha \notin A' \cup B$ implies $\alpha \in \mathcal{U}(Q_1, A' \cup B)$.
      - $\alpha \in \mathcal{U}(Q_2, A \cup B)$. Similar to the previous case.

- $\alpha \in (\mathcal{U}(Q_1, A) \cap \mathcal{U}(Q_2, A)) \cap B$. In such a case, by induction hypothesis, we have that $\alpha \in (\mathcal{U}(Q_1, A') \cap \mathcal{U}(Q_2, A')) \cap B \subseteq \mathcal{U}(Q, A')$.

Rel: $Q = Q_1[\Phi]$.

1. If $\alpha \in \mathcal{U}(Q, A) = \Phi(\mathcal{U}(Q_1, \Phi^{-1}(A)))$ then $\alpha = \Phi(\beta)$ for some $\beta \in \mathcal{U}(Q_1, \Phi^{-1}(A))$. By induction hypothesis $\beta \notin \Phi^{-1}(A) = \{\gamma \mid \Phi(\gamma) \in A\}$ and, hence, $\alpha \notin A$.
2. Again $\alpha \in \mathcal{U}(Q, A) = \Phi(\mathcal{U}(Q_1, \Phi^{-1}(A)))$ then $\alpha = \Phi(\beta)$ for some $\beta \in \mathcal{U}(Q_1, \Phi^{-1}(A))$. Moreover $\alpha \notin A'$ implies $\beta \notin \Phi^{-1}(A')$ and, by induction hypothesis, $\beta \in \mathcal{U}(Q_1, \Phi^{-1}(A'))$. Thus $\alpha = \Phi(\beta) \in \Phi(\mathcal{U}(Q_1, \Phi^{-1}(A'))) = \mathcal{U}(Q, A')$.

Rec: $Q = \mathsf{rec}\, x.Q_1$.

1. $\alpha \in \mathcal{U}(Q, A) = \mathcal{U}(Q_1, A)$ implies, by induction hypothesis, $\alpha \notin A$.
2. $\alpha \in \mathcal{U}(Q, A) = \mathcal{U}(Q_1, A)$ and $\alpha \notin A'$ implies, by induction hypothesis, $\alpha \in \mathcal{U}(Q_1, A') = \mathcal{U}(Q, A')$.

Now we can prove Items 3. and 4.

3. Assume $\alpha \in \mathcal{U}(Q, A')$. Then, by Item 1, $\alpha \notin A'$ and, since $A \subseteq A'$, $\alpha \notin A$. Thus, $\alpha \in \mathcal{U}(Q, A')$, $\alpha \notin A$ and Item 2. imply $\alpha \in \mathcal{U}(Q, A)$.
4. By Item 2, $\alpha \in \mathcal{U}(Q)\backslash A$, i.e. $\alpha \in \mathcal{U}(Q)$ and $\alpha \notin A$, implies $\alpha \in \mathcal{U}(Q, A)$. Moreover, Items 3. implies $\mathcal{U}(Q, A) \subseteq \mathcal{U}(Q)$ (since it is always $\emptyset \subseteq A$). Finally, $\alpha \in \mathcal{U}(Q, A)$ and Item 1. implies $\alpha \notin A$. Thus we can conclude that $\mathcal{U}(Q, A) \subseteq \mathcal{U}(Q)\backslash A$.

□

Proposition A.2 Let $Q, R \in \tilde{\mathbb{P}}$, $x \in \mathcal{X}$ action-guarded in Q and $A \subseteq \mathbb{A}_\tau$. Then $\mathcal{U}(Q\{R/x\}, A) = \mathcal{U}(Q, A)$.

Proof: We proceed by induction on $Q \in \tilde{\mathbb{P}}$.

Nil: $Q = \mathsf{nil}$. In this case x is action-guarded in Q and $Q\{R/x\} = \mathsf{nil}$. Thus $\mathcal{U}(Q\{R/x\}, A) = \mathcal{U}(Q, A) = \emptyset$.

Var: $Q = y$. x action-guarded in Q implies $x \neq y$ and $Q\{R/x\} = y$. Similar to the Nil-case.

Pref: $Q = \alpha.P_1$ or $Q = \underline{\alpha}.P_1$. We prove only the latter case (the formes is simpler). In this case x is guarded in Q and $Q\{R/x\} = \underline{\alpha}.(P_1\{R/x\})$. If $\alpha \notin A$ then $\mathcal{U}(Q\{R/x\}, A) = \mathcal{U}(Q, A) = \{\alpha\}$. Otherwise, $\mathcal{U}(Q\{R/x\}, A) = \mathcal{U}(Q, A) = \emptyset$.

Read: $Q = \alpha \triangleright Q_1$ or $Q = \underline{\alpha} \triangleright Q_1$. Again we prove only the latter case. In this case x action-guarded in Q implies x action-guarded in Q_1 and $Q\{R/x\} = \underline{\alpha} \triangleright (Q_1\{R/x\})$. If $\alpha \notin A$ then, by induction hypothesis, $\mathcal{U}(Q\{R/x\}, A) = \{\alpha\} \cup \mathcal{U}(Q_1\{R/x\}, A) = \{\alpha\} \cup \mathcal{U}(Q_1, A)$. Otherwise, also by induction hypothesis, $\mathcal{U}(Q\{R/x\}, A) = \mathcal{U}(Q_1\{R/x\}, A) = \mathcal{U}(Q_1, A) = \mathcal{U}(Q, A)$.

Sum: $Q = Q_1 + Q_2$. In this case x action-guarded in Q implies x action-guarded in both Q_1 and in Q_2. Moreover $Q\{R/x\} = Q_1\{R/x\} + Q_2\{R/x\}$. By induction hypothesis, we have that $\mathcal{U}(Q\{R/x\}, A) = \mathcal{U}(Q_1\{R/x\}, A) \cup \mathcal{U}(Q_2\{R/x\}, A) = \mathcal{U}(Q_1, A) \cup \mathcal{U}(Q_2, A) = \mathcal{U}(Q, A)$.

Par: $Q = Q_1 \parallel_B Q_2$. Again, x acion-guarded in Q implies x action-guarded in Q_1 and in Q_2. Moreover $Q\{R/x\} = Q_1\{R/x\} \parallel_B Q_2\{R/x\}$. By induction hypothesis $\mathcal{U}(Q\{R/x\}, A) =$
$\mathcal{U}(Q_1\{R/x\}, A \cup B) \cup \mathcal{U}(Q_2\{R/x\}, A \cup B) \cup (\mathcal{U}(Q_1\{R/x\}, A) \cap \mathcal{U}(Q_2\{R/x\}, A) \cap B) =$
$\mathcal{U}(Q_1, A \cup B) \cup \mathcal{U}(Q_2, A \cup B) \cup (\mathcal{U}(Q_1, A) \cap \mathcal{U}(Q_2, A) \cap B) = \mathcal{U}(Q, A)$.

Rel: $Q = Q_1[\Phi]$. In this case x action-guarded in Q implies x action-guarded in Q_1 and $Q\{R/x\} = (Q_1\{R/x\})[\Phi]$. By induction hypothesis, it is $\mathcal{U}(Q\{R/x\}, A) = \Phi(\mathcal{U}(Q_1\{R/x\}, \Phi^{-1}(A))) = \Phi(\mathcal{U}(Q_1, \Phi^{-1}(A))) = \mathcal{U}(Q, A)$.

Rec: $Q = \mathsf{rec}\, y.Q_1$. If $x = y$ then $Q\{R/x\} = Q$ and the statement follows easily. Assume $x \neq y$. Then x action-guarded in Q implies x action-guarded in Q_1 and $Q\{R/x\} = \mathsf{rec}\, y.(Q_1\{R/x\})$. By induction hypothesis $\mathcal{U}(Q\{R/x\}, A) = \mathcal{U}(Q_1\{R/x\}, A) = \mathcal{U}(Q_1, A) = \mathcal{U}(Q, A)$.

□

By Proposition 5.3 both Proposition A.1 and A.2 also hold for labelled terms.

Proposition A.3 Let $Q \in \mathsf{L}(\tilde{\mathbb{P}})$, A and $A' \subseteq \mathbb{A}_\tau$ with $A \subseteq A'$. Then

1. $\mathsf{LE}(Q, A') \subseteq \mathsf{LE}(Q, A)$;
2. $\mathsf{UE}(Q, A') \subseteq \mathsf{UE}(Q, A)$;

Proof: We only prove $\mathsf{LE}(Q, A') \subseteq \mathsf{LE}(Q, A)$ by induction on Q. The proof for the urgent live events is similar.

Nil, Var: $Q = \mathsf{nil}_u$, $Q = x_u$. In these cases $\mathsf{LE}(Q, A') = \mathsf{LE}(Q, A) = \emptyset$.

Pref: $Q = \alpha_u.P_1$ or $Q = \underline{\alpha}_u.P_1$. In both cases $\alpha \notin A'$ and $A \subseteq A'$ imply $\alpha \notin A$ and hence $\mathsf{LE}(Q, A') = \mathsf{LE}(Q, A) = \{u\}$. Otherwise, i.e. if $\alpha \in A'$, it is $\emptyset = \mathsf{LE}(Q, A') \subseteq \mathsf{LE}(Q, A)$.

Read: $Q = \alpha_{u1} \triangleright_u Q_1$ or $Q = \underline{\alpha}_{u1} \triangleright_u Q_1$. Assume $\alpha \notin A'$ and hence (as in the previous case) $\alpha \notin A$. By induction hypothesis, $\mathsf{LE}(Q, A') = \{u1\} \cup \mathsf{LE}(Q_1, A') \subseteq \{u1\} \cup \mathsf{LE}(Q_1, A) = \mathsf{LE}(Q, A)$. If $\alpha \in A'$ then, again by induction hypothesis, it is $\mathsf{LE}(Q, A) = \mathsf{LE}(Q_1, A') \subseteq \mathsf{LE}(Q_1, A) \subseteq \mathsf{LE}(Q, A)$.

Sum: $Q = Q_1 +_u Q_2$. By induction hypothesis it is $\mathsf{LE}(Q_i, A') \subseteq \mathsf{LE}(Q_i, A)$ for $i = 1, 2$. Thus, $\mathsf{LE}(Q_1 +_u Q_2, A') = \mathsf{LE}(Q_1, A') \cup \mathsf{LE}(Q_2, A') \subseteq \mathsf{LE}(Q_1, A) \cup \mathsf{LE}(Q_2, A) = \mathsf{LE}(Q_1 +_u Q_2, A)$.

Par: $Q = Q_1 \parallel_B^u Q_2$. $A \subseteq A'$ implies $A \cup B \subseteq A' \cup B$ and $B \backslash A' \subseteq B \backslash A$. Thus, by induction hypothesis, $\mathsf{LE}(Q_1, A' \cup B) \subseteq \mathsf{LE}(Q_1, A \cup B)$, $\mathsf{LE}(Q_2, A' \cup B) \subseteq \mathsf{LE}(Q_2, A \cup B)$ and, let $A_\alpha = \mathsf{LE}(Q_1, \mathbb{A}_\tau \backslash \{\alpha\}) \cap \mathsf{LE}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\})$, $\bigcup_{\alpha \in B \backslash A'} A_\alpha \subseteq \bigcup_{\alpha \in B \backslash A} A_\alpha$. Hence $\mathsf{LE}(Q, A') \subseteq \mathsf{LE}(Q, A)$.

Rel, Rec: Similar to the previous cases.

□

Proposition A.4 Let $Q, R \in \mathsf{L}(\tilde{\mathbb{P}})$ and $A \subseteq \mathbb{A}_\tau$.

1. $\mathsf{LE}(Q, A) \subseteq \mathsf{LE}(Q\{\!|R/x|\!\}, A)$. If $x \in \mathcal{X}$ action-guarded in Q then $\mathsf{LE}(Q\{\!|R/x|\!\}, A) \subseteq \mathsf{LE}(Q, A)$.
2. $\mathsf{UE}(Q, A) \subseteq \mathsf{UE}(Q\{\!|R/x|\!\}, A)$. If $x \in \mathcal{X}$ action-guarded in Q then $\mathsf{UE}(Q\{\!|R/x|\!\}, A) \subseteq \mathsf{UE}(Q, A)$.
3. x action-guarded in Q implies that $Q \xrightarrow{\alpha}$ if only if $Q\{\!|R/x|\!\} \xrightarrow{\mu}$.

Proof: We only prove Item 1 and Item 3 (Item 2 is similar to Item 1). To prove Item 1 we proceed by induction on Q while to prove Item 3 we proceed by induction on the derivations $Q \xrightarrow{\alpha}$ and $Q\{|R/x|\} \xrightarrow{\alpha}_r$.

Nil: $Q = \mathsf{nil}_u$. In this case x is action-guarded in Q and $Q\{|R/x|\} = \mathsf{nil}$.

1. $\mathsf{LE}(Q, A) = \emptyset = \mathsf{LE}(Q\{|R/x|\}, A)$.
3. $\mathsf{nil}_u \overset{\alpha}{\not\rightarrow}$ and $\mathsf{nil}_u\{|Q/x|\} = \mathsf{nil}_u \overset{\alpha}{\not\rightarrow}$.

Var: $Q = y_u$.

1. $\mathsf{LE}(Q, A) = \emptyset \subseteq \mathsf{LE}(Q\{|R/x|\})$. Assume x action-guarded in Q and, hence, $x \neq y$. Then $Q\{|R/x|\} = y_u$ and $\mathsf{LE}(Q\{|R/x|\}, A) = \mathsf{LE}(Q, A) = \emptyset$
3. As in the previous case x action-guarded in Q implies $x \neq y$ and $Q\{|R/x|\} = y_u$. By operational rules we have both $Q \overset{\alpha}{\not\rightarrow}$ and $Q\{|R/x|\} \overset{\alpha}{\not\rightarrow}$.

Pref: $Q = \mu_u.P_1$. Then $Q\{|R/x|\} = \mu_u.(P_1\{|R/x|\})$ and x is action-guarded in Q.

1. If $\mu \in \{\alpha, \underline{\alpha}\}$ and $\alpha \notin A$ then $\mathsf{LE}(Q, A) = \{\langle u \rangle\} = \mathsf{LE}(Q\{|R/x|\})$. Otherwise, we have $\mathsf{LE}(Q, A) = \mathsf{LE}(Q\{|R/x|\}, A) = \emptyset$.
3. $Q \xrightarrow{\alpha}$ if and only if $\mu \in \{\alpha, \underline{\alpha}\}$ if and only if, by operational rules, $Q\{|R/x|\} \xrightarrow{\alpha}$.

Read: $Q = \mu_{u1} \triangleright_u Q_1$. Then $Q\{|R/x|\} = \mu_{u1} \triangleright_u (Q_1\{|R/x|\})$ and x action-guarded in Q implies x action-guarded in Q_1.

1. If $\mu \in \{\alpha, \underline{\alpha}\}$ and $\alpha \notin A$ then $\mathsf{LE}(Q, A) = \{\langle u1 \rangle\} \cup \mathsf{LE}(Q_1, A) \subseteq \{\langle u1 \rangle\} \cup \mathsf{LE}(Q_1\{|R/x|\})$. Otherwise, we have $\mathsf{LE}(Q, A) = \mathsf{LE}(Q_1, A) \subseteq \mathsf{LE}(Q_1\{|R/x|\}, A) \subseteq \mathsf{LE}(Q\{|R/x|\}, A)$.
3. $Q \xrightarrow{\alpha}$ if and only if either $\mu \in \{\alpha, \underline{\alpha}\}$ or $Q_1 \xrightarrow{\alpha}$ if and only if, by induction hypothesis, either $\mu \in \{\alpha, \underline{\alpha}\}$ or $Q_1\{|R/x|\} \xrightarrow{\alpha}$. Finally, by operational rules, we can conclude that $Q \xrightarrow{\alpha}$ if and only if $Q\{|R/x|\} \xrightarrow{\alpha}$.

Sum: $Q = Q_1 +_u Q_2$. In this case $Q\{|R/x|\} = Q_1\{|R/x|\} +_u Q_2\{|R/x|\}$.

1. By induction hypothesis it is $\mathsf{LE}(Q, A) = \mathsf{LE}(Q_1, A) \cup \mathsf{LE}(Q_2, A) \subseteq \mathsf{LE}(Q_1\{|R/x|\}, A) \cup \mathsf{LE}(Q_2\{|R/x|\}, A) = \mathsf{LE}(Q\{|R/x|\}, A)$. Moreover, if is x action-guarded in Q then it is action-guarded in both Q_1 and in Q_2 and, by induction hypothesis, $\mathsf{LE}(Q\{|R/x|\}, A) = \mathsf{LE}(Q_1\{|R/x|\}, A) \cup \mathsf{LE}(Q_2\{|R/x|\}, A) \subseteq \mathsf{LE}(Q_1, A) \cup \mathsf{LE}(Q_2, A) = \mathsf{LE}(Q, A)$.
3. Assume x action-guarded in Q and, hence, in Q_1 and in Q_2. By operational rules we have that $Q \xrightarrow{\alpha}$ if and only if either $Q_1 \xrightarrow{\alpha}$ or $Q_2 \xrightarrow{\alpha}$ if only if, by induction hypothesis, either $Q_1\{|R/x|\} \xrightarrow{\alpha}$ or $Q_2\{|R/x|\} \xrightarrow{\alpha}$ if only if, again by operational rules, $Q\{|R/x|\} \xrightarrow{\alpha}$.

Par: $Q = Q_1 \parallel_B^u Q_2$. In this case $Q\{|R/x|\} = Q_1\{|R/x|\} \parallel_B^u Q_2\{|R/x|\}$.

1. By inductive reasoning as the previous case.
3. Assume x action-guarded in Q (and, hence, x action-guarded in Q_1 and in Q_2). Consider the following possible subcases:
   - $\alpha \notin B$. By operational rules $Q \xrightarrow{\alpha}$ if and only if either $Q_1 \xrightarrow{\alpha}$ or $Q_2 \xrightarrow{\alpha}$ if only if, by induction hypothesis, either $Q_1\{|R/x|\} \xrightarrow{\alpha}$ or $Q_2\{|R/x|\} \xrightarrow{\alpha}$ if only if, again by operational rules, $Q\{|R/x|\} \xrightarrow{\alpha}$.

- $\alpha \in B$. By operational rules $Q \xrightarrow{\alpha}$ if and only if $Q_1 \xrightarrow{\alpha}$ and $Q_2 \xrightarrow{\alpha}$ if only if, by induction hypothesis, $Q_1\{\!|R/x|\!\} \xrightarrow{\alpha}$ and $Q_2\{\!|R/x|\!\} \xrightarrow{\alpha}$ if only if, again by operational rules, $Q\{\!|R/x|\!\} \xrightarrow{\alpha}$.

Rel: $Q = Q_1[\Phi_u]$. In this case $Q\{\!|R/x|\!\} = (Q_1[\Phi_u])\{\!|R/x|\!\} = (Q_1\{\!|R/x|\!\})[\Phi_u]$

1. By induction hypothesis it is $\mathsf{LE}(Q, A) = \mathsf{LE}(Q_1, \Phi^{-1}(A)) \subseteq \mathsf{LE}(Q_1\{\!|R/x|\!\}, \Phi^{-1}(A)) = \mathsf{LE}(Q\{\!|R/x|\!\}, A)$. If x is action-guarded in Q and, hence in Q_1, again by induction hypothesis, we have that $\mathsf{LE}(Q\{\!|R/x|\!\}, A) = \mathsf{LE}(Q_1\{\!|R/x|\!\}, \Phi^{-1}(A)) \subseteq \mathsf{LE}(Q_1, \Phi^{-1}(A)) = \mathsf{LE}(Q, A)$.
3. Assume x action-guarded in Q and, hence, in Q_1. $Q \xrightarrow{\alpha}$ if only if there exists $\beta \in \Phi^{-1}(\alpha)$ such that $Q_1 \xrightarrow{\beta}$. By induction hypothesis $Q_1 \xrightarrow{\beta}$ if and only if $Q_1\{\!|R/x|\!\} \xrightarrow{\beta}$ if only if, again by operational rules, $Q\{\!|R/x|\!\} \xrightarrow{\alpha}$.

Rec: $Q = \mathsf{rec}\, y_u.Q_1$. If $x = y$ then x is action-guarded in Q, $Q\{\!|R/x|\!\} = Q$ and both statements follow easily. We can assume $x \neq y$ and $Q\{\!|R/x|\!\} = \mathsf{rec}\, y_u.(Q_1\{\!|R/x|\!\})$. By induction hypothesis:

1. $\mathsf{LE}(Q, A) = \mathsf{LE}(Q_1, A) \subseteq \mathsf{LE}(Q_1\{\!|R/x|\!\}, A) = \mathsf{LE}(Q\{\!|R/x|\!\}, A)$. Now assume x action-guarded in Q and, hence, in Q_1. By induction hypothesis it is $\mathsf{LE}(Q\{\!|R/x|\!\}, A) = \mathsf{LE}(Q_1\{\!|R/x|\!\}, A) \subseteq \mathsf{LE}(Q_1, A) = \mathsf{LE}(Q, A)$.
3. Assume x action-guarded in Q and, hence, in Q_1. Then, by operational rules, $Q \overset{\alpha}{\rightsquigarrow}$ iff $Q_1 \overset{\alpha}{\rightsquigarrow}$ iff, by induction hypothesis, $Q_1\{\!|R/x|\!\} \overset{\alpha}{\rightsquigarrow}$ iff, again by operational rules, $Q\{\!|R/x|\!\} \overset{\alpha}{\rightsquigarrow}$. Now, let us denote with $R_1 = \mathsf{unmark}(Q_1)$ and $S = Q_1\{\!|\mathsf{rec}\, y_u.R_1/y|\!\}$. Then x action-guarded in Q_1 implies x action-guarded also in $R_1 = \mathsf{unmark}(Q_1)$ and $S = Q_1\{\!|\mathsf{rec}\, y_u.R_1/y|\!\}$. Moreover, x action-guarded in Q_1 and Proposition A.7-2 implies $\mathsf{unmark}(S) = \mathsf{unmark}(Q_1\{\!|R/x|\!\}) = \mathsf{unmark}(Q_1)\{\!|R/x|\!\} = R_1\{\!|R/x|\!\}$. As a consequence, $S\{\!|R/x|\!\} = (Q_1\{\!|\mathsf{rec}\, y_u.R_1/y|\!\})\{\!|R/x|\!\} = (Q_1\{\!|R/x|\!\})\{\!|\mathsf{rec}\, y_u.(R_1\{\!|R/x|\!\})/y|\!\} = S\{\!|\mathsf{rec}\, y_u.\mathsf{unmark}(S)/y|\!\}$. Notice that, by operational rules, we have that $Q\{\!|R/x|\!\} = \mathsf{rec}\, y_u.(Q_1\{\!|R/x|\!\}) = \mathsf{rec}\, y_u.S \overset{\alpha}{\mapsto}$ if and only if $S\{\!|\mathsf{rec}\, y_u.\mathsf{unmark}(S)/y|\!\} = S\{\!|R/x|\!\} \overset{\alpha}{\mapsto}$. Finally, by operational rules, $Q \overset{\alpha}{\mapsto}$ if only if $S \overset{\alpha}{\mapsto}$ if only if, since x is action-guarded in S and by induction hypothesis, $S\{\!|R/x|\!\} \overset{\alpha}{\mapsto}$ if only if (see above) $Q\{\!|R/x|\!\} \overset{\alpha}{\mapsto}$.

□

A.1 clean and unmark Properties

In this appendix section we prove some useful properties of functions clean and unmark. Most of them are stated for terms in $\mathbb{P}$ but, since the "action" of removing urgencies does not depend from labels we can easily prove that they also hold for terms in $\mathsf{L}(\tilde{\mathbb{P}})$.

Proposition A.5 Let $Q \in \tilde{\mathbb{P}}$ and $A \subseteq \mathbb{A}$. Then:

1. $\mathsf{unmark}(\mathsf{clean}(Q, A)) = \mathsf{unmark}(Q) \in \tilde{\mathbb{P}}_1$;
2. $Q \in \tilde{\mathbb{P}}_1$ implies $\mathsf{clean}(Q, A) = \mathsf{unmark}(Q) = Q$;
3. $\mathcal{U}(Q, A) = \emptyset$ implies $\mathsf{clean}(Q, A) = \mathsf{unmark}(Q)$.

Proof: We prove, by induction on $Q \in \tilde{\mathbb{P}}$, only the latter item. Items 1 and 2 follow directly from Definitions 2.3 and 2.4.

Nil, Var: $Q = \mathsf{nil}$, $Q = x$. In these cases $\mathcal{U}(Q, A) = \emptyset$ and $\mathsf{clean}(Q, A) = \mathsf{unmark}(Q) = Q$ for any A.

Pref: $Q = \alpha.P_1$ or $Q = \underline{\alpha}.P_1$ with $P_1 \in \tilde{\mathbb{P}}_1$. We only prove the latter case (since the former one follows easily). Assume $\mathcal{U}(Q, A) = \emptyset$. Then, by Definition 2.2, we have $\alpha \in A$ and hence $\mathsf{clean}(Q, A) = \alpha.P_1 = \mathsf{unmark}(Q)$.

Read: $Q = \alpha \triangleright Q_1$ or $Q = \underline{\alpha} \triangleright Q_1$. Again we only prove the latter case. Assume $\mathcal{U}(Q, A) = \emptyset$. Then, by Definition 2.2, we have $\alpha \in A$ and $\mathcal{U}(Q_1, A) = \emptyset$. Hence, $\mathsf{clean}(Q, A) = \alpha \triangleright \mathsf{clean}(Q_1, A) = \alpha \triangleright \mathsf{unmark}(Q_1) = \mathsf{unmark}(Q)$.

Sum: $Q = Q_1 + Q_2$. Assume $\mathcal{U}(Q, A) = \mathcal{U}(Q_1, A) \cup \mathcal{U}(Q_2, A) = \emptyset$. By induction hypothesis $\mathsf{clean}(Q, A) = \mathsf{clean}(Q_1, A) + \mathsf{clean}(Q_2, A) = \mathsf{unmark}(Q_1) + \mathsf{unmark}(Q_2) = \mathsf{unmark}(Q)$.

Par: $Q = Q_1 \parallel_B Q_2$. Let $A_1 = A \cup (B \backslash \mathcal{U}(Q_2))$ and $A_2 = A \cup (B \backslash \mathcal{U}(Q_1))B$. We first prove that $\mathcal{U}(Q, A) = \emptyset$ (and hence, by Definition 2.2, $\mathcal{U}(Q_1, A \cup B) = \mathcal{U}(Q_2, A \cup B) = \emptyset$ and $\mathcal{U}(Q_1, A) \cap \mathcal{U}(Q_2, A) \cap B = \emptyset$) implies $\mathcal{U}(Q_1, A_1) = \mathcal{U}(Q_2, A_2) = \emptyset$.

Assume, toward a contradiction, that $\alpha \in \mathcal{U}(Q_1, A_1) \neq \emptyset$. and hence, by Proposition A.1-1., that $\alpha \notin A_1$ (that is $\alpha \notin A$ and $\alpha \notin B \backslash \mathcal{U}(Q_2)$). We distinguish two possible subcases: (i) $\alpha \notin B$ and (ii) $\alpha \in B$. In the (i)-case, $\alpha \in \mathcal{U}(Q_1, A_1)$ and $\alpha \notin A \cup B$ implies (by Proposition A.1-2.) $\alpha \in \mathcal{U}(Q_1, A \cup B) = \emptyset$. In the (ii)-case, $\alpha \in B$ and $\alpha \notin B \backslash \mathcal{U}(Q_2)$ implies $\alpha \in \mathcal{U}(Q_2)$. Thus $\alpha \in \mathcal{U}(Q_1, A_1) \cap \mathcal{U}(Q_2) \cap B$ and $\alpha \notin A$ implies (again Proposition A.1-2.) $\alpha \in \mathcal{U}(Q_1, A) \cap \mathcal{U}(Q_2, A) \cap B = \emptyset$. Similarly we can prove that $\mathcal{U}(Q_2, A_2) = \emptyset$.

Finally, by induction hypothesis, $\mathcal{U}(Q_1, A_1) = \emptyset$ and $\mathcal{U}(Q_2, A_2) = \emptyset$ imply $\mathsf{clean}(Q, A) = \mathsf{clean}(Q_1, A_1) \parallel_B \mathsf{clean}(Q_2, A_2) = \mathsf{unmark}(Q_1) \parallel_B \mathsf{unmark}(Q_2) = \mathsf{unmark}(Q)$.

Rel: $Q = Q_1[\Phi]$. $\mathcal{U}(Q, A) = \Phi^{-1}(\mathcal{U}(Q_1, \Phi^{-1}(A))) = \emptyset$ implies $\mathcal{U}(Q_1, \Phi^{-1}(A)) = \emptyset$ and, by induction hypothesis, $\mathsf{clean}(Q, A) = \mathsf{clean}(Q_1, \Phi^{-1}(A))[\Phi] = \mathsf{unmark}(Q_1)[\Phi] = \mathsf{unmark}(Q)$.

Rec: $Q = \mathsf{rec}\ x.Q_1$. By induction hypothesis $\mathcal{U}(Q, A) = \mathcal{U}(Q_1, A) = \emptyset$ implies $\mathsf{clean}(Q, A) = \mathsf{rec}\ x.\mathsf{clean}(Q_1, A) = \mathsf{rec}\ x.\mathsf{unmark}(Q_1) = \mathsf{unmark}(Q)$.

□

Proposition A.6 Let $Q \in \tilde{\mathbb{P}}$, $A, A' \subseteq \mathbb{A}$ and $A'' \subseteq \mathbb{A}_\tau$. Then:

1. $A' \cap \mathcal{U}(Q, A'') = \emptyset$ implies $\mathsf{clean}(Q, A) = \mathsf{clean}(Q, A \cup (A' \backslash A''))$;
2. $\mathcal{U}(\mathsf{clean}(Q, A), A'') = \mathcal{U}(Q, A \cup A'')$;
3. $\mathcal{A}(\mathsf{clean}(Q, A), A'') = \mathcal{A}(Q, A'')$

Proof: We prove Item 1. and 2. by induction on Q. Item 3. follows directly from Definitions 2.6 and 2.3.

Nil, Var: $Q = \mathsf{nil}$, $Q = x$. In these cases we have that

1. $\mathsf{clean}(Q, A) = \mathsf{clean}(Q, A \cup (A' \backslash A'')) = Q$.
2. $\mathcal{U}(\mathsf{clean}(Q, A), A'') = \mathcal{U}(Q, A \cup A'') = \emptyset$.

Pref: $Q = \alpha.P_1$ or $Q = \underline{\alpha}.P_1$. We prove only the latter case (the former one follows easily since $Q \in \tilde{\mathbb{P}}_1$).

1. If $\alpha \in A''$ then, trivially, $\alpha \notin A' \backslash A''$. If $\alpha \notin A''$, $A' \cap \mathcal{U}(Q, A'') = A' \cap \{\alpha\} = \emptyset$ implies $\alpha \notin A'$ and, again, $\alpha \notin A' \backslash A''$. In both cases, $\alpha \in A$ if and only if $\alpha \in A \cup (A' \backslash A'')$ and, by Definition 2.3, $\mathsf{clean}(Q, A) = \mathsf{clean}(Q, A \cup (A' \backslash A''))$.

2. We have to consider two possible subcases:
   - $\alpha \in A \subseteq A \cup A''$. $\mathcal{U}(\mathsf{clean}(Q, A), A') = \mathcal{U}(\alpha.Q_1, A') = \emptyset = \mathcal{U}(Q, A \cup A'')$.
   - $\alpha \notin A$. In this case $\mathsf{clean}(Q, A) = Q$. Moreover, $\alpha \notin A$ implies $\alpha \in A''$ if and only if $\alpha \in A \cup A''$ and, by Definition 2.2, $\mathcal{U}(Q, A'') = \mathcal{U}(Q, A \cup A'')$. Thus $\mathcal{U}(\mathsf{clean}(Q, A), A'') = \mathcal{U}(Q, A'') = \mathcal{U}(Q, A \cup A'')$.

Read: $Q = \alpha \triangleright Q_1$ or $Q = \underline{\alpha} \triangleright Q_1$. Again we only prove the latter case (the former one is easier).

1. Assume $A' \cap \mathcal{U}(Q, A'') = \emptyset$. Since $\mathcal{U}(Q_1, A) \subseteq \mathcal{U}(Q, A)$ it is also $A' \cap \mathcal{U}(Q_1, A'') = \emptyset$ and, by induction hypothesis, $\mathsf{clean}(Q_1, A) = \mathsf{clean}(Q_1, A \cup (A' \backslash A''))$. Moreover, if $\alpha \notin A''$ then $\alpha \in \mathcal{U}(Q, A'')$ and $A' \cap \mathcal{U}(Q, A'') = \emptyset$ implies $\alpha \notin A'$ and, hence, $\alpha \notin A' \backslash A''$. If, otherwise, $\alpha \in A''$ then it is trivially $\alpha \notin A' \backslash A''$. In both cases we have that $\alpha \in A$ if and only if $\alpha \in A \cup (A' \backslash A'')$. Thus, by Definition 2.3, we can conclude that $\mathsf{clean}(Q, A) = \mathsf{clean}(Q, A \cup (A' \backslash A''))$.
2. We distinguish the following possible subcases:
   (a) $\alpha \in A$.
   $\mathcal{U}(\mathsf{clean}(Q, A), A'') = \mathcal{U}(\alpha \triangleright \mathsf{clean}(Q_1, A), A'') = \mathcal{U}(\mathsf{clean}(Q_1, A), A'') =$ (by induction hypothesis) $\mathcal{U}(Q_1, A \cup A'') =$ (since $\alpha \in A \cup A''$) $\mathcal{U}(\underline{\alpha} \triangleright Q_1, A \cup A'')$.
   (b) $\alpha \notin A$ and $\alpha \notin A''$.
   $\mathcal{U}(\mathsf{clean}(Q, A), A'') = \mathcal{U}(\underline{\alpha} \triangleright \mathsf{clean}(Q_1, A), A'') = \{\alpha\} \cup \mathcal{U}(\mathsf{clean}(Q_1, A), A'') =$ (by induction hypothesis) $\{\alpha\} \cup \mathcal{U}(Q_1, A \cup A'') =$ (since $\alpha \notin A \cup A''$) $\mathcal{U}(\underline{\alpha} \triangleright Q_1, A \cup A'')$.
   (c) $\alpha \notin A$ and $\alpha \in A''$. This case is similar to the former one.

Sum: $Q = Q_1 + Q_2$.

1. Assume $A' \cap \mathcal{U}(Q, A'') = \emptyset$. Then, since $\mathcal{U}(Q, A'') = \mathcal{U}(Q_1, A'') \cup \mathcal{U}(Q_2, A'')$, we also have $A' \cap \mathcal{U}(Q_1, A'') = A' \cap \mathcal{U}(Q_2, A'') = \emptyset$. By induction hypothesis $\mathsf{clean}(Q, A) = \mathsf{clean}(Q_1, A) + \mathsf{clean}(Q_2, A) = \mathsf{clean}(Q_1, A \cup (A' \backslash A'')) + \mathsf{clean}(Q_2, A \cup (A' \backslash A'')) = \mathsf{clean}(Q, A \cup (A' \backslash A''))$.
2. By induction hypothesis it is $\mathcal{U}(\mathsf{clean}(Q, A), A'') = \mathcal{U}(\mathsf{clean}(Q_1, A) + \mathsf{clean}(Q_2, A), A'') = \mathcal{U}(\mathsf{clean}(Q_1, A), A'') \cup \mathcal{U}(\mathsf{clean}(Q_2, A), A'') = \mathcal{U}(Q_1, A \cup A'') \cup \mathcal{U}(Q_2, A \cup A'') = \mathcal{U}(Q, A \cup A'')$.

Par: $Q = Q_1 \parallel_B Q_2$. Let $A_1 = B \backslash \mathcal{U}(Q_2)$ and $A_2 = B \backslash \mathcal{U}(Q_1)$.

1. First we prove that $A' \cap \mathcal{U}(Q, A'') = \emptyset$ implies $(A \backslash A_i) \cap \mathcal{U}(Q_i, A'') = \emptyset$ for $i = 1, 2$.
   Assume, by contradiction, $\alpha \in (A' \backslash A_1) \cap \mathcal{U}(Q_1, A'')$. Then $\alpha \in \mathcal{U}(Q_1, A'')$ such that $\alpha \in A'$, $\alpha \notin A_1 = B \backslash \mathcal{U}(Q_2)$ and $\alpha \notin A''$ (see Proposition A.1-1.). We further distinguish the following subcases: $\alpha \notin B$ and $\alpha \in B$. In the former case, $\alpha \in \mathcal{U}(Q_1, A'')$ and $\alpha \notin A'' \cup B$ implies, Proposition A.1-2., $\alpha \in \mathcal{U}(Q_1, A'' \cup B) \subseteq \mathcal{U}(Q, A'')$. Thus $\alpha \in A' \cap \mathcal{U}(Q, A'') = \emptyset$. Otherwise, if $\alpha \in B$ then $\alpha \notin A_1 = B \backslash \mathcal{U}(Q_2)$ implies $\alpha \in \mathcal{U}(Q_2)$ and, hence, $\alpha \in \mathcal{U}(Q_1, A'') \cap \mathcal{U}(Q_2) \cap B$. Finally, $\alpha \notin A''$ and again Proposition A.1-2. imply $\alpha \in \mathcal{U}(Q_1, A'') \cap \mathcal{U}(Q_2, A'') \cap B \subseteq \mathcal{U}(Q, A'')$. Again it is $\alpha \in A' \cap \mathcal{U}(Q, A'') = \emptyset$. We can conclude that $(A' \backslash A_1) \cap \mathcal{U}(Q_1, A'') = \emptyset$ and, similarly, that $(A' \backslash A_2) \cap \mathcal{U}(Q_2, A'') = \emptyset$. By induction hypothesis it is $\mathsf{clean}(Q_i, A \cup A_i) = \mathsf{clean}(Q_i, A \cup A_i \cup ((A' \backslash A_i) \backslash A'')) = \mathsf{clean}(Q_i, A \cup A_i \cup (A' \backslash A''))$—because for generic sets A, B and C we have $A \cup ((B \backslash A) \backslash C) = A \cup (B \backslash C)$—for $i = 1, 2$. Thus, $\mathsf{clean}(Q, A) = \mathsf{clean}(Q_1, A \cup A_1) \parallel_B \mathsf{clean}(Q_2, A \cup A_2) = \mathsf{clean}(Q_1, A \cup A_1 \cup (A' \backslash A'')) \parallel_B \mathsf{clean}(Q_2, A \cup A_2 \cup (A' \backslash A'')) = \mathsf{clean}(Q, A \cup (A' \backslash A''))$.

2. Let us denote, for $i = 1, 2$, $R_i = \mathsf{clean}(Q_i, A \cup A_i)$. Then, by Definition 2.3 it is $\mathsf{clean}(Q, A) = R_1 \parallel_B R_2$ and, by induction hypothesis, $\mathcal{U}(R_i, A'' \cup B) = \mathcal{U}(Q_i, (A \cup A_1) \cup (A'' \cup B)) = \mathcal{U}(Q_i, (A \cup A'') \cup B)$ (since $A_i \subseteq B$). Again by induction hypothesis, it is also $\mathcal{U}(R_1, A'') \cap \mathcal{U}(R_2, A'') \cap B = \mathcal{U}(Q_1, A \cup A_1 \cup A'') \cap \mathcal{U}(Q_2, A \cup A_2 \cup A'') \cap B \subseteq \mathcal{U}(Q_1, A \cup A'') \cap \mathcal{U}(Q_2, A \cup A'') \cap B$ (this is because $A \cup A'' \subseteq (A \cup A'') \cup A_i$ implies, by Proposition A.1-3 $\mathcal{U}(Q_i, A \cup A_i \cup A'') \subseteq \mathcal{U}(Q_i, A \cup A'')$ for $i = 1, 2$). Moreover, $\alpha \in \mathcal{U}(Q_1, A \cup A_1 \cup A'') \cap \mathcal{U}(Q_2, A \cup A_2 \cup A'') \cap B$ and Proposition A.1-1 implies $\alpha \notin A \cup A''$, and hence (by Proposition A.1-2) $\alpha \in \mathcal{U}(Q_1, A \cup A'') \cap \mathcal{U}(Q_2, A \cup A'') \cap B$. This allows us to conclude that $\mathcal{U}(R_1, A'') \cap \mathcal{U}(R_2, A'') \cap B = \mathcal{U}(Q_1, A \cup A'') \cap \mathcal{U}(Q_2, A \cup A'') \cap B$. Finally, by Definition 2.2, $\mathcal{U}(\mathsf{clean}(Q, A), A'') = \mathcal{U}(R_1 \parallel_B R_2, A'') = \mathcal{U}(R_1, A'' \cup B) \cup \mathcal{U}(R_2, A'' \cup B) \cup (\mathcal{U}(R_1, A'') \cap \mathcal{U}(R_2, A'') \cap B) = \mathcal{U}(Q_1, (A \cup A'') \cup B) \cup \mathcal{U}(Q_2, (A \cup A'') \cup B) \cup \mathcal{U}(Q_1, A \cup A'') \cap \mathcal{U}(Q_2, A \cup A'') \cap B = \mathcal{U}(Q, A \cup A'')$.

Rel: $Q = Q_1[\Phi]$.

1. Let ut assume $A' \cap \mathcal{U}(Q, A'') = A' \cap \Phi(\mathcal{U}(Q_1, \Phi^{-1}(A''))) = \emptyset$ and hence $\Phi^{-1}(A') \cap \mathcal{U}(Q_1, \Phi^{-1}(A'')) = \emptyset$. By induction hypothesis $\mathsf{clean}(Q, A) = \mathsf{clean}(Q_1, \Phi^{-1}(A))[\Phi] = \mathsf{clean}(Q_1, \Phi^{-1}(A) \cup (\Phi^{-1}(A' \backslash A'')))[\Phi] = \mathsf{clean}(Q_1, \Phi^{-1}(A \cup (A' \backslash A'')))[\Phi] = \mathsf{clean}(Q, A \cup (A' \backslash A'')))$
2. Let us denote with $R_1 = \mathsf{clean}(Q_1, \Phi^{-1}(A))$. By induction hypothesis we have that $\mathcal{U}(R_1, \Phi^{-1}(A'')) = \mathcal{U}(Q_1, \Phi^{-1}(A) \cup \Phi^{-1}(A'')) = \mathcal{U}(Q_1, \Phi^{-1}(A \cup A''))$. As a consequence, $\mathcal{U}(\mathsf{clean}(Q, A), A'') = \mathcal{U}(R_1[\Phi], A'') = \Phi(\mathcal{U}(R_1, \Phi^{-1}(A''))) = \Phi(\mathcal{U}(Q_1, \Phi^{-1}(A \cup A''))) = \mathcal{U}(Q, A \cup A'')$.

Rec: $Q = \mathsf{rec}\, x.Q_1$.

1. If $A' \cap \mathcal{U}(Q, A'') = A' \cap \mathcal{U}(Q_1, A'') = \emptyset$ then, by induction hypothesis, $\mathsf{clean}(Q, A) = \mathsf{rec}\, x.\mathsf{clean}(Q_1, A) = \mathsf{rec}\, x.\mathsf{clean}(Q_1, A \cup (A' \backslash A'')) = \mathsf{clean}(Q, A \cup (A' \backslash A''))$.
2. $\mathcal{U}(\mathsf{clean}(Q, A), A'') = \mathcal{U}(\mathsf{rec} x.\mathsf{clean}(Q_1, A), A'') = \mathcal{U}(\mathsf{clean}(Q_1, A), A'') = \mathcal{U}(Q_1, A \cup A'') = \mathcal{U}(Q, A \cup A'')$.

□

Proposition A.7 Let $Q, R \in \tilde{\mathbb{P}}$, $x \in \mathcal{X}$ action-guarded in Q and $A \subseteq \mathbb{A}$. Then:

1. $\mathsf{clean}(Q\{R/x\}, A) = \mathsf{clean}(Q, A)\{R/x\}$;
2. $\mathsf{unmark}(Q\{R/x\}) = \mathsf{unmark}(Q)\{R/x\}$.

Proof: We prove only Item 1 by induction on Q (Item 2 can be proved similarly).

Nil: $Q = \mathsf{nil}$. Trivially $\mathsf{clean}(\mathsf{nil}\{R/x\}, A) = \mathsf{clean}(\mathsf{nil}, A) = \mathsf{nil}$

Var: $Q = y$. In this case x action-guarded in Q implies $x \neq y$ and $Q\{R/x\} = y$. Similar to the previous case.

Pref: $Q = \alpha.P_1$ or $Q = \underline{\alpha}.P_1$. We prove only the latter case (the former is simpler). In this case, x is action-guarded in Q and $Q\{R/x\} = \underline{\alpha}.(P_1\{R/x\})$. Assume $\alpha \in A$. Then $\mathsf{clean}(Q\{R/x\}, A) = \alpha.(P_1\{R/x\}) = (\alpha.P_1)\{R/x\} = \mathsf{clean}(Q, A)\{R/x\}$. Similarly if $\alpha \notin A$ then $\mathsf{clean}(Q\{R/x\}, A) = \underline{\alpha}.(P_1\{R/x\}) = (\underline{\alpha}.P_1)\{R/x\} = \mathsf{clean}(Q, A)\{R/x\}$.

Read: $Q = \mu \triangleright Q_1$. In this case x is action-guarded in Q if it is so in Q_1. Moreover $Q\{R/x\} = \mu \triangleright (Q_1\{R/x\})$. Let us first consider the case where $\mu = \underline{\alpha}$ and $\alpha \in A$. Then, by Definition 2.3 and induction hypothesis it is $\mathsf{clean}(Q\{R/x\}, A) = \alpha \triangleright \mathsf{clean}(Q_1\{R/x\}, A) = \alpha \triangleright (\mathsf{clean}(Q_1, A)\{R/x\}) = (\alpha \triangleright \mathsf{clean}(Q_1, A)\{R/x\}) = \mathsf{clean}(Q, A)\{R/x\}$. If either $\mu = \underline{\alpha}$ and $\alpha \notin A$ or $\mu = \alpha$ then, similarly, $\mathsf{clean}(Q\{R/x\}, A) = \mu \triangleright \mathsf{clean}(Q_1\{R/x\}, A) = \mu \triangleright (\mathsf{clean}(Q_1, A)\{R/x\}) = (\mu \triangleright \mathsf{clean}(Q_1, A)\{R/x\}) = \mathsf{clean}(Q, A)\{R/x\}$.

Sum: $Q = Q_1 + Q_2$. In such a case x is action-guarded in Q iff it is action-guarded in Q_1 and in Q_2. Moreover, $\mathsf{clean}(Q\{R/x\}, A) = \mathsf{clean}(Q_1\{R/x\} + Q_2\{R/x\}, A) = \mathsf{clean}(Q_1\{R/x\}, A) + \mathsf{clean}(Q_2\{R/x\}, A) =$ (by induction hypothesis) $(\mathsf{clean}(Q_1, A)\{R/x\}) + (\mathsf{clean}(Q_2, A)\{R/x\}) = (\mathsf{clean}(Q_1, A) + \mathsf{clean}(Q_2, A))\{R/x\} = \mathsf{clean}(Q, A)\{R/x\}$.

Par: $Q = Q_1 \parallel_B Q_2$. Assume x action-guarded in Q and, hence, in Q_1 and in Q_2. Let us denote with $A_1 = B \backslash \mathcal{U}(Q_2\{R/x\})$ and with $A_2 = B \backslash \mathcal{U}(Q_1\{R/x\})$. Then x action-guarded in Q_1, Q_2 and Proposition A.2 imply $A_1 = B \backslash \mathcal{U}(Q_2)$ and $A_2 = B \backslash \mathcal{U}(Q_1)$. By induction hypothesis we have that $\mathsf{clean}(Q\{R/x\}, A) = \mathsf{clean}(Q_1\{R/x\}, A \cup A_1) \parallel_B \mathsf{clean}(Q_2\{R/x\}, A \cup A_2) = (\mathsf{clean}(Q_1, A \cup A_1)\{R/x\}) \parallel_B (\mathsf{clean}(Q_2, A \cup A_2)\{R/x\}) =$

$(\mathsf{clean}(Q_1, A \cup A_1) \parallel_B (\mathsf{clean}(Q_2, A \cup A_2))\{R/x\} = \mathsf{clean}(Q, A)\{R/x\}$

Rel: $Q = Q_1[\Phi_u]$. Assume x action-guarded in Q and, hence, in Q_1. By induction hypothesis $\mathsf{clean}(Q\{R/x\}, A) = \mathsf{clean}((Q_1\{R/x\})[\Phi], A) = \mathsf{clean}(Q_1\{R/x\}, \Phi^{-1}(A))[\Phi] =$

$(\mathsf{clean}(Q_1, \Phi^{-1}(A))\{R/x\})[\Phi] = (\mathsf{clean}(Q_1, \Phi^{-1}(A))[\Phi])\{R/x\} = \mathsf{clean}(Q, A)\{R/x\}$.

Rec: $Q = \mathsf{rec}\, y.Q_1$. If $x = y$ then x is action-guarded in Q, $Q\{R/x\} = R$ and the statement follows easily. Now assume $x \neq y$. In this case x is action-guarded if it is action-guarded in Q_1 and $Q\{R/x\} = \mathsf{rec}\, y.(Q_1\{R/x\})$. Finally, $\mathsf{clean}(Q\{R/x\}, A) = \mathsf{rec}\, y.\mathsf{clean}(Q_1\{R/x\}, A) = \mathsf{rec}\, y.(\mathsf{clean}(Q_1, A)\{R/x\}) = (\mathsf{rec}\, y.\mathsf{clean}(Q_1, A))\{R/x\} = \mathsf{clean}(Q, A)\{R/x\}$, by induction hypothesis.

□

Proposition A.8 Let $Q, Q' \in \tilde{\mathbb{P}}$ and $X \subseteq \mathbb{A}$ such that $Q \xrightarrow{X}_r Q'$. Then:

1. $\mathcal{A}(Q', A) = \mathcal{U}(Q', A) = \mathcal{A}(Q, A)$ for every $A \subseteq \mathbb{A}_\tau$;
2. $\mathsf{unmark}(Q) = \mathsf{unmark}(Q')$;
3. $Q = P \in \tilde{\mathbb{P}}_1$ implies $\mathsf{unmark}(Q') = P$.

Proof: First, we prove, by induction on $Q \in \tilde{\mathbb{P}}$, Items 1 and 2.

Var: $Q = x$. This case is not possible since $Q \not\xrightarrow{X}_r$.

Nil: $Q = \mathsf{nil}$. $Q \xrightarrow{X}_r \mathsf{nil} = Q'$. In this case:

1. $\mathcal{A}(Q', A) = \mathcal{U}(Q', A) = \mathcal{A}(Q, A) = \emptyset$;
2. $\mathsf{unmark}(Q) = \mathsf{unmark}(Q') = \mathsf{nil}$.

Pref: $Q = \alpha.P_1$ or $Q = \underline{\alpha}.P_1$. In both cases $Q \xrightarrow{X}_r \underline{\alpha}.P_1 = Q'$.

1. $\alpha \notin A$ implies $\mathcal{A}(Q', A) = \mathcal{U}(Q', A) = \mathcal{A}(Q, A) = \{\alpha\}$. Otherwise, $\mathcal{A}(Q', A) = \mathcal{U}(Q', A) = \mathcal{A}(Q, A) = \emptyset$.
2. $\mathsf{unmark}(Q) = \mathsf{unmark}(Q') = \alpha.P_1$.

Read: $Q = \alpha \triangleright Q_1$ or $Q = \underline{\alpha} \triangleright Q_1$. In both cases $Q \xrightarrow{X}_r Q'$ if $Q_1 \xrightarrow{X}_r Q'_1$ and $Q' = \underline{\alpha} \triangleright Q'_1$. By induction hypothesis:

1. $\alpha \notin A$ implies $\mathcal{A}(Q', A) = \{\alpha\} \cup \mathcal{A}(Q'_1, A) = \{\alpha\} \cup \mathcal{U}(Q'_1, A) = \mathcal{U}(Q', A) = \{\alpha\} \cup \mathcal{A}(Q_1, A) = \mathcal{A}(Q, A)$. Otherwise, $\mathcal{A}(Q', A) = \mathcal{A}(Q'_1, A) = \mathcal{U}(Q_1, A) = \mathcal{U}(Q', A) = \mathcal{A}(Q_1, A) = \mathcal{A}(Q, A)$.
2. $\mathsf{unmark}(Q) = \alpha \triangleright \mathsf{unmark}(Q_1) = \alpha \triangleright \mathsf{unmark}(Q'_1) = \mathsf{unmark}(Q')$.

Sum: $Q = Q_1 + Q_2$. By operational semantics $Q \xrightarrow{X}_r Q'$ implies $Q_1 \xrightarrow{X}_r Q'_1$, $Q_2 \xrightarrow{X}_r Q'_2$ and $Q' = Q'_1 + Q'_2$. By induction hypothesis:

1. $\mathcal{A}(Q', A) = \mathcal{A}(Q'_1, A) \cup \mathcal{A}(Q'_2, A) = \mathcal{A}(Q_1, A) \cup \mathcal{A}(Q_2, A) = \mathcal{A}(Q, A)$. Similarly we can prove that $\mathcal{U}(Q', A) = \mathcal{A}(Q, A)$.
2. By induction hypothesis, we have that $\mathsf{unmark}(Q) = \mathsf{unmark}(Q_1) + \mathsf{unmark}(Q_2) = \mathsf{unmark}(Q'_1) + \mathsf{unmark}(Q'_2) = \mathsf{unmark}(Q')$.

Par: $Q = Q_1 \|_B Q_2$. Assume $Q \xrightarrow{X}_r Q'$. Then, by operational rules, $Q_i \xrightarrow{X_i}_r Q'_i$ for $i = 1, 2$ and $X \subseteq (B \cap (X_1 \cup X_2)) \cup ((X_1 \cap X_2) \backslash B)$ and $Q' = \mathsf{clean}(Q'_1 \|_B Q'_2)$.

1. By induction hypothesis and Proposition A.6-3, it is $\mathcal{A}(Q', A) = \mathcal{A}(Q'_1 \|_B Q'_2, A) =$
$\mathcal{A}(Q'_1, A \cup B) \cup \mathcal{A}(Q'_2, A \cup B) \cup (\mathcal{A}(Q'_1, A) \cap \mathcal{A}(Q'_2, A) \cap B) =$
$\mathcal{A}(Q_1, A \cup B) \cup \mathcal{A}(Q_2, A \cup B) \cup (\mathcal{A}(Q_1, A) \cap \mathcal{A}(Q_2, A) \cap B) = \mathcal{A}(Q, A)$.
Similarly we can prove that $\mathcal{A}(Q', A) = \mathcal{U}(Q'_1 \|_B Q'_2, A)$. We are done since, by Proposition A.6-2, it is $\mathcal{U}(Q'_1 \|_B Q'_2, A) = \mathcal{U}(\mathsf{clean}(Q'_1 \|_B Q'_2), A) = \mathcal{U}(Q', A)$.
2. By Proposition A.5-1 $\mathsf{unmark}(Q') = \mathsf{unmark}(\mathsf{clean}(Q'_1 \|_B Q'_2)) = \mathsf{unmark}(Q'_1 \|_B Q'_2)$. Moreover, by inductuction hypothesis, $\mathsf{unmark}(Q'_1 \|_B Q'_2) = \mathsf{unmark}(Q'_1) \|_B \mathsf{unmark}(Q'_2) = \mathsf{unmark}(Q_1) \|_B \mathsf{unmark}(Q_2) = \mathsf{unmark}(Q)$ and we are done.

Rel: $Q = Q_1[\Phi]$. By operational semantics $Q \xrightarrow{X}_r Q'$ implies $Q_1 \xrightarrow{X'}_r Q'_1$, with $X' = \Phi^{-1}(X \cup \{\tau\}) \backslash \{\tau\}$, and $Q' = Q'_1[\Phi]$. By induction hypothesis:

1. $\mathcal{A}(Q', A) = \Phi(\mathcal{A}(Q'_1, \Phi^{-1}(A)) = \Phi(\mathcal{A}(Q_1, \Phi^{-1}(A)) = \mathcal{A}(Q, A)$. Similarly we can prove that $\mathcal{U}(Q', A) = \mathcal{A}(Q, A)$.
2. $\mathsf{unmark}(Q) = \mathsf{unmark}(Q_1)[\Phi] = \mathsf{unmark}(Q'_1)[\Phi] = \mathsf{unmark}(Q')$.

Rec: $Q = \mathsf{rec}\, x.Q_1$. By operational semantics $Q \xrightarrow{X}_r Q'$ implies $Q_1 \xrightarrow{X}_r Q'_1$ and $Q' = \mathsf{rec}\, x.Q'_1$. By induction hypothesis:

1. $\mathcal{A}(Q', A) = \mathcal{A}(Q'_1, A) = \mathcal{A}(Q_1, A) = \mathcal{A}(Q, A)$. Similarly it is $\mathcal{U}(Q', A) = \mathcal{A}(Q, A)$.
2. $\mathsf{unmark}(Q) = \mathsf{rec}\, x.\mathsf{unmark}(Q_1) = \mathsf{rec}\, x.\mathsf{unmark}(Q'_1) = \mathsf{unmark}(Q')$.

Now we prove Item 3. Assume $Q = P \in \tilde{\mathbb{P}}_1$ and that $P \xrightarrow{X}_r Q'$. Then Item 2 and Proposition A.5-2 imply $\mathsf{unmark}(Q') = \mathsf{unmark}(P) = P$. □

Proposition A.9 Let $Q \in \tilde{\mathbb{P}}$, $A, A' \subseteq \mathbb{A}$ with $A \subseteq A'$. Then: $\mathsf{clean}(\mathsf{clean}(Q, A), A') = \mathsf{clean}(Q, A')$.

Proof: By induction on $Q \in \tilde{\mathbb{P}}$.

Nil, Var: $Q = \mathsf{nil}$ or $Q = x$. In both cases $\mathsf{clean}(\mathsf{clean}(Q, A), A') = \mathsf{clean}(Q, A') = Q$.

Pref: $Q = \alpha.P_1$ or $Q = \underline{\alpha}.P_1$. We only prove the latter case (the former case can be proved as the in the Nil-case). If $\alpha \in A \subseteq A'$ then $\mathsf{clean}(\mathsf{clean}(Q, A), A') = \mathsf{clean}(Q, A') = \alpha.P_1$. Otherwise, if $\alpha \notin A$ then $\mathsf{clean}(Q, A) = Q$ and the statement follows easily.

Read: $Q = \alpha \triangleright Q_1$ or $Q = \underline{\alpha} \triangleright Q_1$. We only prove the latter case (the former one is easier). If $\alpha \in A \subseteq A'$ then, induction hypothesis, $\mathsf{clean}(\mathsf{clean}(Q, A), A') = \mathsf{clean}(\alpha \triangleright \mathsf{clean}(Q_1, A), A') = \alpha \triangleright \mathsf{clean}(\mathsf{clean}(Q_1, A), A') = \alpha \triangleright \mathsf{clean}(Q_1, A') = \mathsf{clean}(Q, A')$.

If $\alpha \notin A$, it is $\mathsf{clean}(\mathsf{clean}(Q, A), A') = \mathsf{clean}(\underline{\alpha} \triangleright \mathsf{clean}(Q_1, A), A')$. Now, $\alpha \in A'$ implies $\mathsf{clean}(\underline{\alpha} \triangleright \mathsf{clean}(Q_1, A), A') = \alpha \triangleright \mathsf{clean}(\mathsf{clean}(Q_1, A), A') = \alpha \triangleright \mathsf{clean}(Q_1, A') = \mathsf{clean}(Q, A')$ by induction hypothesis. Finally, if $\alpha \notin A'$ we have that $\mathsf{clean}(\underline{\alpha} \triangleright \mathsf{clean}(Q_1, A), A') = \underline{\alpha} \triangleright \mathsf{clean}(\mathsf{clean}(Q_1, A), A') = \underline{\alpha} \triangleright \mathsf{clean}(Q_1, A') = \mathsf{clean}(Q, A')$, again by induction hypothesis.

Sum: $Q = Q_1 + Q_2$. In this a case the statement follows easily by induction hypothesis.

Par: $Q = Q_1 \parallel_B Q_2$. Let us denote with $A_1 = (\mathcal{U}(Q_1) \backslash \mathcal{U}(Q_2)) \cap B \subseteq B \backslash \mathcal{U}(Q_2)$. Moreover, if $\alpha \in (B \backslash \mathcal{U}(Q_2)) \backslash A_1$ then $\alpha \notin \mathcal{U}(Q_1)$. Thus, $B \backslash \mathcal{U}(Q_2) = A_1 \cup ((B \backslash \mathcal{U}(Q_2)) \backslash A_1)$ where $((B \backslash \mathcal{U}(Q_2)) \backslash A_1) \cap \mathcal{U}(Q_1) = \emptyset$. Then, by proposition A.6-1., $Q_1' = \mathsf{clean}(Q_1, A \cup A_1) = \mathsf{clean}(Q_1, A \cup A_1 \cup ((B \backslash \mathcal{U}(Q_2)) \backslash A_1)) = \mathsf{clean}(Q_1, A \cup (B \backslash \mathcal{U}(Q_2)))$. Similarly we can prove that $Q_2' = \mathsf{clean}(Q_2, A \cup A_2) = \mathsf{clean}(Q_1, A \cup (B \backslash \mathcal{U}(Q_1)))$ where $A_2 = (\mathcal{U}(Q_2) \backslash \mathcal{U}(Q_1)) \cap B \subseteq B \backslash \mathcal{U}(Q_1)$. By Definition 2.3, it is $\mathsf{clean}(Q, A) = Q_1' \parallel_B Q_2'$.

Now let $A_1' = B \backslash \mathcal{U}(Q_2')$ and $A_2' = B \backslash \mathcal{U}(Q_1')$ Since, by Propositions A.6-2. and A.1-4., $\mathcal{U}(Q_1') = \mathcal{U}(Q_1, A \cup A_1) \subseteq \mathcal{U}(Q_1)$ and $\mathcal{U}(Q_2') = \mathcal{U}(Q_2, A \cup A_2) \subseteq \mathcal{U}(Q_2)$, it is $A_1 \subseteq B \backslash \mathcal{U}(Q_2) \subseteq B \backslash \mathcal{U}(Q_2') = A_1'$ and similarly $A_2 \subseteq A_2'$. Hence $A \cup A_i \subseteq A' \cup A_i'$ for $i = 1, 2$ and, by induction hypothesis, $\mathsf{clean}(Q_i', A' \cup A_i') = \mathsf{clean}(\mathsf{clean}(Q_i, A \cup A_i), A' \cup A_i') = \mathsf{clean}(Q_i, A' \cup A_i')$ for $i = 1, 2$. By Definition 2.3 we have that $\mathsf{clean}(\mathsf{clean}(Q, A), A') = \mathsf{clean}(Q_1' \parallel_B Q_2', A') = \mathsf{clean}(Q_1', A' \cup A_1') \parallel_B \mathsf{clean}(Q_2', A' \cup A_2') = \mathsf{clean}(Q_1, A' \cup A_1') \parallel_B \mathsf{clean}(Q_2, A' \cup A_2')$.

Due to the fact that (exactly as above) $\mathsf{clean}(Q_1, A' \cup A_1) = \mathsf{clean}(Q_1, A' \cup (B \backslash \mathcal{U}(Q_2)))$ and $\mathsf{clean}(Q_2, A' \cup A_2) = \mathsf{clean}(Q_2, A' \cup (B \backslash \mathcal{U}(Q_1)))$, again by By Definition 2.3, it still remains to prove that $\mathsf{clean}(Q_1, A' \cup A_1') = \mathsf{clean}(Q_1, A' \cup A_1)$ and, similarly, that $\mathsf{clean}(Q_2, A' \cup A_2') = \mathsf{clean}(Q_2, A' \cup A_2)$.

To this aim, let us consider a given $\alpha \in A_1' \backslash A_1$. $\alpha \in A_1'$ implies $\alpha \in B$ and $\alpha \notin \mathcal{U}(Q_2')$. So, $\alpha \in B$ and $\alpha \notin A_1$ implies either (1) $\alpha \notin \mathcal{U}(Q_1)$ or (2) both $\alpha \in \mathcal{U}(Q_1)$ and $\alpha \in \mathcal{U}(Q_2)$. In this latter case $\alpha \in \mathcal{U}(Q_2)$ and $\alpha \notin \mathcal{U}(Q_2') = \mathcal{U}(Q_2, A \cup A_2) = \mathcal{U}(Q_2) \backslash (A \cup A_2)$ implies $\alpha \in A \cup A_2$ and hence $\alpha \in A$ (this is because $\alpha \in \mathcal{U}(Q_1)$ and $\alpha \in \mathcal{U}(Q_2)$ implies $\alpha \notin A_2$).

Summing up, $\alpha \in A_1' \backslash A_1$ implies either (1) $\alpha \notin \mathcal{U}(Q_1)$ or (2) $\alpha \in A$. As a consequence, $A_1' = A_1 \cup \{\alpha \in (A_1' \backslash A_1) \mid \alpha \notin \mathcal{U}(Q_1)\} \cup \{\alpha \in (A_1' \backslash A_1) \mid \alpha \in A\}$ and, since $A \subseteq A'$, $A' \cup A_1' = A' \cup A_1 \cup \{\alpha \in (A_1' \backslash A_1) \mid \alpha \notin \mathcal{U}(Q_1)\}$. Finally, since all actions in $\{\alpha \in (A_1' \backslash A_1) \mid \alpha \notin \mathcal{U}(Q_1)\}$ are sure not urgent in Q_1, again by Proposition A.6-2, we can conclude that $\mathsf{clean}(Q_1, A' \cup A_1) = \mathsf{clean}(Q_1, A' \cup A_1')$. With similar arguments we can prove that $\mathsf{clean}(Q_2, A' \cup A_2) = \mathsf{clean}(Q_2, A' \cup A_2')$ and, hence, we are done.

Rel: $Q = Q_1[\Phi]$. In this case we have that $\mathsf{clean}(Q, A) = \mathsf{clean}(Q_1, \Phi^{-1}(A))[\Phi] = Q_1'[\Phi]$ and $\mathsf{clean}(\mathsf{clean}(Q, A), A') = \mathsf{clean}(Q_1'[\Phi], A') = \mathsf{clean}(Q_1', \Phi^{-1}(A'))[\Phi]$. Since $A \subseteq A'$ implies $\Phi^{-1}(A) \subseteq \Phi^{-1}(A')$, by induction hypothesis it is $\mathsf{clean}(Q_1', \Phi^{-1}(A')) = \mathsf{clean}(Q_1, \Phi^{-1}(A'))$. Finally $\mathsf{clean}(\mathsf{clean}(Q, A), A') = \mathsf{clean}(Q_1, \Phi^{-1}(A')[\Phi] = \mathsf{clean}(Q, A')$.

Rec: $Q = \mathsf{rec}\, x.Q_1$. By induction hypothesis $\mathsf{clean}(\mathsf{clean}(Q_1, A), A') = \mathsf{clean}(Q_1, A')$. Thus it is also $\mathsf{clean}(\mathsf{clean}(Q, A), A') = \mathsf{rec}\, x.\mathsf{clean}(\mathsf{clean}(Q_1, A), A') = \mathsf{rec}\, x.\mathsf{clean}(Q_1, A') = \mathsf{clean}(Q, A')$.

□

Proposition A.10 Let $Q \in \mathsf{L}(\tilde{\mathbb{P}})$, $A \subseteq \mathbb{A}$ and $A' \subseteq \mathbb{A}_\tau$. Then:

1. $\mathsf{LE}(\mathsf{clean}(Q, A), A') = \mathsf{LE}(Q, A')$;
2. $A \subseteq A'$ implies $\mathsf{UE}(\mathsf{clean}(Q, A), A') = \mathsf{UE}(Q, A')$.

Proof: We prove, by induction on $Q \in \mathsf{L}(\tilde{\mathbb{P}})$, only the Items 2 and 3. Item 1 follows directly since cleaning inactive urgencies cannot affect the set of live events of a term Q (see Definitions 2.3 and 5.7 for more details).

Nil, Var: $Q = \mathsf{nil}_u$, $Q = x_u$. In these cases $\mathsf{UE}(\mathsf{clean}(Q, A), A') = \mathsf{UE}(Q, A') = \emptyset$.

Pref: $Q = \alpha_u.P_1$ or $Q = \underline{\alpha}_u.P_1$. We prove only the latter case (the former is similar to the previous cases). Consider the following cases:

- $\alpha \in A \subseteq A'$. $\mathsf{UE}(\mathsf{clean}(Q, A), A') = \mathsf{UE}(\alpha_u.P_1, A') = \emptyset = \mathsf{UE}(Q, A')$.
- $\alpha \notin A$. In this case $\mathsf{clean}(Q, A) = Q$ and the statement follows easily.

Read: $Q = \alpha_{u1} \rhd_u Q_1$ or $Q = \underline{\alpha}_{u1} \rhd_u Q_1$. We prove only the latter case (the former one can be proved similarly). Consider the following cases:

- $\alpha \in A \subseteq A'$. $\mathsf{UE}(\mathsf{clean}(Q, A), A') = \mathsf{UE}(\alpha_{u1} \rhd_u \mathsf{clean}(Q_1, A), A') = \mathsf{UE}(\mathsf{clean}(Q_1, A), A') =$ (by induction hypothesis) $\mathsf{UE}(Q_1, A') = \mathsf{UE}(\underline{\alpha}_{u1} \rhd_u Q_1, A') = \mathsf{UE}(Q, A')$.
- $\alpha \notin A$ and $\alpha \notin A'$. In this case, $\mathsf{UE}(\mathsf{clean}(Q, A), A') = \mathsf{UE}(\underline{\alpha}_{u1} \rhd_u \mathsf{clean}(Q_1, A), A') = \{\langle u1 \rangle\} \cup \mathsf{UE}(\mathsf{clean}(Q_1, A), A') =$ (by induction hypothesis) $\{\langle u1 \rangle\} \cup \mathsf{UE}(Q_1, A') =$ (since $\alpha \notin A'$) $\mathsf{UE}(\underline{\alpha}_{u1} \rhd_u Q_1, A') = \mathsf{UE}(Q, A')$.
- $\alpha \notin A$ and $\alpha \in A'$ can be proved as the former case.

Sum: $Q = Q_1 +_u Q_2$. Then, by induction hypothesis, $\mathsf{UE}(\mathsf{clean}(Q, A), A') = \mathsf{UE}(\mathsf{clean}(Q_1, A) +_u \mathsf{clean}(Q_2, A), A') = \mathsf{UE}(\mathsf{clean}(Q_1, A), A') \cup \mathsf{UE}(\mathsf{clean}(Q_2, A), A') = \mathsf{UE}(Q_1, A') \cup \mathsf{UE}(Q_2, A') = \mathsf{UE}(Q, A')$.

Par: $Q = Q_1 \parallel_B^u Q_2$. Let $A_1 = B \backslash \mathcal{U}(Q_2)$, $A_2 = B \backslash \mathcal{U}(Q_1)$ and $R_i = \mathsf{clean}(Q_i, A \cup A_i)$ for $i = 1, 2$. By Definitions 2.3 and 5.8, we have that $\mathsf{clean}(Q, A) = R_1 \parallel_B^u R_2 = R$ and $\mathsf{UE}(R, A') = \mathsf{UE}(R_1, A' \cup B) \cup \mathsf{UE}(R_2, A' \cup B) \cup (\cup_{\alpha \in (B \backslash A')} R_\alpha)$ where, for each $\alpha \in B \backslash A'$, we denote with $R_\alpha = \mathsf{UE}(R_1, \mathbb{A}_\tau \backslash \{\alpha\}) \times \mathsf{UE}(R_2, \mathbb{A}_\tau \backslash \{\alpha\})$. In the following, given $\alpha \in B \backslash A'$, we use Q_α to denote the set $\mathsf{UE}(Q_1, \mathbb{A}_\tau \backslash \{\alpha\}) \times \mathsf{UE}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\})$. Since $A \cup A_i \subseteq A' \cup B$, then by induction hypothesis it is $\mathsf{UE}(R_i, A' \cup B) = \mathsf{UE}(\mathsf{clean}(Q_i, A \cup A_i), A' \cup B) = \mathsf{UE}(Q_i, A' \cup B)$, for $i = 1, 2$. Thus, again due to Definition 5.8, to prove our statement it still remains to prove that $\cup_{\alpha \in (B \backslash A')} R_\alpha = \cup_{\alpha \in (B \backslash A')} Q_\alpha$. Let $\alpha \in B \backslash A'$ and consider the following possible subcases:

- $\alpha \in A_1 = B \backslash \mathcal{U}(Q_2)$. In this case $\alpha \notin \mathcal{U}(Q_2)$ and $\mathcal{U}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\}) \subseteq \mathcal{U}(Q_2)$ (see Proposition A.1-3) implies $\alpha \notin \mathcal{U}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\})$. Moreover, if there exists some β such that $\beta \in \mathcal{U}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\})$ then, by Proposition A.1-1, it is $\beta \notin \mathbb{A}_\tau \backslash \{\alpha\}$ and, hence, $\beta = \alpha$. We can conclude that $\mathcal{U}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\}) = \emptyset$ and hence (see Proposition 5.11-3) $\mathsf{UE}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\}) = \emptyset$. Thus $Q_\alpha = \mathsf{UE}(Q_1, \mathbb{A}_\tau \backslash \{\alpha\}) \times \mathsf{UE}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\}) = \emptyset$.

  Moreover, since $\mathcal{U}(R_2) = \mathcal{U}(\mathsf{clean}(Q_2, A \cup A_2)) = \mathcal{U}(Q_2, A \cup A_2) \subseteq \mathcal{U}(Q_2)$ (see Propositions A.6-2 and A.1-3) it is also $\alpha \notin \mathcal{U}(R_2)$. As above we have that $\mathcal{U}(R_2, \mathbb{A}_\tau \backslash \{\alpha\}) = \emptyset$, $\mathsf{UE}(R_2, \mathbb{A}_\tau \backslash \{\alpha\}) = \emptyset$ and $R_\alpha = \mathsf{UE}(R_1, \mathbb{A}_\tau \backslash \{\alpha\}) \times \mathsf{UE}(R_2, \mathbb{A}_\tau \backslash \{\alpha\}) = \emptyset$.
- $\alpha \in A_2$. As in the previous case we can prove that $Q_\alpha = R_\alpha = \emptyset$.

- $\alpha \notin A_1$ and $\alpha \notin A_2$. In this case $\alpha \in B \backslash A'$ implies $\alpha \notin A \subseteq A'$ and, hence, $\alpha \notin A \cup A_i$ for $i = 1, 2$. Thus $A \cup A_i \subseteq \mathbb{A}_\tau \backslash \{\alpha\}$ and, by induction hypothesis, $\mathsf{UE}(R_i, \mathbb{A}_\tau \backslash \{\alpha\}) = \mathsf{UE}(Q_i, \mathbb{A}_\tau \backslash \{\alpha\})$, for $i = 1, 2$. Hence $R_\alpha = Q_\alpha$.

This allows us to conclude that $\cup_{\alpha \in B \backslash A'} R_\alpha = \cup_{\alpha \in B \backslash A'} Q_\alpha$ and, hence, we are done.

Rel: $Q = Q_1[\Phi_u]$. Let us assume $A \subseteq A'$ and, hence, $\Phi^{-1}(A) \subseteq \Phi^{-1}(A')$. By induction hypothesis $\mathsf{UE}(\mathsf{clean}(Q, A), A') = \mathsf{UE}(\mathsf{clean}(Q_1, \Phi^{-1}(A))[\Phi_u], A') = \mathsf{UE}(\mathsf{clean}(Q_1, \Phi^{-1}(A)), \Phi^{-1}(A')) = \mathsf{UE}(Q_1, \Phi^{-1}(A')) = \mathsf{UE}(Q, A')$.

Rec: $Q = \mathsf{rec}\, x_u.Q_1$. By induction hypothesis we have that $\mathsf{UE}(\mathsf{clean}(Q, A), A') =$

$\mathsf{UE}(\mathsf{rec}\, x_u.\mathsf{clean}(Q_1, A), A') = \mathsf{UE}(\mathsf{clean}(Q_1, A), A') = \mathsf{UE}(Q_1, A') = \mathsf{UE}(Q, A')$.

□

Lemma A.11 Let $Q \in \mathsf{L}(\tilde{\mathbb{P}})$. Then $\mathsf{UE}(\mathsf{clean}(Q), A) = \mathsf{UE}(Q, A)$ for every $A \subseteq \mathbb{A}_\tau$.

Proof: This follows directly from Proposition A.10-2 □

B A Proof of Proposition 2.10

This section is devoted to proving Proposition 2.10.

Proposition 2.10 Let $Q, \in \tilde{\mathbb{P}}$ be a clean process. Then:

1. $Q \xrightarrow{\mu} Q'$ and $Q \xrightarrow{X}_r Q'$ implies Q' clean;
2. $\mathcal{U}(Q) = \emptyset$ implies $Q \in \tilde{\mathbb{P}}_1$.

Proof: We first prove Item 2. By Proposition A.5-3., $\mathcal{U}(Q) = \emptyset$ implies $Q = \mathsf{clean}(Q) = \mathsf{unmark}(Q)$ and, hence $Q \in \tilde{\mathbb{P}}_1$.

Let us prove Item 1. If $Q \overset{\mu}{\rightsquigarrow} Q'$ then $Q' = Q$ implies trivially Q' clean. So, it remains to prove our statement when (i) $Q \overset{\mu}{\mapsto} Q'$ and (ii) $Q \xrightarrow{X}_r Q'$. In the former case we proceed by induction on the derivation $Q \overset{\mu}{\mapsto} Q'$, in the latter one by induction on Q.

Var: $Q = x$. This case is not possible since (i) $Q \overset{\mu}{\not\mapsto}$ and (ii) $Q \overset{X}{\not\longrightarrow}_r$.

Nil: $Q = \mathsf{nil}$, $Q = x$. (i) $Q \overset{\mu}{\not\mapsto}$ and (ii) $Q \xrightarrow{X}_r \mathsf{nil}$ with nil clean.

Pref: $Q = \alpha.P_1$ or $Q = \underline{\alpha}.P_1$ with $P_1 \in \tilde{\mathbb{P}}_1$. In both cases Q is clean. Moreover:

(i) $Q \overset{\mu}{\mapsto} Q'$ implies $\mu = \alpha$ and $Q = P_1 \in \tilde{\mathbb{P}}_1$ and, hence, clean.

(ii) $Q \xrightarrow{X}_r \underline{\alpha}.P_1 = Q'$ with $\mathsf{clean}(\underline{\alpha}.P_1) = \underline{\alpha}.P_1$ and, hence Q' clean.

We only prove the latter case (since the former one follows easily). Assume $\mathcal{U}(Q, A) = \emptyset$. Then, by Definition 2.2, we have $\alpha \in A$ and hence $\mathsf{clean}(Q, A) = \alpha.P_1 = \mathsf{unmark}(Q)$.

Read: $Q = \alpha \triangleright Q_1$ or $Q = \underline{\alpha} \triangleright Q_1$. In both cases Q clean implies Q_1 clean.

(i) $Q \overset{\mu}{\mapsto} Q'$ implies $Q_1 \overset{\mu}{\mapsto} Q'$ and the statement follows by induction hypothesis.

(ii) $Q \xrightarrow{X}_r \underline{\alpha} \triangleright Q'_1 = Q'$ if $Q_1 \xrightarrow{X}_r Q'_1$. By induction hypothesis $\mathsf{clean}(Q') = \underline{\alpha} \triangleright \mathsf{clean}(Q'_1) = \underline{\alpha} \triangleright Q'_1 = Q'$, that is Q' is clean.

Sum: $Q = Q_1 + Q_2$. In this case if Q is clean then both Q_1 and Q_2 are clean.

(i) $Q \overset{\mu}{\mapsto} Q'$ if either $Q_1 \overset{\mu}{\mapsto} Q'$ or $Q_2 \overset{\mu}{\mapsto} Q'$. In both cases, the statement follows by induction hypothesis.

(ii) $Q \xrightarrow{X}_r Q'_1 + Q'_2 = Q'$ if $Q_i \xrightarrow{X}_r Q'_i$ for $i = 1, 2$. By induction hypothesis Q'_1 and Q_2 are clean. Thus, $\mathsf{clean}(Q') = \mathsf{clean}(Q'_1) + \mathsf{clean}(Q'_2) = Q'_1 + Q'_2 = Q'$. Again Q' is clean.

Par: $Q = Q_1 \,\|_B\, Q_2$. By operational rules, if (i) $Q \overset{\mu}{\mapsto} Q'$ or (ii) $Q \xrightarrow{X}_r Q'$ then there exists a proper R such that $Q' = \mathsf{clean}(R)$. By Proposition A.9, $\mathsf{clean}(Q') = \mathsf{clean}(\mathsf{clean}(R)) = \mathsf{clean}(R) = Q'$. Also in this case Q' is clean.

Rel: $Q = Q_1[\Phi]$. Similar to the Read-case

Rec: $Q = \mathsf{rec}\, x.Q_1$. Again Q clean implies Q_1 clean.

(i) Let $P_1 = \mathsf{unmark}(Q_1)$, $P = \mathsf{rec}\, x.P_1$ and $S = Q_1\{P/x\}$. x action-guarded in Q_1 and Q_1 clean implies $\mathsf{clean}(S) = \mathsf{clean}(Q_1)\{P/x\} = Q_1\{P/x\} = S$ (see Proposition A.7-1.). By operational rules, $Q \overset{\mu}{\mapsto} Q'$ if $S \overset{\mu}{\mapsto} Q'$ and, since S is clean, the statement follows by induction hypothesis.

(ii) $Q \xrightarrow{X}_r \mathsf{rec}\, x.Q'_1 = Q'$ if $Q_1 \xrightarrow{X}_r Q'_1$. By induction hypothesis Q'_1 is clean. Thus, $\mathsf{clean}(Q') = \mathsf{rec}\, x.\mathsf{clean}(Q'_1) = \mathsf{rec}\, x.Q'_1 = Q'$. Again Q' is clean.

□

C Proofs of Proposition 4.2 and Theorem 4.3

As usual, some preliminary results are needed.

Proposition C.1 $[\![Q]\!] \overset{\alpha}{\not\leadsto}$ for all read-guarded $Q \in \tilde{\mathbb{S}}$.

Proof: By induction on $Q \in \tilde{\mathbb{S}}$

Nil, Var, Pref: $Q = \mathsf{nil}$, $Q = x$ or $Q = \mu.P$. By Definition 4.1, it is $[\![Q]\!] = \mathsf{nil}$, $[\![Q]\!] = x$ or $[\![Q]\!] = \mu.[\![P]\!]$. In all of such cases $[\![Q]\!] \overset{\alpha}{\not\leadsto}$.

Read: $Q = \{\mu_1, \dots, \mu_n\} \triangleright Q_1$. This case is not possible since Q is not read-guarded.

Sum: $Q = Q_1 + Q_2$ with both Q_1 and Q_2 read-guarded. By induction hypothesis, $[\![Q_1]\!] \overset{\alpha}{\not\leadsto}$ and $[\![Q_2]\!] \overset{\alpha}{\not\leadsto}$. Thus, by operational rules, $[\![Q]\!] = [\![Q_1]\!] + [\![Q_2]\!] \overset{\alpha}{\not\leadsto}$.

Par: $Q = Q_1 \parallel_B Q_2$ with both Q_1 and Q_2 read-guarded. The statement follows easily by induction hypothesis as in the previous case.

Rel: $Q = Q_1[\Phi]$ with Q_1 read-guarded. By induction hypothesis, $[\![Q_1]\!] \overset{\alpha}{\not\leadsto}$. By operational rules we can conclude that $[\![Q]\!] \overset{\alpha}{\not\leadsto}$.

Rec: Similar to the previous case.

□

Proposition C.2 Let $Q, R \in \tilde{\mathbb{S}}$ and $x \in \mathcal{X}$ action-guarded in Q. If Q is read-guarded, then so is $Q\{R/x\}$.

Proof: By induction on $Q \in \tilde{\mathbb{S}}$.

Nil: $Q = \mathsf{nil}$. Then $Q = Q\{R/x\} = \mathsf{nil}$ is read-guarded.

Var: $Q = y$. If x is action-guarded in Q, then $x \neq y$ and $Q\{R/x\} = y$ is trivially read-guarded.

Pref: $Q = \mu.P$. In this case it is $Q\{R/x\} = \mu.(P\{R/x\})$ and, hence, read-guarded.

Read: $Q = \{\mu_1, \dots, \mu_n\} \triangleright Q_1$. This case is not possible, since Q is not read-guarded.

Sum: $Q = Q_1 + Q_2$ with Q_i read-guarded and x action-guarded in Q_i, for $i = 1, 2$. By induction hypothesis, $Q_1\{R/x\}$ and $Q_2\{R/x\}$ are read-guarded. Hence, $Q\{R/x\} = Q_1\{R/x\} + Q_2\{R/x\}$ is read-guarded.

Par: $Q = Q_1 \parallel_B Q_2$ with Q_i read-guarded and x action-guarded in Q_i, for $i = 1, 2$. The statement follows by induction hypothesis as in the previous case.

Rel: $Q = Q_1[\Phi]$ with Q_1 read-guarded and x action-guarded in Q_1. By induction hypothesis, $Q_1\{R/x\}$ and hence $Q\{R/x\} = (Q_1\{R/x\})[\Phi]$ is read-guarded.

Rec: $Q = \mathsf{rec}\, y.Q_1$ with Q_1 read-guarded and x action-guarded in Q_1. We consider two subcases:

- $x = y$. In this case $Q\{R/x\} = Q$ is read-guarded.
- $x \neq y$. By induction hypothesis we have $Q_1\{R/x\}$ and hence $Q\{R/x\} = \mathsf{rec}\, y.Q_1\{R/x\}$ read-guarded.

□

Proposition C.3 Let Q' be a subterm of $Q \in \tilde{\mathbb{S}}$. If Q is read-proper, then so is Q'; if x is read-guarded in Q, then it is also read-guarded in Q'.

Proof: It can be easily proven by induction on $Q \in \tilde{\mathbb{S}}$. □

Proposition C.4 Let $Q, R \in \tilde{\mathbb{S}}$ be read-proper and $x \in \mathcal{X}$ be read-guarded in Q. Then: $Q\{R/x\}$ is read-proper.

Proof: By induction on $Q \in \tilde{\mathbb{S}}$. We repeatedly apply Proposition C.3 in the following without mentioning this explicitly.

Nil: $Q = \mathsf{nil}$. $Q\{R/x\} = Q = \mathsf{nil}$ and the statement follows easily.

Var: $Q = y$. If $x \neq y$ then $Q\{R/x\} = y$. If $x = y$ then $Q\{R/x\} = R$. In both cases $Q\{R/x\}$ is read-proper.

Pref: $Q = \mu.P$ with P read-proper and x read-guarded in P. By induction hypothesis, we have $P\{R/x\}$ and $Q\{R/x\} = \mu.(P\{R/x\})$ read-proper.

Read: $Q = \{\mu_1, \ldots, \mu_n\} \triangleright Q_1$ with Q_1 both read-guarded and read-proper. In this case, x read-guarded in Q implies that x is both action-guarded and read-guarded in Q_1. By induction hypothesis, Q_1, R read-proper and x read-guarded in Q_1 implies $Q_1\{R/x\}$ read-proper. Moreover, Q_1 read-guarded and x action-guarded in Q_1 implies $Q_1\{R/x\}$ read-guarded (by Proposition C.2). By definition of read-properness we can conclude that $Q\{R/x\}$ is both read-guarded and read-proper.

Sum: $Q = Q_1 + Q_2$ with Q_1, Q_2 read-proper and x read-guarded in Q_1 and in Q_2. By induction hypothesis, it is $Q_1\{R/x\}$, $Q_2\{R/x\}$ read-proper. Hence $Q\{R/x\} = Q_1\{R/x\} + Q_2\{R/x\}$ is read-proper.

Par: $Q = Q_1 \|_B Q_2$ with Q_1, Q_2 read-proper and x read-guarded in Q_1 and in Q_2. The statement follows by induction hypothesis as in the previous case.

Rel: $Q = Q_1[\Phi]$ with Q_1 read-proper and x read-guarded in Q_1. By induction hypothesis, we have $Q\{R/x\} = (Q_1\{R/x\})[\Phi]$ read-proper.

Rec: $Q = \mathsf{rec}\, y.Q_1$ with Q_1 read-proper and x read-guarded in Q_1. We distinguish two possible subcases. If $x = y$ then $Q\{R/x\} = Q$ which is trivially read-proper. Otherwise, if $x \neq y$, then $Q\{R/x\} = \mathsf{rec}\, y.(Q_1\{R/x\})$ with $Q_1\{R/x\}$ read-proper by induction hypothesis. Again, we can conclude that $Q\{R/x\}$ is read-proper.

□

Proposition C.5 Let $Q \in \tilde{\mathbb{S}}$ and $A \subseteq \mathbb{A}$. Then:

1. $\mathcal{U}([\![Q]\!], A) = \mathcal{U}(Q, A)$;
2. $\mathsf{unmark}([\![Q]\!]) = [\![\mathsf{unmark}(Q)]\!]$;
3. $\mathsf{clean}([\![Q]\!], A) = [\![\mathsf{clean}(Q, A)]\!]$.

Proof: Item 1 follows easily by Definitions 2.2, 3.2 and 4.1. The proof of Item 2 is similar to that of Item 3. So, we only prove the latter item by induction on $Q \in \tilde{\mathbb{S}}$.

Nil, Var: $Q = \mathsf{nil}$ or $Q = x$. In both cases, $\mathsf{clean}([\![Q]\!], A) = [\![\mathsf{clean}(Q, A)]\!] = Q$.

Pref: $Q = \alpha.P$ or $Q = \underline{\alpha}.P$. We prove only the latter case (the former one is easier). Consider the following possible cases:

- $\alpha \in A$. $\mathsf{clean}([\![Q]\!], A) = \mathsf{clean}(\underline{\alpha}.[\![P]\!], A) = \alpha.[\![P]\!] = [\![\alpha.P]\!] = [\![\mathsf{clean}(\underline{\alpha}.P, A)]\!]$.
- $\alpha \notin A$. $\mathsf{clean}([\![Q]\!], A) = \mathsf{clean}(\underline{\alpha}.[\![P]\!], A) = \underline{\alpha}.[\![P]\!] = [\![\underline{\alpha}.P]\!] = [\![\mathsf{clean}(\underline{\alpha}.P, A)]\!]$.

Read: $Q = \{\mu_1, \ldots, \mu_n\} \triangleright Q_1$. Let $\{\nu_1, \ldots, \nu_n\}$ the subset of $\mathbb{A}_\tau \cup \underline{\mathbb{A}}_\tau$ such that for each $i \in [1, n]$, $\nu_i = \alpha$ if $\mu = \underline{\alpha}$ and $\alpha \in A$; $\nu_i = \mu_i$ otherwise. By Definition 4.1, it is $\mathsf{clean}([\![Q]\!], A) = \mathsf{clean}(\mu_1 \triangleright \ldots \triangleright \mu_n \triangleright [\![Q_1]\!], A) = \nu_1 \triangleright \ldots \triangleright \nu_n \triangleright \mathsf{clean}([\![Q_1]\!], A) = \nu_1 \triangleright \ldots \triangleright \nu_n \triangleright [\![\mathsf{clean}(Q_1, A)]\!]$ (by induction hypothesis) $= [\![\{\nu_1, \ldots, \nu_n\} \triangleright \mathsf{clean}(Q_1, A)]\!] = [\![\mathsf{clean}(Q, A)]\!]$.

Sum: $Q = Q_1 + Q_2$. By Definitions 3.3 and 4.1, we have that $\mathsf{clean}([\![Q]\!], A) = \mathsf{clean}([\![Q_1]\!] + [\![Q_2]\!], A) = \mathsf{clean}([\![Q_1]\!], A) + \mathsf{clean}([\![Q_2]\!], A)$ By induction hypothesis it is $\mathsf{clean}([\![Q_i]\!], A) = [\![\mathsf{clean}(Q_i, A)]\!]$, for $i = 1, 2$. Finally: $\mathsf{clean}([\![Q]\!], A) = \mathsf{clean}([\![Q_1]\!], A) + \mathsf{clean}([\![Q_2]\!], A) = [\![\mathsf{clean}(Q_1, A)]\!] + [\![\mathsf{clean}(Q_2, A)]\!] = [\![\mathsf{clean}(Q_1 + Q_2, A)]\!] = [\![\mathsf{clean}(Q, A)]\!]$.

Par: $Q = Q_1 \parallel_B Q_2$. Let $A_1 = B \backslash \mathcal{U}([\![Q_2]\!]) = B \backslash \mathcal{U}(Q_2)$ and $A_2 = B \backslash \mathcal{U}([\![Q_1]\!]) = B \backslash \mathcal{U}(Q_1)$ by Item 1. By Definitions 3.3 and 4.1, we have that $\mathsf{clean}([\![Q]\!], A) = \mathsf{clean}([\![Q_1]\!] \parallel_B [\![Q_2]\!], A) = \mathsf{clean}([\![Q_1]\!], A \cup A_1) \parallel_B \mathsf{clean}([\![Q_2]\!], A \cup A_2) = [\![\mathsf{clean}(Q_1, A \cup A_1)]\!] \parallel_B [\![\mathsf{clean}(Q_2, A \cup A_2)]\!] = [\![\mathsf{clean}(Q_1, A \cup A_1) \parallel_B \mathsf{clean}(Q_2, A \cup A_2)]\!] = [\![\mathsf{clean}(Q, A)]\!]$ by induction hypothesis.

Rel: $Q = Q_1[\Phi]$. In this case, again by Definitions 3.3 and 4.1, $\mathsf{clean}([\![Q]\!], A) = \mathsf{clean}([\![Q_1]\!][\Phi], A) = \mathsf{clean}([\![Q_1]\!], \Phi^{-1}(A))[\Phi] = [\![\mathsf{clean}(Q_1, \Phi^{-1}(A))]\!][\Phi] = [\![\mathsf{clean}(Q_1, \Phi^{-1}(A))[\Phi]]\!] = [\![\mathsf{clean}(Q, A)]\!]$.

Rec: $Q = \mathsf{rec}\ x.Q_1$. By induction hypothesis (and by Definitions 3.3 and 4.1), $\mathsf{clean}([\![Q]\!], A) = \mathsf{clean}(\mathsf{rec}\ x.[\![Q_1]\!], A) = \mathsf{rec}\ x.\mathsf{clean}([\![Q_1]\!], A) = \mathsf{rec}\ x.[\![\mathsf{clean}(Q_1, A)]\!] = [\![\mathsf{rec}\ x.\mathsf{clean}(Q_1, A)]\!] = [\![\mathsf{clean}(Q, A)]\!]$.

□

Proposition 4.2 Let $Q \in \tilde{\mathbb{S}}$ be read-proper and assume $Q \xrightarrow{\alpha} Q'$ ($Q \xrightarrow{X}_r Q'$). Then Q' is read-proper.

Proof: If $Q \xrightarrow{X}_r Q'$ then $\mathsf{unmark}(Q) = \mathsf{unmark}(Q')$ (Q' is Q with all activated actions marked as urgent) and we are done since Q read-proper implies Q' read-proper. We prove that if $Q \xrightarrow{\alpha} Q'$ then Q' is read proper by induction on the length of the derivation $Q \xrightarrow{\alpha} Q'$.

Nil, Var: $Q = \mathsf{nil}$ or $Q = x$. These cases are not possible since $Q \overset{\alpha}{\not\rightarrow}$ for any α.

Pref: $Q = \mu.P$ with P read-proper. By the operational semantics, $Q \xrightarrow{\alpha} Q'$ if $\mu \in \{\alpha, \underline{\alpha}\}$ and $Q = P$ (and, hence, read-proper).

Read: $Q = \{\mu_1, \ldots, \mu_n\} \triangleright Q_1$ with Q_1 read-proper. If $Q \xrightarrow{\alpha} \{\mu_1, \ldots, \mu_n\} \triangleright Q = Q'$ because there exists some $\mu_i \in \{\alpha, \underline{\alpha}\}$, then trivially Q' is read-proper. Otherwise, $Q \xrightarrow{\alpha} Q'$ because of $Q_1 \xrightarrow{\alpha} Q'$ and, by induction hypothesis, Q' read-proper.

Sum: $Q = Q_1 + Q_2$ with Q_1 and Q_2 read-proper. In this case $Q \xrightarrow{\alpha} Q'$ if either $Q_1 \xrightarrow{\alpha} Q'$ or $Q_2 \xrightarrow{\alpha} Q'$. In both cases, by induction hypothesis, Q' is read-proper.

Par: $Q = Q_1 \parallel_B Q_2$ with Q_1, Q_2 read-proper. Assume $Q \xrightarrow{\alpha} Q'$ and consider the following possible subcases:

- $\alpha \notin B$, $Q_1 \xrightarrow{\alpha} Q_1'$ and $Q' = \mathsf{clean}(Q_1' \parallel_B Q_2)$. Q_1 read-proper and $Q_1 \xrightarrow{\alpha} Q_1'$ implies by induction hypothesis Q_1' read-proper. Finally, Q_1' and Q_2 read-proper implies $Q_1' \parallel_B Q_2$ and hence $\mathsf{clean}(Q_1' \parallel_B Q_2)$ read-proper (as above, we have that $\mathsf{unmark}(\mathsf{clean}(Q_1' \parallel_B Q_2)) = \mathsf{unmark}(Q_1' \parallel_B Q_2)$).
- either $\alpha \notin B$, $Q_2 \xrightarrow{\alpha} Q_2'$ and $Q' = \mathsf{clean}(Q_1 \parallel_B Q_2')$ or $\alpha \in B$, $Q_i \xrightarrow{\alpha} Q_i'$ for $i = 1, 2$, and $Q' = \mathsf{clean}(Q_1' \parallel_B Q_2')$. These cases can be proven as the previous one.

Rel: $Q = Q_1[\Phi]$ with Q_1 read-proper. In this case, $Q \xrightarrow{\alpha} Q_1'[\Phi] = Q'$ if there exists $\beta \in \Phi^{-1}(\alpha)$ such that $Q_1 \xrightarrow{\alpha} Q_1'$. By induction hypothesis, $Q' = Q_1'[\Phi]$ is read-proper.

Rec: $Q = \mathsf{rec}\, x.Q_1$ with Q_1 read-proper. In this case $Q \xrightarrow{\alpha} Q'$ if $Q\{\mathsf{rec}\, x.\mathsf{unmark}(Q_1)/x\} \xrightarrow{\alpha} Q'$. Moreover, Q read-proper implies Q_1 and $\mathsf{rec}x.\mathsf{unmark}(Q_1)$ read-proper. Then, x read-guarded in Q_1 and Proposition C.4 imply $Q\{\mathsf{rec}\, x.\mathsf{unmark}(Q_1)/x\}$ and (by induction hypothesis) Q' read-proper.

$\square$

We subdivide Theorem 4.3 into the following two propositions, which we prove separately.

Proposition C.6 For all read-proper $Q \in \tilde{\mathbb{S}}$:

1. $Q \xrightarrow{\alpha} Q'$ implies $[\![Q]\!] \xrightarrow{\alpha} [\![Q']\!]$;
2. $[\![Q]\!] \xrightarrow{\alpha} Q''$ implies $Q \xrightarrow{\alpha} Q'$ with $[\![Q']\!] = Q''$.

Proof: We prove these statements by induction on the length of derivations $Q \xrightarrow{\alpha} Q'$ and $[\![Q]\!] \xrightarrow{\alpha} Q''$. We proceed by a case analysis on the structure of Q.

Nil, Var: $Q = [\![Q]\!] = \mathsf{nil}$ or $Q = [\![Q]\!] = x$. Not possible since $Q \overset{\alpha}{\not\rightarrow}$ for any α.

Pref: $Q = \mu.P$ and $[\![Q]\!] = \mu.[\![P]\!]$.

1. If $Q \xrightarrow{\alpha} Q'$ then $\mu \in \{\alpha, \underline{\alpha}\}$ and $Q' = P$. Moreover, $[\![Q]\!] = \mu.[\![P]\!] \xrightarrow{\alpha} [\![P]\!]$.
2. Similar to the previous one.

Read: $Q = A \triangleright Q_1$ with $A = \{\mu_1, \ldots, \mu_n\}$. In this case $[\![Q]\!] = \mu_1 \triangleright \ldots \triangleright \mu_n \triangleright [\![Q_1]\!]$.

1. If $Q \xrightarrow{\alpha} A \triangleright Q_1$ then there exists some $\mu_i \in A$ such that $\mu_i \in \{\alpha, \underline{\alpha}\}$. So, by rule $\textsc{Read}_{r_1}$, $\mu_i \triangleright \ldots \triangleright \mu_n \triangleright [\![Q_1]\!] \overset{\alpha}{\rightsquigarrow} \mu_i \triangleright \ldots \triangleright \mu_n \triangleright [\![Q_1]\!]$ and, by repeated applications of rule $\textsc{Read}_{r_2}$, $[\![Q]\!] = \mu_1 \triangleright \ldots \mu_i \triangleright \ldots \triangleright [\![Q]\!] \overset{\alpha}{\rightsquigarrow} \mu_1 \triangleright \ldots \triangleright \mu_i \triangleright \ldots \triangleright [\![Q_1]\!]$. If $Q \xrightarrow{\alpha} Q'$ because of $Q_1 \xrightarrow{\alpha} Q'$ then $[\![Q_1]\!] \xrightarrow{\alpha} [\![Q']\!]$ by induction hypothesis. Since, Q read-proper implies Q_1 is read-guarded, we even have (by Proposition C.1) $[\![Q_1]\!] \overset{\alpha}{\mapsto} [\![Q']\!]$. We are done by repeated applications of $\textsc{Read}_o$.
2. If $[\![Q]\!] = \mu_1 \triangleright \ldots \triangleright \mu_n \triangleright [\![Q_1]\!] \xrightarrow{\alpha} Q''$ we have either $\{\alpha, \underline{\alpha}\} \cap A \neq \emptyset$ (i.e. there exists some $\mu_i \in A$ such that $\mu_i \in \{\alpha, \underline{\alpha}\}$) and $Q'' = [\![Q]\!]$ and we are done similarly to the previous case; or (again by Proposition C.1, since Q_1 is read-guarded) $[\![Q_1]\!] \overset{\alpha}{\mapsto} Q''$. Then, by induction hypothesis $Q_1 \xrightarrow{\alpha} Q'$ and, by operational rules, $Q \xrightarrow{\alpha} Q'$ with $[\![Q']\!] = Q''$.

Sum: $Q = Q_1 + Q_2$ with Q_1 and Q_2 read-proper and read-guarded.

1. Assume that $Q \xrightarrow{\alpha} Q'$ because of $Q_1 \xrightarrow{\alpha} Q'$ (if $Q \xrightarrow{\alpha} Q'$ since $Q_2 \xrightarrow{\alpha} Q'$, the proof is symmetric). By induction hypothesis, we have $[\![Q_1]\!] \xrightarrow{\alpha} [\![Q']\!]$. Moreover, since Q_1 is read-guarded, also $[\![Q_1]\!] \overset{\alpha}{\mapsto} [\![Q']\!]$ (again by Proposition C.1). Thus, $[\![Q]\!] = [\![Q_1]\!]+[\![Q_2]\!] \xrightarrow{\alpha} [\![Q']\!]$.
2. Since Q_1 and Q_2 are read-guarded, we have $Q_1 \overset{\alpha}{\not\mapsto}$ and $Q_2 \overset{\alpha}{\not\mapsto}$ again by Proposition C.1. Thus, $[\![Q]\!] = [\![Q_1]\!] + [\![Q_2]\!] \xrightarrow{\alpha} Q''$ if either $[\![Q_1]\!] \overset{\alpha}{\mapsto} Q''$ or $[\![Q_2]\!] \overset{\alpha}{\mapsto} Q''$. By induction hypothesis, it is either $Q_1 \xrightarrow{\alpha} Q'$ or $Q_2 \xrightarrow{\alpha} Q'$ with $[\![Q']\!] = Q''$. By operational rules, we have $Q \xrightarrow{\alpha} Q'$.

Par: $Q = Q_1 \|_B Q_2$ with Q_1 and Q_2 read-proper.

1. Assume $Q \xrightarrow{\alpha} Q'$ and consider the following possible subcases:
   - $\alpha \notin B$, $Q_1 \xrightarrow{\alpha} Q_1'$ and $Q' = \mathsf{clean}(Q_1' \|_B Q_2)$. If $Q_1 \xrightarrow{\alpha} Q_1'$ then, by induction hypothesis, $[\![Q_1]\!] \xrightarrow{\alpha} [\![Q_1']\!]$ and $[\![Q]\!] = [\![Q_1]\!] \|_B [\![Q_2]\!] \xrightarrow{\alpha} \mathsf{clean}([\![Q_1']\!] \|_B [\![Q_2]\!]) = \mathsf{clean}([\![Q_1' \|_B Q_2]\!]) = [\![\mathsf{clean}(Q_1' \|_B Q_2)]\!] = [\![Q']\!]$ by Proposition C.5-3.
   - either $\alpha \notin B$, $Q_2 \xrightarrow{\alpha} Q_2'$ and $Q' = \mathsf{clean}(Q_1 \|_B Q_2')$ or $\alpha \in B$, $Q_i \xrightarrow{\alpha} Q_i'$, for $i = 1, 2$, and $Q' = \mathsf{clean}(Q_1' \|_B Q_2')$. Both these cases can be proved as the previous one.
2. Assume $[\![Q]\!] = [\![Q_1]\!] \|_B [\![Q_2]\!] \xrightarrow{\alpha} Q''$ and consider the following possible subcases:
   - $\alpha \notin B$, $[\![Q_1]\!] \xrightarrow{\alpha} Q_1''$ and $Q'' = \mathsf{clean}(Q_1'' \|_B [\![Q_2]\!])$. By induction hypothesis, $[\![Q_1]\!] \xrightarrow{\alpha} Q_1''$ implies $Q_1 \xrightarrow{\alpha} Q_1'$ with $[\![Q_1']\!] = Q_1''$. Thus, $Q \xrightarrow{\alpha} \mathsf{clean}(Q_1' \|_B Q_2) = Q'$. Moreover, $Q'' = \mathsf{clean}(Q_1'' \|_B [\![Q_2]\!]) = \mathsf{clean}([\![Q_1']\!] \|_B [\![Q_2]\!]) = \mathsf{clean}([\![Q_1' \|_B Q_2]\!]) = [\![\mathsf{clean}(Q_1' \|_B Q_2)]\!] = [\![Q']\!]$, by Proposition C.5-3.
   - either $\alpha \notin B$, $[\![Q_2]\!] \xrightarrow{\alpha} Q_2''$ and $Q'' = \mathsf{clean}([\![Q_1]\!] \|_B Q_2'')$ or $\alpha \in B$, $[\![Q_i]\!] \xrightarrow{\alpha} Q_i''$, for $i = 1, 2$, and $Q'' = \mathsf{clean}(Q_1'' \|_B Q_2'')$. These cases can be proved as the previous one.

Rel: $Q = Q_1[\Phi]$ with Q_1 read-proper.

1. $Q \xrightarrow{\alpha} Q'$ if there exists $\beta \in \Phi^{-1}(\alpha)$ such that $Q_1 \xrightarrow{\beta} Q_1'$ and $Q' = Q_1'[\Phi]$. By induction hypothesis, $[\![Q_1]\!] \xrightarrow{\beta} [\![Q_1']\!]$ and, hence $[\![Q]\!] = [\![Q_1]\!][\Phi] \xrightarrow{\alpha} [\![Q_1']\!][\Phi] = [\![Q']\!]$.
2. $[\![Q]\!] = [\![Q_1]\!][\Phi] \xrightarrow{\alpha} Q_1''[\Phi] = Q''$ if there exists $\beta \in \Phi^{-1}(\alpha)$ such that $[\![Q_1]\!] \xrightarrow{\beta} Q_1''$. By induction hypothesis, $Q_1 \xrightarrow{\beta} Q_1'$ with $Q_1'' = [\![Q_1']\!]$. Thus, $Q \xrightarrow{\alpha} Q_1'[\Phi] = Q'$. Moreover, $[\![Q']\!] = [\![Q_1']\!][\Phi] = Q_1''[\Phi] = Q''$.

Rec: $Q = \mathsf{rec}\, x.Q_1$ with Q_1 read-proper. Let us define $P = \mathsf{rec}\, x.\mathsf{unmark}(Q_1)$. By an easy induction on Q_1 we can prove that $[\![Q_1\{P/x\}]\!] = [\![Q_1]\!]\{[\![P]\!]/x\}$ and thus $[\![Q_1\{P/x\}]\!] = [\![Q_1]\!]\{\mathsf{rec}\, x.[\![\mathsf{unmark}(Q_1)]\!]/x\} = [\![Q_1]\!]\{\mathsf{rec}\, x.\mathsf{unmark}([\![Q_1]\!])/x\}$ by Proposition C.5-2.

1. We have $Q \xrightarrow{\alpha} Q'$ since $Q_1\{P/x\} \xrightarrow{\alpha} Q'$. By induction hypothesis, $[\![Q_1\{P/x\}]\!] = [\![Q_1]\!]\{\mathsf{rec}\, x.\mathsf{unmark}([\![Q_1]\!])/x\} \xrightarrow{\alpha} [\![Q']\!]$. By operational rules $[\![Q]\!] = \mathsf{rec}\, x.[\![Q_1]\!] \xrightarrow{\alpha} [\![Q']\!]$.
2. By the operational rules, $[\![Q]\!] \xrightarrow{\alpha} Q''$ if $[\![Q_1]\!]\{\mathsf{rec} x.\mathsf{unmark}([\![Q_1]\!])/x\} = [\![Q_1\{P/x\}]\!] \xrightarrow{\alpha} Q''$. By induction hypothesis, we have $Q_1\{P/x\} \xrightarrow{\alpha} Q'$ with $Q'' = [\![Q']\!]$. Finally, $Q_1\{P/x\} \xrightarrow{\alpha} Q'$ implies $Q \xrightarrow{\alpha} Q'$.

□

Proposition C.7 For all read-proper $Q \in \tilde{\mathbb{S}}$:

1. $Q \xrightarrow{X}_r Q'$ implies $[\![Q]\!] \xrightarrow{X}_r [\![Q']\!]$;

2. $[\![Q]\!] \xrightarrow{X}_r Q''$ implies $Q \xrightarrow{X}_r Q'$ with $[\![Q']\!] = Q''$.

Proof: We prove these statements by induction on Q. We only prove some significant cases; the other ones follow easily by induction hypothesis.

Nil: $Q = \mathsf{nil}$. Trivial, since $Q = \mathsf{nil} = [\![Q]\!] \xrightarrow{X}_r \mathsf{nil}$.

Var: $Q = x$. Not possible since neither Q nor $[\![Q]\!]$ can perform any X-step.

Pref: $Q = \alpha.P$ or $Q = \underline{\alpha}.P$ We prove only the latter case (the former is easier).

1. $Q \xrightarrow{X}_r Q'$ implies $\alpha \notin X \cup \{\tau\}$ and $Q' = \underline{\alpha}.P$. By the operational rules, if $\alpha \notin X \cup \{\tau\}$ then $[\![Q]\!] = \underline{\alpha}.[\![P]\!] \xrightarrow{X}_r \underline{\alpha}.[\![P]\!] = [\![Q']\!]$.
2. Similar to the previous one.

Read: $Q = \{\mu_1, \ldots, \mu_n\} \triangleright Q_1$. Let $A = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{A}_\tau$ such that $\mu_i \in \{\alpha_i, \underline{\alpha_i}\}$ for each $i \in [1, n]$.

1. By operational rules, $Q \xrightarrow{X}_r \underline{\{\mu_1, \ldots, \mu_n\}} \triangleright Q'_1 = \{\underline{\alpha_1}, \ldots, \underline{\alpha_n}\} \triangleright Q'_1 = Q'$ if $Q_1 \xrightarrow{X}_r Q'_1$ and $\mathcal{U}(\{\mu_1, \ldots, \mu_n\}) \cap (X \cup \{\tau\}) = \emptyset$. By induction hypothesis, $[\![Q_1]\!] \xrightarrow{X}_r [\![Q'_1]\!]$. Moreover, $\mathcal{U}(\{\mu_1, \ldots, \mu_n\}) \cap (X \cup \{\tau\}) = \emptyset$ implies either $\mu_i = \alpha_i$ or $\mu_i = \underline{\alpha_i}$ and $\alpha_i \notin X \cup \{\tau\}$ for each $i \in [1, n]$. Thus, by repeated applications of rules $\textsc{Read}_{t1}$ and $\textsc{Read}_{t2}$, $[\![Q]\!] = \mu_1 \triangleright \ldots \triangleright \mu_n \triangleright [\![Q_1]\!] \xrightarrow{X}_r \underline{\alpha_1} \triangleright \ldots \triangleright \underline{\alpha_n} \triangleright [\![Q'_1]\!] = [\![Q']\!]$.
2. $[\![Q]\!] = \mu_1 \triangleright \ldots \triangleright \mu_n \triangleright [\![Q_1]\!] \xrightarrow{X}_r Q''$ is derived by repeated applications of rules $\textsc{Read}_{t1}$ and $\textsc{Read}_{t2}$, hence (i) $[\![Q_1]\!] \xrightarrow{X}_r Q''_1$, (ii) for each $i \in [1, n]$, either $\mu_i = \alpha_i$ or $\mu_i = \underline{\alpha_i}$ and $\alpha_i \notin X \cup \{\tau\}$ and (iii) $Q'' = \underline{\alpha_1} \triangleright \ldots \triangleright \underline{\alpha_n} \triangleright Q''_1$. If $[\![Q_1]\!] \xrightarrow{X}_r Q''_1$ then, by induction hypothesis, $Q_1 \xrightarrow{X}_r Q'_1$ and $Q''_1 = [\![Q'_1]\!]$. Moreover, by (ii), we have also $\mathcal{U}(\{\mu_1, \ldots, \mu_n\}) \cap (X \cup \{\tau\}) = \emptyset$. Thus, $Q \xrightarrow{X}_r \underline{\{\mu_1, \ldots, \mu_n\}} \triangleright Q'_1 = \{\underline{\alpha_1}, \ldots, \underline{\alpha_n}\} \triangleright Q'_1 = Q'$ with $[\![Q']\!] = \underline{\alpha_1} \triangleright \ldots \triangleright \underline{\alpha_n} \triangleright [\![Q'_1]\!] = \underline{\alpha_1} \triangleright \ldots \triangleright \underline{\alpha_n} \triangleright Q''_1 = Q''$.

$\square$

D A Proof of Proposition 5.10

This section is devoted to proving Proposition 5.10.

Proposition 5.10 Let $Q, Q' \in \mathsf{L}(\tilde{\mathbb{P}})$ and $A \subseteq \mathbb{A}_\tau$. Then:

1. $Q \xrightarrow{\alpha} Q'$ and $s = \langle v_1, \ldots, v_n \rangle \in \mathsf{UE}(Q, A)$ implies either $s \in \mathsf{UE}(Q', A)$ or there exists some $j \in [1, n]$ such that $v_j \notin \mathsf{LAB}(Q')$.

2. $Q \xrightarrow{X}_r Q'$ implies $\mathsf{LE}(Q, A) = \mathsf{LE}(Q', A) = \mathsf{UE}(Q', A)$.

Proof:
We start proving Item 1. Since $Q \overset{\alpha}{\rightsquigarrow} Q'$ implies $Q' = Q$ and, hence, $\mathsf{UE}(Q', A) = \mathsf{UE}(Q, A)$, we only prove that $Q \overset{\alpha}{\mapsto} Q'$ and $s = \langle v_1, \ldots, v_n \rangle \in \mathsf{UE}(Q, A)$ implies either $s \in \mathsf{UE}(Q', A)$ or $v_j \notin \mathsf{LAB}(Q')$ for some $j \in [1, n]$. We proceed by induction on derivation $Q \overset{\alpha}{\mapsto} Q'$.

Nil, Var: $Q = \mathsf{nil}_u$, $Q = x_u$. This case is not possible since $Q \overset{\alpha}{\not\mapsto}$.

Pref: $Q = \alpha_u.P_1$ or $Q = \underline{\alpha}_u.P_1$ with $P_\in \mathsf{L}_{u1}(\tilde{\mathbb{P}}_1)$. Of course only the latter case is possible. Then $Q \overset{\alpha}{\mapsto} P_1$, $s \in \mathsf{UE}(Q, A)$ implies $\alpha \notin A$ and $s = \langle u \rangle$. By Fact 5.5-2, $u \notin \mathsf{LAB}(P_1)$.

Read: $Q = \mu_{u1} \rhd_u Q_1$ with $Q_1 \in \mathsf{L}_{u2}(\tilde{\mathbb{P}})$. By operational rules $Q \overset{\alpha}{\mapsto} Q'$ implies $Q_1 \overset{\alpha}{\mapsto} Q'$ and, by Fact 5.5-2, $Q' \in \mathsf{L}_v(\tilde{\mathbb{P}})$ with $u2 \leq v$. Moreover, $s \in \mathsf{UE}(Q, A)$ implies either $s = \langle u1 \rangle$ or $s = \langle v_1, \ldots, v_n \rangle \in \mathsf{UE}(Q_1, A)$. In the latter case the statement follows by induction hypothesis. In the former one, as in the Pref case, $s = \langle u1 \rangle$ and $Q' \in \mathsf{L}_v(\tilde{\mathbb{P}})$ with $u2 \leq v$ implies $u \notin \mathsf{LAB}(Q')$.

Sum: $Q = Q_1 +_u Q_2$. By the operational semantics $Q_1 +_u Q_2 \overset{\alpha}{\mapsto} Q'$ if either $Q_1 \overset{\alpha}{\mapsto} Q'$ or $Q_2 \overset{\alpha}{\mapsto} Q'$. Assume the former case. The other one is similar. If $s \in \mathsf{UE}(Q_1, A)$ then by induction hypothesis either $s \in \mathsf{UE}(Q', A)$ or $v_j \notin \mathsf{LAB}(Q')$ for some $j \in [1, n]$. Then assume $s \in \mathsf{UE}(Q_2, A)$. By the labelling function, each $v_j \in \mathsf{LAB}(Q_1)$ is of the form $v_j = u1u_j$ while each $v_j \in \mathsf{LAB}(Q_2)$ is of the form $v_j = u2u_j$, for some label u_j. Then by Fact 5.5-2, we have $v_j \notin \mathsf{LAB}(Q_1)$ and $v_j \notin \mathsf{LAB}(Q')$, for any $j \in [1, n]$.

Par: $Q = Q_1 \parallel_B^u Q_2$. Assume $Q \xrightarrow{\alpha} Q'$ and $s = \langle v_1, \ldots, v_n \rangle \in \mathsf{UE}(Q, A)$.
By $s \in \mathsf{UE}(Q_1 \parallel_B^u Q_2, A)$ we have either (i) $s \in \mathsf{UE}(Q_1, A \cup B)$, (ii) $s \in \mathsf{UE}(Q_2, A \cup B)$ or (iii) $s \in \mathsf{UE}(Q_1, \mathbb{A}_\tau \backslash \{\alpha\}) \times \mathsf{UE}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\})$, for some $\alpha \in B \backslash A$. Now consider the following three possible cases:

- $\alpha \notin B$, $Q_1 \overset{\alpha}{\mapsto} Q_1'$ and $Q' = \mathsf{clean}(Q_1' \parallel_B^u Q_2)$.
  Consider case (i). By induction hypothesis it is either $s \in \mathsf{UE}(Q_1', A \cup B)$ or $v_j \notin \mathsf{LAB}(Q_1')$ for some $j \in [1, n]$. If $s \in \mathsf{UE}(Q_1', A \cup B)$ then $s \in \mathsf{UE}(Q_1' \parallel_B^u Q_2, A) = \mathsf{UE}(\mathsf{clean}(Q_1' \parallel_B^u Q_2), A) = \mathsf{UE}(Q', A)$ by Lemma A.11. Now assume that $v_j \notin \mathsf{LAB}(Q_1')$, for some $j \in [1, n]$. Since $s \in \mathsf{UE}(Q_1, A \cup B)$ implies also $v_j \neq u$ and $v_j \notin \mathsf{LAB}(Q_2)$ (Fact 5.4), we have that $v_j \notin \mathsf{LAB}(Q_1' \parallel_B^u Q_2) = \mathsf{LAB}(\mathsf{clean}(Q_1' \parallel_B^u Q_2)) = \mathsf{LAB}(Q')$.
  Consider case (ii). Then, again by Lemma A.11, we have that $s \in \mathsf{UE}(Q_2, A \cup B) \subseteq \mathsf{UE}(Q_1' \parallel_B^u Q_2, A) = \mathsf{UE}(\mathsf{clean}(Q_1' \parallel_B^u Q_2), A) = \mathsf{UE}(Q', A)$.
  Finally consider case (iii). By definition of $\times$, if $s \in \mathsf{UE}(Q_1, \mathbb{A}_\tau \backslash \{\alpha\}) \times \mathsf{UE}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\})$ then there exist s_1 and s_2 process labels such that $s_1 \in \mathsf{UE}(Q_1, \mathbb{A}_\tau \backslash \{\alpha\})$ and $s_2 \in \mathsf{UE}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\})$ and $s = s_1 \times s_2$. By induction hypothesis, either $s_1 \in \mathsf{UE}(Q_1', \mathbb{A}_\tau \backslash \{\alpha\})$ or $v_j \notin \mathsf{LAB}(Q_1')$ for some $j \in [1, n]$. If $s_1 \in \mathsf{UE}(Q_1', \mathbb{A}_\tau \backslash \{\alpha\})$ then it is also $s = s_1 \times s_2 \in \mathsf{UE}(Q_1', \mathbb{A}_\tau \backslash \{\alpha\}) \times \mathsf{UE}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\}) \subseteq \mathsf{UE}(Q_1' \parallel_B^u Q_2, A) = \mathsf{UE}(\mathsf{clean}(Q_1' \parallel_B^u Q_2), A) =$

$\mathsf{UE}(Q', A)$, again by Lemma A.11. If $v_j \notin \mathsf{LAB}(Q'_1)$ as in the previous cases we have that $v_j \notin \mathsf{LAB}(Q'_1 \|^u_B Q_2) = \mathsf{LAB}(Q')$.

- $\alpha \notin B$, $Q_2 \overset{\alpha}{\mapsto} Q'_2$ and $Q' = \mathsf{clean}(Q_1 \|^u_B Q'_2)$. This case is similar to the previous one.
- $\alpha \in B$, $Q_i \overset{\alpha}{\mapsto} Q'_i$ for $i = 1, 2$ and $Q' = \mathsf{clean}(Q'_1 \|^u_B Q'_2)$. Consider case (i). By induction hypothesis either $s \in \mathsf{UE}(Q'_1, A \cup B)$ or $v_j \notin \mathsf{LAB}(Q'_1)$ for some $j \in [1, n]$. As in the previous items, we can prove that $s \in \mathsf{UE}(Q'_1, A \cup B)$ implies $s \in \mathsf{UE}(Q'_1 \|_B Q'_2, A) = \mathsf{UE}(Q', A)$ and $v_j \notin \mathsf{LAB}(Q'_1)$ implies $v_j \notin \mathsf{LAB}(Q'_1 \|^u_B Q'_2) = \mathsf{LAB}(Q')$. Case (ii) can be proven similarly. Then consider case (iii). By definition of $\times$, $s \in \mathsf{UE}(Q_1, \mathbb{A}_\tau \backslash \{\alpha\}) \times \mathsf{UE}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\})$ implies there are s_1 and s_2 process labels such that $s_i = \langle v_{i1}, \ldots, v_{in_i} \rangle \in \mathsf{UE}(Q_i, \mathbb{A}_\tau \backslash \{\alpha\})$. By induction hypothesis, if both $s_1 \in \mathsf{UE}(Q'_1, \mathbb{A}_\tau \backslash \{\alpha\})$ and $s_2 \in \mathsf{UE}(Q'_2, \mathbb{A}_\tau \backslash \{\alpha\})$ then $s_1 \times s_2 \in \mathsf{UE}(Q'_1, \mathbb{A}_\tau \backslash \{\alpha\}) \times \mathsf{UE}(Q'_2, \mathbb{A}_\tau \backslash \{\alpha\}) \subseteq \mathsf{UE}(Q'_1 \|^u_B Q'_2, A) = \mathsf{UE}(Q', A)$. If $v_{1j} \notin \mathsf{LAB}(Q'_1)$ (similarly for $v_{2j} \notin \mathsf{LAB}(Q'_2)$) then, as in the Sum case, we have that $v_{1j} \notin \mathsf{LAB}(Q'_2)$ and hence also $v_{1j} \notin \mathsf{LAB}(Q')$.

Rel: $Q = Q_1[\Phi_u]$. By the operational semantics, $Q \overset{\alpha}{\mapsto} Q'_1[\Phi_u]$ if there exists $\beta \in \Phi^{-1}(\alpha)$ such that $Q_1 \overset{\beta}{\mapsto} Q'_1$. Moreover, $s = \langle v_1, \ldots v_n \rangle \in \mathsf{UE}(Q_1[\Phi_u], A)$ if $s \in \mathsf{UE}(Q_1, \Phi^{-1}(A))$. By induction hypothesis either $s \in \mathsf{UE}(Q'_1, \Phi^{-1}(A))$ or $v_j \notin \mathsf{LAB}(Q'_1)$ for some $j \in [1, n]$. In the former case $s \in \mathsf{UE}(Q'_1[\Phi_u], A)$. In the latter one, $v_j \in \mathsf{LAB}(Q_1)$ implies $v \neq u$ and hence also $v_j \notin \mathsf{LAB}(Q'_1[\Phi_u])$.

Rec: $Q = \mathsf{rec}x_u.Q_1$. Let $S_1 = \mathsf{unmark}(Q_1)$ and $S = Q_1\{\!|\mathsf{rec}x_u.S_1/x|\!\}$. By operational rules $Q \overset{\alpha}{\mapsto} Q'$ implies $S \overset{\alpha}{\mapsto} Q'$. Now assume $s \in \mathsf{UE}(Q, A) = \mathsf{UE}(Q_1, A) = \mathsf{UE}(S, A)$ by Proposition A.4-2. By induction hypothesis we have either $s \in \mathsf{UE}(Q', A)$ or $v_j \notin \mathsf{LAB}(Q')$ for some $j \in [1, n]$.

Then prove Item 2. The two statements are proven by induction on Q.

Var, Stop: $Q = x_u$. This case is not possible since $Q \overset{X}{\not\longrightarrow}$.

Nil: $Q = \mathsf{nil}_u$. In this case $Q \xrightarrow{X} \mathsf{nil} = Q'$ and $\mathsf{LE}(Q, A) = \mathsf{LE}(Q', A) = \mathsf{UE}(Q', A) = \emptyset$ for any $A \subseteq \mathbb{A}_\tau$.

Pref: $Q = \alpha_u.P_1$ or $Q = \underline{\alpha}_u.P_1$. We prove only the latter case (the former one can be proven similarly). Assume that $Q \xrightarrow{X} \underline{\alpha}_u.P_1 = Q'$. If $\alpha \notin A$ then $\mathsf{LE}(Q, A) = \mathsf{LE}(Q', A) = \mathsf{UE}(Q', A) = \{\langle u \rangle\}$, otherwise $\mathsf{LE}(Q, A) = \mathsf{LE}(Q', A) = \mathsf{UE}(Q', A) = \emptyset$.

Read: $Q = \alpha_u \triangleright Q_1$ or $Q = \underline{\alpha}_u \triangleright Q_1$. We prove only the latter case (the former one can be proven similarly). By operational rules, $Q \xrightarrow{X} Q'$ implies $Q_1 \xrightarrow{X} Q'_1$ and $Q' = \underline{\alpha}_u \triangleright Q'_1$. If $\alpha \notin A$ then, by induction hypothesis, $\mathsf{LE}(Q, A) = \{\langle u \rangle\} \cup \mathsf{LE}(Q_1, A) = \{\langle u \rangle\} \cup \mathsf{LE}(Q'_1, A) = \mathsf{LE}(Q', A)$. Similarly we can prove that $\mathsf{LE}(Q, A) = \mathsf{UE}(Q', A)$. If $\alpha \notin A$, then $\mathsf{LE}(Q, A) = \mathsf{LE}(Q_1, A) = \mathsf{LE}(Q', A)$ and, similarly, $\mathsf{LE}(Q, A) = \mathsf{LE}(Q_1, A) = \mathsf{UE}(Q'_1, A) = \mathsf{UE}(Q', A)$.

Sum: $Q = Q_1 +_u Q_2$. By the operational rule $Q \xrightarrow{X} Q'_1 +_u Q'_2$ implies $Q_1 \xrightarrow{X} Q'_1$ and $Q_2 \xrightarrow{X} Q'_2$. By induction hypothesis $\mathsf{LE}(Q, A) = \mathsf{LE}(Q_1, A) \cup \mathsf{LE}(Q_2, A) = \mathsf{LE}(Q'_1, A) \cup \mathsf{LE}(Q'_2, A) = \mathsf{LE}(Q', A)$. Similarly $\mathsf{LE}(Q', A) = \mathsf{LE}(Q'_1, A) \cup \mathsf{LE}(Q'_2, A) = \mathsf{UE}(Q'_1, A) \cup \mathsf{UE}(Q'_2, A) = \mathsf{UE}(Q', A)$.

Par: $Q = Q_1 \|^u_A Q_2$. In this case, we have that $Q \xrightarrow{X} Q'$ if $X \subseteq (A \cap (X_1 \cup X_2)) \cup ((X_1 \cap X_2) \backslash A)$, $Q_1 \xrightarrow{X_1} Q'_1$ and $Q_2 \xrightarrow{X_2} Q'_2$ and $Q' = Q'_1 \|^u_A Q'_2$. By induction hypothesis $\mathsf{LE}(Q_i, A) = \mathsf{LE}(Q'_i, A) = \mathsf{UE}(Q'_i, A)$, for $i = 1, 2$. Then: $\mathsf{LE}(Q, A) =$

$\mathsf{LE}(Q_1, A \cup B) \cup \mathsf{LE}(Q_2, A \cup B) \cup (\cup_{\alpha \in B \backslash A} \mathsf{LE}(Q_1, \mathbb{A}_\tau \backslash \{\alpha\}) \times \mathsf{LE}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\})) =$

$\mathsf{LE}(Q'_1, A \cup B) \cup \mathsf{LE}(Q'_2, A \cup B) \cup (\cup_{\alpha \in B \setminus A} \mathsf{LE}(Q'_1, \mathbb{A}_\tau \setminus \{\alpha\}) \times \mathsf{LE}(Q'_2, \mathbb{A}_\tau \setminus \{\alpha\})) =$ $\mathsf{LE}(Q'_1 \|_A Q'_2, A) = \mathsf{LE}(\mathsf{clean}(Q'_1 \|_A Q'_2), A) = \mathsf{LE}(Q', A)$, by Proposition A.10-1. Similarly we can prove that $\mathsf{LE}(Q', A) = \mathsf{UE}(Q'_1 \|_A Q'_2, A) = \mathsf{UE}(\mathsf{clean}(Q'_1 \|_A Q'_2), A) = \mathsf{UE}(Q', A)$, by Lemma A.11.

Rel: $Q = Q_1[\Phi_u]$. $Q \xrightarrow{X} Q'_1[\Phi_u]$ implies $Q_1 \xrightarrow{\Phi^{-1}(X \cup \{\tau\}) \setminus \{\tau\}} Q'_1$. By induction hypothesis we have that $\mathsf{LE}(Q_1, A) = \mathsf{LE}(Q'_1, A) = \mathsf{UE}(Q'_1, A)$. Thus $\mathsf{LE}(Q, A) = \mathsf{LE}(Q_1, \Phi^{-1}(A)) = \mathsf{LE}(Q'_1, \Phi^{-1}(A)) = \mathsf{LE}(Q', A)$. Similarly, $\mathsf{LE}(Q'_1, \Phi^{-1}(A)) = \mathsf{UE}(Q'_1, \Phi^{-1}(A)) = \mathsf{UE}(Q', A)$.

Rec: $Q = \mathsf{rec}\, x_u.Q_1$. $Q \xrightarrow{X}_r Q'$ implies $Q_1 \xrightarrow{X}_r Q'_1$ and $Q' = \mathsf{rec}\, x_u.Q'_1$. By induction hypothesis $\mathsf{LE}(Q', A) = \mathsf{LE}(Q'_1, A) = \mathsf{LE}(Q_1, A) = \mathsf{LE}(Q, A)$ and $\mathsf{UE}(Q', A) = \mathsf{UE}(Q'_1, A) = \mathsf{LE}(Q_1, A) = \mathsf{LE}(Q, A)$.

□

E A Proof of Proposition 5.11

This section is devoted to proving Proposition 5.11. A preliminary lemma is needed.

Lemma E.1 Let $Q \in \mathsf{L}(\tilde{\mathbb{P}})$, X and $Y \subseteq \mathbb{A}$.

1. $Q \xrightarrow{X}_r Q'$ and $Y \cap \mathcal{U}(Q, A) = \emptyset$ imply $Q \xrightarrow{X \cup (Y \backslash A)}_r Q'$;
2. $Q \xrightarrow{X}_r$ implies $\mathcal{U}(Q, \mathbb{A} \backslash X) = \emptyset$;
3. Q action-guarded and $\mathcal{U}(Q, \mathbb{A} \backslash X) = \emptyset$ implies $Q \xrightarrow{X}_r$.

Proof: We prove these items together by induction on Q.

Var: $Q = x_u$. This case is not possible since $Q \not\xrightarrow{X}_r$ and Q is not action-guarded.

Nil: $Q = \mathsf{nil}_u$. In this case:

1. $Q \xrightarrow{X}_r \mathsf{nil}$, $Y \cap \mathcal{U}(Q, A) = \emptyset$ and $Q \xrightarrow{X \cup (Y \backslash A)}_r \mathsf{nil}$.
2. $Q \xrightarrow{X}_r$ and $\mathcal{U}(Q, \mathbb{A} \backslash X) = \emptyset$.
3. Q is action-guarded, $\mathcal{U}(Q, \mathbb{A} \backslash X) = \emptyset$ and $Q \xrightarrow{X}_r$.

Pref: $Q = \alpha_u.P_1$ or $Q = \underline{\alpha}_u.P_1$. Consider only the second case (the first case is similar to the Nil-case).

1. $Q \xrightarrow{X}_r Q'$ implies, by operational rules, $\alpha \notin X \cup \{\tau\}$ and $Q' = \underline{\alpha}_u.P_1$. If $\alpha \in A$ then, easily, $\alpha \notin Y \backslash A$. Otherwise, $\alpha \notin A$ and $Y \cap \mathcal{U}(Q, A) = Y \cap \{\alpha\} = \emptyset$ imply $\alpha \notin Y$. So, again $\alpha \notin Y \backslash A$. Thus, $\alpha \notin (X \cup (Y \backslash A)) \cup \{\tau\}$ and, again by operational rules, $Q \xrightarrow{X \cup (Y \backslash A)}_r \underline{\alpha}_u.P_1 = Q'$
2. $Q \xrightarrow{X}_r$ implies $\alpha \notin X \cup \{\tau\}$ and, hence, $\alpha \in \mathbb{A} \backslash X$. Then $\mathcal{U}(Q, \mathbb{A} \backslash X) = \emptyset$.
3. Q is action-guarded. Moreover, $\mathcal{U}(Q, \mathbb{A} \backslash X) = \emptyset$ implies $\alpha \in \mathbb{A} \backslash X$. Since $\tau \neq \alpha \in \mathbb{A}$ and $\alpha \notin X$, we have that $\alpha \notin X \cup \{\tau\}$ and, hence, $Q \xrightarrow{X}_r$.

Read: $Q = \alpha_{u1} \triangleright_u Q_1$ or $Q = \underline{\alpha}_{u1} \triangleright_u Q_1$. Consider only the second case (the first case is easier).

1. $Q \xrightarrow{X}_r Q'$ implies, by operational rules, $\alpha \notin X \cup \{\tau\}$, $Q_1 \xrightarrow{X}_r Q_1'$ and $Q' = \underline{\alpha}_{u1} \triangleright_u Q_1'$. By induction hypothesis, $Q_1 \xrightarrow{X}_r Q_1'$ and $Y \cap \mathcal{U}(Q_1, A) \subseteq Y \cap \mathcal{U}(Q, A) = \emptyset$ implies $Q_1 \xrightarrow{X \cup (Y \backslash A)}_r Q_1'$. Furthermore, if $\alpha \in A$ then, easily, $\alpha \notin Y \backslash A$. If $\alpha \notin A$ then $\emptyset = Y \cap \mathcal{U}(Q, A) \supseteq Y \cap \{\alpha\}$ imply $\alpha \notin Y$. So, it is again $\alpha \notin Y \backslash A$. Thus, $\alpha \notin (X \cup (Y \backslash A)) \cup \{\tau\}$. Finally, again by operational rules, $Q \xrightarrow{X \cup (Y \backslash A)}_r \underline{\alpha}_{u1} \triangleright_u Q_1' = Q'$.
2. $Q \xrightarrow{X}_r$ implies $\alpha \notin X \cup \{\tau\}$ and $Q_1 \xrightarrow{X}_r$. By induction hypothesis it is $\mathcal{U}(Q_1, \mathbb{A} \backslash X) = \emptyset$. Moreover, $\alpha \notin X \cup \{\tau\}$ implies $\alpha \in \mathbb{A} \backslash X$. Then $\mathcal{U}(Q, \mathbb{A} \backslash X) = \emptyset$.
3. Assume Q and, hence, Q_1 action-guarded. Assume, moreover, $\mathcal{U}(Q, \mathbb{A} \backslash X) = \emptyset$, i.e. $\alpha \in \mathbb{A} \backslash X$ and $\mathcal{U}(Q_1, \mathbb{A} \backslash X) = \emptyset$. Q_1 action-guarded and $\mathcal{U}(Q_1, \mathbb{A} \backslash X) = \emptyset$ implies, by induction hypothesis that $Q_1 \xrightarrow{X}_r$. Moreover, $\alpha \in \mathbb{A} \backslash X$ implies $\tau \neq \alpha \in \mathbb{A}$ and $\alpha \notin X$, that is $\alpha \notin X \cup \{\tau\}$. Finally, by operational rule, $Q \xrightarrow{X}_r$.

Sum: $Q = Q_1 +_u Q_2$

1. $Q \xrightarrow{X}_r Q'$ and $Y \cap \mathcal{U}(Q, A) = Y \cap (\mathcal{U}(Q_1, A) \cup \mathcal{U}(Q_2, A)) = \emptyset$ imply $Q_i \xrightarrow{X}_r Q_i'$, $Y \cap \mathcal{U}(Q_i, A) = \emptyset$ for $i = 1, 2$ and $Q' = Q_1' +_u Q_2'$. By induction hypothesis $Q_1 \xrightarrow{X\cup(Y\backslash A)}_r Q_1'$, $Q_2 \xrightarrow{X\cup(Y\backslash A)}_r Q_2'$. Thus, by operational rules, $Q \xrightarrow{X\cup(Y\backslash A)}_r Q_1' + Q_2' = Q'$.
2. If $Q \xrightarrow{X}_r$ then $Q_1 \xrightarrow{X}_r$ and $Q_2 \xrightarrow{X}_r$. By induction hypothesis $\mathcal{U}(Q_1, \mathbb{A}\backslash X) = \emptyset$, $\mathcal{U}(Q_2, \mathbb{A}\backslash X) = \emptyset$ and, hence, $\mathcal{U}(Q, \mathbb{A}\backslash X) = \emptyset$.
3. In this case Q guarded implies both Q_1 and Q_2 guarded. Moreover if $\mathcal{U}(Q, \mathbb{A}\backslash X) = \emptyset$ then $\mathcal{U}(Q_1, \mathbb{A}\backslash X) = \mathcal{U}(Q_2, \mathbb{A}\backslash X) = \emptyset$. By induction hypothesis $Q_1 \xrightarrow{X}_r$, $Q_2 \xrightarrow{X}_r$ and, hence, $Q \xrightarrow{X}_r$.

Par: $Q = Q_1 \parallel_B^u Q_2$.

1. If $Q \xrightarrow{X}_r Q'$ then, by operational rules, there exist X_1, X_2 such that $Q_1 \xrightarrow{X_1}_r Q_1'$, $Q_2 \xrightarrow{X_2}_r Q_2'$, $X \subseteq (B \cap (X_1 \cup X_2)) \cup ((X_1 \cap X_2)\backslash B)$ and $Q' = \mathsf{clean}(Q_1' \parallel_B^u Q_2')$. Now, let $A_1 = (\mathcal{U}(Q_1)\backslash\mathcal{U}(Q_2)) \cap B$ and $A_2 = (\mathcal{U}(Q_2)\backslash\mathcal{U}(Q_1)) \cap B$. Now we want to prove that $Y \cap \mathcal{U}(Q, A) = \emptyset$ implies $Y \cap \mathcal{U}(Q_1, A \cup A_1) = Y \cap \mathcal{U}(Q_2, A \cup A_2) = \emptyset$.
   Assume $Y \cap \mathcal{U}(Q, A) = \emptyset$ and, by contradiction, $\alpha \in Y \cap \mathcal{U}(Q_1, A \cup A_1) \neq \emptyset$. Then, by Propositions A.1-1 and A.1-4, $\alpha \in Y$ such that $\alpha \in \mathcal{U}(Q_1)$, $\alpha \notin A$ and $\alpha \notin A_1$.
   We have to consider two possible subcases. If $\alpha \notin B$ then $\alpha \in \mathcal{U}(Q_1)$ and $\alpha \notin A \cup B$ implies $\alpha \in \mathcal{U}(Q_1, A \cup B) \subseteq \mathcal{U}(Q, A)$ (see Proposition A.1-2). Otherwise, if $\alpha \in B$ then $\alpha \in \mathcal{U}(Q_1)$ and $\alpha \notin A_1$ implies also $\alpha \in \mathcal{U}(Q_2)$. Thus $\alpha \in \mathcal{U}(Q_1) \cap \mathcal{U}(Q_2) \cap B$ such that $\alpha \notin A$. Again by Proposition A.1-2, it is $\alpha \in \mathcal{U}(Q_1, A) \cap \mathcal{U}(Q_2, A) \cap B \subseteq \mathcal{U}(Q, A)$. In both cases, we have $\alpha \in Y \cap \mathcal{U}(Q, A) = \emptyset$
   This prove that $Y \cap \mathcal{U}(Q, A) = \emptyset$ implies $Y \cap \mathcal{U}(Q_1, A \cup A_1) = \emptyset$. Similarly we can prove that if $Y \cap \mathcal{U}(Q, A) = \emptyset$ then also $Y \cap \mathcal{U}(Q_2, A \cup A_2) = \emptyset$. By induction hypothesis $Q_1 \xrightarrow{X_1\cup(Y\backslash(A\cup A_1))}_r Q_1'$ and $Q_2 \xrightarrow{X_2\cup(Y\backslash(A\cup A_2))}_r Q_2'$. Let $X_1' = X_1 \cup (Y\backslash(A \cup A_1))$ and $X_2' = X_2 \cup (Y\backslash(A \cup A_2))$. By operational rules it remains to prove that $X \cup (Y\backslash A) \subseteq (B \cap (X_1' \cup X_2')) \cup ((X_1' \cap X_2')\backslash B)$. First of all, $X_1 \subseteq X_1'$ and $X_2 \subseteq X_2'$ imply $X \subseteq (B \cap (X_1 \cup X_2)) \cup ((X_1 \cap X_2)\backslash B) \subseteq (B \cap (X_1' \cup X_2')) \cup ((X_1' \cap X_2')\backslash B)$. Now assume $\alpha \in Y\backslash A$ and consider the following possible subcases:
   - $\alpha \notin B$. Then $\alpha \notin A_1$ and $\alpha \notin A_2$ implies $\alpha \in Y\backslash(A \cup A_1) \subseteq X_1'$ and $\alpha \in Y\backslash(A \cup A_2) \subseteq X_2'$. Thus, $\alpha \in (X_1' \cap X_2')\backslash B$.
   - $\alpha \in B$. In this case, $\alpha \notin A_1$ implies $\alpha \in Y\backslash(A \cup A_1) \subseteq X_1'$. If, otherwise, $\alpha \in A_1 = (\mathcal{U}(Q_1)\backslash\mathcal{U}(Q_2)) \cap B$, then $\alpha \notin \mathcal{U}(Q_2)$ implies $\alpha \notin A_2$ and, hence, $\mu \in Y\backslash(A \cup A_2) \subseteq X_2'$. In both cases $\mu \in B \cap (X_1' \cup X_2')$.

   We can conclude that $Q \xrightarrow{X\cup(Y\backslash A)}_r Q'$.
2. If $Q \xrightarrow{X}_r$ then $Q_1 \xrightarrow{X_1}_r$, $Q_2 \xrightarrow{X_2}_r$ with $X \subseteq (B \cap (X_1 \cup X_2)) \cup ((X_1 \cap X_2)\backslash B)$. By induction hypothesis $\mathcal{U}(Q_1, \mathbb{A}\backslash X_1) = \mathcal{U}(Q_2, \mathbb{A}\backslash X_2) = \emptyset$.
   Now, assume toward a contradiction that $\alpha \in \mathcal{U}(Q, \mathbb{A}\backslash X) \neq \emptyset$. By Proposition A.1 (Items 4. and 1.) $\alpha \in \mathcal{U}(Q)$ such that $\alpha \notin \mathbb{A}\backslash X$. We distinguish the following possible subcases:
   - $\alpha = \tau$. Then $\alpha \in \mathcal{U}(Q)$ implies either $\alpha \in \mathcal{U}(Q_1, B)$ or $\alpha \in \mathcal{U}(Q_2, B)$. Moreover, surely $\tau \notin \mathbb{A}\backslash X_1$ and $\tau \notin \mathbb{A}\backslash X_2$. By Proposition A.1-2., we have either $\alpha \in \mathcal{U}(Q_1, \mathbb{A}\backslash X_1) = \emptyset$ or $\alpha \in \mathcal{U}(Q_2, \mathbb{A}\backslash X_2) = \emptyset$.
   - $\alpha \in \mathbb{A}$ (and, since $\alpha \notin \mathbb{A}\backslash X$, also $\alpha \in X$) such that $\alpha \notin B$. As in the above case, we have that $\alpha \in \mathcal{U}(Q)$ implies either $\alpha \in \mathcal{U}(Q_1, B)$ or $\alpha \in \mathcal{U}(Q_2, B)$. Moreover, $\alpha \in X\backslash B$ implies $\alpha \in X_1 \cap X_2$ and hence both $\alpha \notin \mathbb{A}\backslash X_1$ and $\alpha \notin \mathbb{A}\backslash X_2$. As in the above case we can prove that either $\alpha \in \mathcal{U}(Q_1, \mathbb{A}\backslash X_1) = \emptyset$ or $\alpha \in \mathcal{U}(Q_2, \mathbb{A}\backslash X_2) = \emptyset$.

- $\alpha \in \mathbb{A}$ (and, since $\alpha \notin \mathbb{A}\backslash X$, also $\alpha \in X$) such that $\alpha \in B$. In such a case $\alpha \in \mathcal{U}(Q)$ implies $\alpha \in \mathcal{U}(Q_1, A) \cap \mathcal{U}(Q_2, A) \cap B$. Moreover, $\alpha \in X \cap B$ implies $\alpha \in X_1 \cup X_2$ and hence either $\alpha \notin \mathbb{A}\backslash X_1$ or $\alpha \notin \mathbb{A}\backslash X_2$. Also in this case we can conclude that either $\alpha \in \mathcal{U}(Q_1, \mathbb{A}\backslash X_1) = \emptyset$ or $\alpha \in \mathcal{U}(Q_2, \mathbb{A}\backslash X_2) = \emptyset$.

3. Let us assume Q action-guarded (and, hence, both Q_1 and Q_2 action-guarded) and $\mathcal{U}(Q, \mathbb{A}\backslash X) = \emptyset$. By Definition 2.2 it is $\mathcal{U}(Q_1, (\mathbb{A}\backslash X) \cup B) = \mathcal{U}(Q_2, (\mathbb{A}\backslash X) \cup B) = \emptyset$ and $\mathcal{U}(Q_1, \mathbb{A}\backslash X) \cap \mathcal{U}(Q_2, \mathbb{A}\backslash X) \cap B = \emptyset$. First we prove that $(\mathbb{A}\backslash X) \cup B = \mathbb{A}\backslash(X\backslash B)$.

"$\subseteq$" If $\alpha \in \mathbb{A}\backslash X$ then $\mathbb{A}\backslash X \subseteq \mathbb{A}\backslash(X\backslash B)$ implies also $\alpha \in \mathbb{A}\backslash(X\backslash B)$. If, otherwise, $\alpha \in B \subseteq \mathbb{A}$ then we have $\alpha \notin X\backslash B$ and hence $\alpha \in \mathbb{A}\backslash(X\backslash B)$.

"$\supseteq$" If $\alpha \in \mathbb{A}\backslash(X\backslash B)$, then $\alpha \in \mathbb{A}$ such that either $\alpha \notin X$ or $\alpha \in X \cap B$. Thus, it is either $\alpha \in \mathbb{A}\backslash X$ or $\alpha \in B$. In both cases $\alpha \in (\mathbb{A}\backslash X) \cup B$.

Q_i action-guarded and $\mathcal{U}(Q_i, (\mathbb{A}\backslash X) \cup B) = \mathcal{U}(Q_i, \mathbb{A}\backslash(X\backslash B)) = \emptyset$ implies by induction hypothesis, $Q_i \xrightarrow{X\backslash B}_r$ for $i = 1, 2$. Now, let us denote with $X'_1 = (X \cap B)\backslash\mathcal{U}(Q_1)$ and with $X'_2 = (X \cap B)\backslash\mathcal{U}(Q_2)$. Then: $Q_i \xrightarrow{X\backslash B}_r$ and $X'_i \cap \mathcal{U}(Q_i) = \emptyset$ implies by Item 1 $Q_i \xrightarrow{(X\backslash B) \cup X'_i}_r$. Moreover it is also $X'_1 \cup X'_2 = X \cap B$. Indeed, let us assume (toward a contradiction) that there exists $\alpha \in X \cap B$ such that both $\alpha \notin X'_1$ and $\alpha \notin X'_2$ and, hence, that both $\alpha \in \mathcal{U}(Q_1)$ and $\alpha \in \mathcal{U}(Q_2)$. Then, $\alpha \in \mathcal{U}(Q_1) \cap \mathcal{U}(Q_2) \cap B$ such that $\alpha \notin \mathbb{A}\backslash X$. Thus, by A.1-2, $\alpha \in \mathcal{U}(Q_1, \mathbb{A}\backslash X) \cap \mathcal{U}(Q_2, \mathbb{A}\backslash X) \cap B \subseteq \mathcal{U}(Q, \mathbb{A}\backslash X) = \emptyset$.

Let $X_1 = (X\backslash B) \cup X'_1$ and $X_2 = (X\backslash B) \cup X'_2$. By operational rules, to prove that $Q \xrightarrow{X}_r$ it still remains to show that $X \subseteq (B \cap (X_1 \cup X_2)) \cup ((X_1 \cap X_2)\backslash B)$. This follows easily because $(X_1 \cap X_2)\backslash B) = X\backslash B$ and $B \cap (X_1 \cup X_2) = X'_1 \cup X'_2 = X \cap B$.

Rel: $Q = Q_1[\Phi_u]$. As a first statement (that wil be useful in the remain of the proof) we prove that if $X \subseteq \mathbb{A}$ and $X' = \Phi^{-1}(X \cup \{\tau\})\backslash\{\tau\}$ then $\Phi^{-1}(\mathbb{A}\backslash X) = \mathbb{A}\backslash X'$. Let $\mu \in \Phi^{-1}(\mathbb{A}\backslash X)$. Then $\mu \in \mathbb{A}_\tau$ such that $\Phi(\mu) \in \mathbb{A}$ and $\Phi(\mu) \notin X$. In particular, $\Phi(\mu) \in \mathbb{A}$ implies $\mu \neq \tau$ (since $\Phi(\tau) = \tau \notin \mathbb{A}$) and $\Phi(\mu) \notin \{\tau\}$. Thus, $\mu \in \mathbb{A}$ such that $\Phi(\mu) \notin X \cup \{\tau\}$ and, hence, $\mu \notin \Phi^{-1}(X \cup \{\tau\}) \supseteq X'$. We can conclude that $\mu \in \mathbb{A}\backslash X'$.

Now, let $\mu \in \mathbb{A}\backslash X'$. Then $\mu \neq \tau$ and $\mu \notin X'$ imply $\mu \notin \Phi^{-1}(X \cup \{\tau\})$ and, hence, $\Phi(\mu) \notin X \cup \{\tau\}$. Finally, $\Phi(\mu) \notin X$ and $\Phi(\mu) \neq \tau$ imply $\Phi(\mu) \in \mathbb{A}\backslash X$ and, hence, $\mu \in \Phi^{-1}(\mathbb{A}\backslash X)$.

1. Assume $Q \xrightarrow{X}_r Q'_1[\Phi_u] = Q'$ and $Y \cap \mathcal{U}(Q, A) = Y \cap \Phi(\mathcal{U}(Q_1, \Phi^{-1}(A))) = \emptyset$. Then (by operational rules) $Q_1 \xrightarrow{X'}_r Q'_1$ and $\Phi^{-1}(Y) \cap \mathcal{U}(Q_1, \Phi^{-1}(A)) = \emptyset$. By induction hypothesis $Q_1 \xrightarrow{X' \cup (\Phi^{-1}(Y)\backslash\Phi^{-1}(A))}_r Q'_1$. Moreover, since $\Phi(\tau) = \tau$ and $Y \subseteq \mathbb{A}$ imply $\tau \notin \Phi^{-1}(Y)$, $\tau \notin \Phi^{-1}(Y)\backslash\Phi^{-1}(A) = \Phi^{-1}(Y\backslash A)$ and, hence, $\Phi^{-1}(Y\backslash A) = \Phi^{-1}(Y\backslash A)\backslash\{\tau\}$, we have $X' \cup (\Phi^{-1}(Y)\backslash\Phi^{-1}(A)) = (\Phi^{-1}(X \cup \{\tau\})\backslash\{\tau\}) \cup (\Phi^{-1}(Y\backslash A)\backslash\{\tau\}) =$
$(\Phi^{-1}(X \cup \{\tau\}) \cup \Phi^{-1}(Y\backslash A))\backslash\{\tau\} = \Phi^{-1}(X \cup \{\tau\} \cup (Y\backslash A))\backslash\{\tau\} =$
$\Phi^{-1}((X \cup (Y\backslash A)) \cup \{\tau\})\backslash\{\tau\}$.
Finally, again by operational rules, $Q \xrightarrow{X \cup (Y\backslash A)}_r Q'_1[\Phi_u] = Q'$

2. Assume $Q \xrightarrow{X}_r$ and, hence, $Q_1 \xrightarrow{X'}_r$. By induction hypothesis it is $\mathcal{U}(Q_1, \mathbb{A}\backslash X') = \mathcal{U}(Q_1, \Phi^{-1}(\mathbb{A}\backslash X)) = \emptyset$. Thus $\mathcal{U}(Q, \mathbb{A}\backslash X) = \Phi(\mathcal{U}(Q_1, \Phi^{-1}(\mathbb{A}\backslash X))) = \emptyset$.

3. Assume Q action-guarded and $\mathcal{U}(Q, \mathbb{A}\backslash X) = \Phi(\mathcal{U}(Q_1, \Phi^{-1}(\mathbb{A}\backslash X))) = \emptyset$. Then we have also Q_1 action-guarded and $\mathcal{U}(Q_1, \Phi^{-1}(\mathbb{A}\backslash X)) = \mathcal{U}(Q_1, \mathbb{A}\backslash X') = \emptyset$. By induction hypothesis $Q_1 \xrightarrow{X'}_r$ ad, hence, $Q \xrightarrow{X}_r$.

Rec: $Q = \mathsf{rec}\, x_u.Q_1$.

1. If $Q \xrightarrow{X}_r Q'$ then $Q_1 \xrightarrow{X}_r Q'_1$ and $Q' = \mathsf{rec}\, x_u.Q'_1$. Moreover, assume $Y \cap \mathcal{U}(Q, A) = Y \cap \mathcal{U}(Q_1) = \emptyset$. By induction hypothesis we have that $Q_1 \xrightarrow{X \cup (Y \backslash A)}_r Q'_1$ and, by the operational semantics, $Q \xrightarrow{X \cup (Y \backslash A)}_r \mathsf{rec}\, x_u.Q'_1 = Q'$.
2. Assume $Q \xrightarrow{X}_r$ and hence $Q_1 \xrightarrow{X}_r$. By induction hypothesis we have that $\mathcal{U}(Q, \mathbb{A} \backslash X) = \mathcal{U}(Q_1, \mathbb{A} \backslash X) = \emptyset$.
3. In such a case Q action-guarded implies Q_1 action-guarded. Moreover, $\mathcal{U}(Q, \mathbb{A} \backslash X) = \mathcal{U}(Q_1, \mathbb{A} \backslash X) = \emptyset$ implies, by induction hypothesis, $Q_1 \xrightarrow{X}_r$ and, hence, $Q \xrightarrow{X}_r$.

$\square$

Proposition 5.11 Let $Q \in \mathsf{L}(\tilde{\mathbb{P}})$ and $A \subseteq \mathbb{A}_\tau$. Then:

1. $Q \xrightarrow{1}$ implies $\mathcal{U}(Q) = \emptyset$;
2. Q guarded and $\mathcal{U}(Q) = \emptyset$ implies $Q \xrightarrow{1}$;
3. $\mathcal{U}(Q, A) = \emptyset$ if and only if $\mathsf{UE}(Q, A) = \emptyset$.

Proof: Items 1. and 2. are easy corollary of Lemma E.1. Item 3. is proven below by induction on Q.

Nil, Var: $Q = \mathsf{nil}_u$, $Q = x_u$. In these cases $\mathcal{U}(Q, A) = \emptyset$ and $\mathsf{UE}(Q, A) = \emptyset$.

Pref: $Q = \alpha_u.P_1$ or $Q = \underline{\alpha}_u.P_1$. Consider only the latter case (the former is similar to the previous ones). $\mathcal{U}(Q, A) = \emptyset$ if and only if $\alpha \in A$ and, hence, if and only if $\mathsf{UE}(Q, A) = \emptyset$.

Read: $Q = \alpha_{u1} \triangleright_u Q_1$ or $Q = \underline{\alpha}_{u1} \triangleright_u Q_1$. Consider only the latter case (the former is similar to the previous ones). $\mathcal{U}(Q, A) = \emptyset$ if and only if $\alpha \in A$ and $\mathcal{U}(Q_1, A) = \emptyset$ if and only if, by induction hypothesis $\alpha \in A$ and $\mathsf{UE}(Q_1, A) = \emptyset$ if and only if $\mathsf{UE}(Q, A) = \emptyset$.

Sum: $Q = Q_1 +_u Q_2$. $\mathcal{U}(Q, A) = \emptyset$ iff $\mathcal{U}(Q_1, A) = \emptyset$ and $\mathcal{U}(Q_2, A) = \emptyset$, iff, by induction hypothesis, $\mathsf{UE}(Q_1, A) = \emptyset$, $\mathsf{UE}(Q_2, A) = \emptyset$ and $\mathsf{UE}(Q, A) = \emptyset$.

Par: $Q = Q_1 \|_B^u Q_2$. We first prove that $\mathcal{U}(Q_1, A) \cap \mathcal{U}(Q_2, A) \cap B = \emptyset$ if and only if, for each $\alpha \in B \backslash A$, either $\mathcal{U}(Q_1, \mathbb{A}_\tau \backslash \{\alpha\}) = \emptyset$ or $\mathcal{U}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\}) = \emptyset$.

Assume $\mathcal{U}(Q_1, A) \cap \mathcal{U}(Q_2, A) \cap B = \emptyset$ and – toward a contradiction – that there exists $\alpha \in B \backslash A$ such that $\mathcal{U}(Q_1, \mathbb{A}_\tau \backslash \{\alpha\}) \neq \emptyset$ and $\mathcal{U}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\}) \neq \emptyset$. Since $\beta \in \mathcal{U}(Q_i, \mathbb{A}_\tau \backslash \{\alpha\})$ implies (by Proposition A.1-1.) $\beta \notin \mathbb{A}_\tau \backslash \{\alpha\}$ (that is $\beta = \alpha$), we have that $\emptyset \neq \mathcal{U}(Q_i, \mathbb{A}_\tau \backslash \{\alpha\}) \subseteq \{\alpha\}$ and hence $\mathcal{U}(Q_i, \mathbb{A}_\tau \backslash \{\alpha\}) = \{\alpha\}$ for $i = 1, 2$. Thus, $\alpha \in \mathcal{U}(Q_1, \mathbb{A}_\tau \backslash \{\alpha\}) \cap \mathcal{U}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\})$. Finally, since it is also $\alpha \in B \backslash A$, $\alpha \in \mathcal{U}(Q_1, A) \cap \mathcal{U}(Q_2, A) \cap B = \emptyset$ (see Proposition A.1-2.).

To prove the inverse implication we observe that, if $\alpha \in \mathcal{U}(Q_1, A) \cap \mathcal{U}(Q_2, A) \cap B \neq \emptyset$ then $\alpha \in B \backslash A$ (again by Proposition A.1-1.) such that $\alpha \in \mathcal{U}(Q_1, A) \cap \mathcal{U}(Q_2, A)$. Since surely $\alpha \notin \mathbb{A}_\tau \backslash \{\alpha\}$, it is also (by Proposition A.1-2.) $\alpha \in \mathcal{U}(Q_1, \mathbb{A}_\tau \backslash \{\alpha\}) \cap \mathcal{U}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\}) \neq \emptyset$. As above this allows us to conclude that $\mathcal{U}(Q_1, \mathbb{A}_\tau \backslash \{\alpha\}) = \mathcal{U}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\}) = \{\alpha\} \neq \emptyset$

Now, $\mathcal{U}(Q, A) = \emptyset$ iff $\mathcal{U}(Q_1, A \cup B) = \mathcal{U}(Q_2, A \cup B) = \emptyset$ and $\mathcal{U}(Q_1, A) \cap \mathcal{U}(Q_2, A) \cap B = \emptyset$, iff $\mathcal{U}(Q_1, A \cup B) = \mathcal{U}(Q_2, A \cup B) = \emptyset$ and, for each $\alpha \in B \backslash A$, either $\mathcal{U}(Q_1, \mathbb{A}_\tau \backslash \{\alpha\}) = \emptyset$ or $\mathcal{U}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\}) = \emptyset$, iff (by induction hypothesis) $\mathsf{UE}(Q_1, A \cup B) = \mathsf{UE}(Q_2, A \cup B) = \emptyset$ and, for each $\alpha \in B \backslash A$, either $\mathsf{UE}(Q_1, \mathbb{A}_\tau \backslash \{\alpha\}) = \emptyset$ or $\mathsf{UE}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\}) = \emptyset$, iff $\mathsf{UE}(Q_1, A \cup B) = \mathsf{UE}(Q_2, A \cup B) = \emptyset$ and, for each $\alpha \in B \backslash A$, $\mathsf{UE}(Q_1, \mathbb{A}_\tau \backslash \{\alpha\}) \times \mathsf{UE}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\}) = \emptyset$, iff $\mathsf{UE}(Q, A) = \emptyset$.

Rel: $Q = Q_1[\Phi_u]$. $\mathcal{U}(Q, A) = \Phi(\mathcal{U}(Q_1, \Phi^{-1}(A))) = \emptyset$ iff $\mathcal{U}(Q_1, \Phi^{-1}(A)) = \emptyset$ iff, by induction hypothesis, $\mathsf{UE}(Q_1, \Phi^{-1}(A)) = \mathsf{UE}(Q, A) = \emptyset$.

Rec: $Q = \mathsf{rec}\, x_u.Q_1$. In this case $\mathcal{U}(Q, A) = \mathcal{U}(Q_1, A) = \emptyset$ if and only if, by induction hypothesis, $\mathsf{UE}(Q_1, A) = \mathsf{UE}(Q, A) = \emptyset$.

□

F A Proof of Proposition 5.12

This section is devoted to proving Proposition 5.12. A preliminary lemma is needed.

Proposition F.1 Let $Q, \in \mathsf{L}(\tilde{\mathbb{P}})$, $A, X \subseteq \mathbb{A}$. Then $\mathsf{clean}(Q, A) \xrightarrow{X}_r Q'$ implies $Q \xrightarrow{X\backslash A}_r Q'$.

Proof: By induction on $Q \in \mathsf{L}(\tilde{\mathbb{P}})$

Nil, Var: $Q = \mathsf{nil}_u$, $Q = x_u$. The latter case is not possible since $\mathsf{clean}(Q, A) = x_u \not\xrightarrow{X}_r$. Assume $Q = \mathsf{nil}_u$. Then $\mathsf{clean}(Q, A) = \mathsf{nil}_u \xrightarrow{X}_r \mathsf{nil}_u$ and $Q \xrightarrow{X\backslash A}_r \mathsf{nil}_u$.

Pref: $Q = \alpha_u.P_1$ or $Q = \underline{\alpha}_u.P_1$. We prove only the latter case (the former one is similar to the Nil-case). We have two possible subcases:

- $\alpha \in A$. In this case $\mathsf{clean}(Q, A) = \alpha_u.P_1 \xrightarrow{X}_r \underline{\alpha}_u.P_1$. Moreover $\alpha \in A \subseteq \mathbb{A}$ implies $\alpha \notin (X\backslash A) \cup \{\tau\}$ and, by operational rules, $Q \xrightarrow{X\backslash A}_r \underline{\alpha}_u.P_1$.
- $\alpha \notin A$ In this case $\mathsf{clean}(Q, A) = \underline{\alpha}_u.P_1 \xrightarrow{X}_r \underline{\alpha}_u.P_1$ implies $\alpha \notin X \cup \{\tau\} \supseteq (X\backslash A) \cup \{\tau\}$. By operational rules, $Q \xrightarrow{X\backslash A}_r \underline{\alpha}_u.P_1$.

Read: $Q = \alpha_{u1} \rhd_u Q_1$ or $Q = \underline{\alpha}_{u1} \rhd_u Q_1$. We prove only the latter case (the former one is easier). We have two possible subcases:

- $\alpha \in A$. By operational rules, $\mathsf{clean}(Q, A) = \alpha_{u1} \rhd_u \mathsf{clean}(Q_1, A) \xrightarrow{X}_r \underline{\alpha}_{u1} \rhd_u Q_1' = Q$ if $\mathsf{clean}(Q_1, A) \xrightarrow{X}_r Q_1'$. By induction hypothesis $Q_1 \xrightarrow{X\backslash A}_r Q_1'$. Moreover $\alpha \in A \subseteq \mathbb{A}$ implies $\alpha \notin (X\backslash A) \cup \{\tau\}$. Thus, again by operational rules, $Q \xrightarrow{X\backslash A}_r \underline{\alpha}_{u1} \rhd_u Q_1' = Q'$.
- $\alpha \notin A$. In this case $\mathsf{clean}(Q, A) = \underline{\alpha}_{u1} \rhd_u \mathsf{clean}(Q_1, A) \xrightarrow{X}_r \underline{\alpha}_{u1} \rhd_u Q_1' = Q$ if it is both $\alpha \notin X \cup \{\tau\} \supseteq (X\backslash A) \cup \{\tau\}$ and $\mathsf{clean}(Q_1, A) \xrightarrow{X}_r Q_1'$. As in the above case we can prove that $Q \xrightarrow{X\backslash A}_r \underline{\alpha}_{u1} \rhd_u Q_1' = Q'$.

Sum: $Q = Q_1 +_u Q_2$. By operational rules $\mathsf{clean}(Q, A) = \mathsf{clean}(Q_1, A) +_u \mathsf{clean}(Q_2, A) \xrightarrow{X}_r Q'$ implies $\mathsf{clean}(Q_1, A) \xrightarrow{X}_r Q_1'$, $\mathsf{clean}(Q_2, A) \xrightarrow{X}_r Q_2'$ and $Q' = Q_1' +_u Q_2'$. By induction hypothesis $Q_1 \xrightarrow{X\backslash A}_r Q_1'$, $Q_2 \xrightarrow{X\backslash A}_r Q_2'$ and, by operational rules, $Q \xrightarrow{X\backslash A}_r Q'$.

Par: $Q = Q_1 \|_B^u Q_2$. Let $A_1 = B\backslash\mathcal{U}(Q_2)$ and $A_2 = B)\backslash\mathcal{U}(Q_1)$. Let us assume that $\mathsf{clean}(Q, A) = \mathsf{clean}(Q_1, A \cup A_1) \|_B^u \mathsf{clean}(Q_2, A \cup A_2) \xrightarrow{X}_r Q'$. By operational rules, there exist $X_1, X_2 \subseteq \mathbb{A}$ such that $\mathsf{clean}(Q_i, A \cup A_i) \xrightarrow{X_i}_r Q_i'$ for $i = 1, 2$, $X \subseteq (B \cap (X_1 \cup X_2)) \cup ((X_1 \cap X_2)\backslash B)$ and $Q' = \mathsf{clean}(Q_1' \|_B^u Q_2')$. By induction hypothesis $Q_1 \xrightarrow{X_1\backslash(A\cup A_1)}_r Q_1'$ and $Q_2 \xrightarrow{X_2\backslash(A\cup A_2)}_r Q_2'$. Moreover, $A_1 \cap \mathcal{U}(Q_2) = A_2 \cap \mathcal{U}(Q_1) = \emptyset$ and Proposition E.1-1 implies $Q_1 \xrightarrow{X_1'}_r Q_1'$ and $Q_2 \xrightarrow{X_2'}_r Q_2'$, where $X_1' = (X_1\backslash(A \cup A_1)) \cup A_2$ and $X_2' = (X_2\backslash(A \cup A_2)) \cup A_1$. By operational semantics, it remains to prove that $X\backslash A \subseteq (B \cap (X_1' \cup X_2')) \cup ((X_1' \cap X_2')\backslash B)$. Let $\alpha \in X\backslash A$.

- $\alpha \in B$. Then $\alpha \in X$ implies either $\alpha \in X_1$ or $\alpha \in X_2$. Assume $\alpha \in X_1$ (if $\alpha \in X_2$ the statement can be proved similarly). If $\alpha \notin A_1$ then, trivially, $\alpha \in X_1\backslash(A \cup A_1) \subseteq X_1'$. Otherwise, if $\alpha \in A_1$, then $\alpha \in (X_2\backslash(A \cup A_2)) \cup A_1 = X_2'$. In both cases $\alpha \in X_1' \cup X_2'$.
- $\alpha \notin B$. Then $\alpha \in X$ implies both $\alpha \in X_1$ and $\alpha \in X_2$. Moreover $\alpha \notin A_1, A_2 \subseteq B$ implies $\alpha \in X_1\backslash(A \cup A_1) \subseteq X_1'$ and $\alpha \in X_2\backslash(A \cup A_2) \subseteq X_2'$. Thus $\alpha \in X_1' \cap X_2'$.

Rel: $Q = Q_1[\Phi_u]$. Assume that $\mathsf{clean}(Q, A) = \mathsf{clean}(Q_1, \Phi^{-1}(A))[\Phi_u] \xrightarrow{X}_r Q'$. By operational rules, $\mathsf{clean}(Q_1, \Phi^{-1}(A)) \xrightarrow{X'}_r Q_1'$, with $X' = \Phi^{-1}(X \cup \{\tau\})\backslash\{\tau\}$ and $Q' = Q_1'[\Phi_u]$. By induction hypothesis we have that $Q_1 \xrightarrow{X'\backslash\Phi^{-1}(A)}_r Q_1'$. Moreover $X'\backslash\Phi^{-1}(A) =$ $\big(\Phi^{-1}(X \cup \{\tau\})\backslash\{\tau\}\big)\backslash\Phi^{-1}(A) = \big(\Phi^{-1}(X \cup \{\tau\})\backslash\Phi^{-1}(A)\big)\backslash\{\tau\} = \big(\Phi^{-1}((X \cup \{\tau\})\backslash A)\big)\backslash\{\tau\} =$ $\big(\Phi^{-1}((X\backslash A) \cup \{\tau\}\big)\backslash\{\tau\}$. By operational rules $Q \xrightarrow{X\backslash A}_r Q_1'[\Phi_u] = Q'$.

Rec: $Q = \mathsf{rec}\, x_u.Q_1$. $\mathsf{clean}(Q, A) = \mathsf{rec}\, x_u.\mathsf{clean}(Q_1, A) \xrightarrow{X}_r Q'$ implies $\mathsf{clean}(Q_1, A) \xrightarrow{X}_r Q_1'$ and $Q' = \mathsf{rec}\, x_u.Q_1'$. By induction hypothesis $Q_1 \xrightarrow{X\backslash A}_r Q_1'$ and, by operational rules $Q \xrightarrow{X\backslash A}_r \mathsf{rec}\, x_u.Q_1' = Q'$.

□

Lemma F.2 Let $Q, Q', Q'' \in \mathsf{L}(\tilde{\mathbb{P}})$ and $X, X' \subseteq \mathbb{A}$. Then:

1. $Q \xrightarrow{X}_r Q' \xrightarrow{X'}_r Q''$ implies $Q \overset{\mu}{\not\rightarrow}$ and $Q' \overset{\mu}{\not\rightarrow}$ for any $\mu \in X' \cup \{\tau\}$. Moreover $Q' = Q''$;
2. Q action-guarded and $Q \overset{\mu}{\not\rightarrow}$ for any $\mu \in X \cup \{\tau\}$ implies $Q \xrightarrow{X}_r Q'$ and $Q' \overset{\mu}{\not\rightarrow}$ for any $\mu \in X \cup \{\tau\}$.

Proof: We prove these items together by induction on Q.

Var: $Q = x_u$. This case is not possible since $Q \overset{X}{\not\rightarrow}_r$ and Q is not action-guarded.

Nil**:** $Q = \mathsf{nil}_u$.

1. $Q \xrightarrow{X}_r \mathsf{nil}_u = Q' \xrightarrow{X'}_r \mathsf{nil}_u = Q''$, $Q = Q' = \mathsf{nil}_u \overset{\mu}{\not\rightarrow}$ for any $\mu \in X' \cup \{\tau\}$ and, trivially, $\mathsf{nil}_u = \mathsf{nil}_u$.
2. Q is action-guarded, $Q \overset{\mu}{\not\rightarrow}$ for any $\mu \in X \cup \{\tau\}$, $Q \xrightarrow{X}_r \mathsf{nil}_u = Q'$ and $Q' \overset{\mu}{\not\rightarrow}$ for any $\mu \in X \cup \{\tau\}$.

Pref: $Q = \alpha_u.P_1$ or $Q = \underline{\alpha}_u.P_1$. Consider only the latter case (the former case is simpler).

1. $\underline{\alpha}_u.P_1 \xrightarrow{X}_r \underline{\alpha}_u.P_1 = Q' \xrightarrow{X'}_r \underline{\alpha}_u.P_1 = Q''$ implies $\alpha \notin X' \cup \{\tau\}$. Thus, by operational semantics, both Q and $Q' \overset{\mu}{\not\rightarrow}$ for any $\mu \in X' \cup \{\tau\}$. Clearly $Q' = Q''$.
2. In this case Q is action-guarded. Moreover $Q \overset{\mu}{\not\rightarrow}$ for any $\mu \in X \cup \{\tau\}$ implies $\alpha \notin X \cup \{\tau\}$ and, by operational rules, $Q \xrightarrow{X}_r \underline{\alpha}_u.P_1 = Q'$. Again by operational rules, $\alpha \notin X \cup \{\tau\}$ implies $Q' \overset{\mu}{\not\rightarrow}$ for any $\mu \in X \cup \{\tau\}$.

Read: $Q = \alpha_{u1} \rhd_u Q_1$ or $Q = \underline{\alpha}_{u1} \rhd_u Q_1$. Consider only the latter case (the former case is simpler).

1. $\underline{\alpha}_{u1} \rhd_u Q_1 \xrightarrow{X}_r \underline{\alpha}_{u1} \rhd_u Q_1' = Q' \xrightarrow{X'}_r \underline{\alpha}_{u1} \rhd_u Q_1'' = Q''$ implies $\alpha \notin X' \cup \{\tau\}$ and $Q_1 \xrightarrow{X}_r Q_1' \xrightarrow{X'}_r Q_1''$. By induction hypothesis we have that $Q_1, Q_1' \overset{\mu}{\not\rightarrow}$ for any $\mu \in X' \cup \{\tau\}$; moreover $Q_1' = Q_1''$. By operational rules, $\alpha \notin X' \cup \{\tau\}$ and $Q_1, Q_1' \overset{\mu}{\not\rightarrow}$ for any $\mu \in X' \cup \{\tau\}$ implies $Q, Q' \overset{\mu}{\not\rightarrow}$ for any $\mu \in X' \cup \{\tau\}$. Finally $Q_1' = Q_2''$ implies $Q' = Q''$.
2. Q action-guarded implies that also Q_1 it is so. Moreover, $Q \overset{\mu}{\not\rightarrow}$ for any $\mu \in X \cup \{\tau\}$ implies $\alpha \notin X \cup \{\tau\}$ and $Q_1 \overset{\mu}{\not\rightarrow}$ for any $\mu \in X \cup \{\tau\}$. By induction hypothesis, $Q_1 \xrightarrow{X}_r Q_1'$ and $Q_1' \overset{\mu}{\not\rightarrow}$ for any $\mu \in X \cup \{\tau\}$. By the operational rules, $\alpha \notin X \cup \{\tau\}$ and $Q_1 \xrightarrow{X}_r Q_1'$ implies $Q = \underline{\alpha}_{u1} \rhd_u Q_1 \xrightarrow{X}_r \underline{\alpha}_{u1} \rhd_u Q_1' = Q'$. Finally, $\alpha \notin X \cup \{\tau\}$ and $Q_1' \overset{\mu}{\not\rightarrow}$ for any $\mu \in X \cup \{\tau\}$ implies that $Q' = \underline{\alpha}_{u1} \rhd_u Q_1' \overset{\mu}{\not\rightarrow}$ for any $\mu \in X \cup \{\tau\}$.

Sum: $Q = Q_1 +_u Q_2$.

1. If $Q \xrightarrow{X}_r Q' \xrightarrow{X'}_r Q''$ then $Q_1 \xrightarrow{X}_r Q_1' \xrightarrow{X'}_r Q_1''$, $Q_2 \xrightarrow{X}_r Q_2' \xrightarrow{X'}_r Q_2''$, $Q' = Q_1' +_u Q_2'$ and $Q'' = Q_1'' +_u Q_2''$. By induction hypothesis $Q_1 \not\xrightarrow{\mu}$, $Q_1' \not\xrightarrow{\mu}$ and $Q_2 \not\xrightarrow{\mu}$, $Q_2' \not\xrightarrow{\mu}$ for any $\mu \in X \cup \{\tau\}$ and, hence, by operational rules, both $Q \not\xrightarrow{\mu}$ and $Q' \not\xrightarrow{\mu}$ for any $\mu \in X' \cup \{\tau\}$. Again by induction hypothesis, $Q_1' = Q_1''$ and $Q_2' = Q_2''$. Thus $Q' = Q''$.
2. Q guarded implies both Q_1 and Q_2 guarded. Assume $Q \not\xrightarrow{\mu}$ for any $\mu \in X \cup \{\tau\}$. Then, by operational rules, $Q_1 \not\xrightarrow{\mu}$ and $Q_2 \not\xrightarrow{\mu}$ for any $\mu \in X \cup \{\tau\}$. By induction hypothesis, $Q_1 \xrightarrow{X}_r Q_1'$, $Q_2 \xrightarrow{X}_r Q_2'$ and, hence, $Q \xrightarrow{X}_r Q_1' +_u Q_2' = Q'$. Moreover, again by induction hypothesis, we have that both Q_1' and $Q_2' \not\xrightarrow{\mu}$ for any $\mu \in X \cup \{\tau\}$. Then, by operational rules, also $Q' \not\xrightarrow{\mu}$ for any $\mu \in X \cup \{\tau\}$.

Par: $Q = Q_1 \parallel_B^u Q_2$.

1. $Q \xrightarrow{X}_r Q'$ implies that there exist X_1, X_2 such that $Q_1 \xrightarrow{X_1}_r Q_1'$, $Q_2 \xrightarrow{X_2}_r Q_2'$ with $X \subseteq (B \cap (X_1 \cup X_2)) \cup ((X_1 \cap X_2) \backslash B)$ and $Q' = \mathsf{clean}(Q_1' \parallel_B^u Q_2')$. Let $A_1 = B \backslash \mathcal{U}(Q_2')$ and $A_2 = B \backslash \mathcal{U}(Q_1')$ and assume $Q' = \mathsf{clean}(Q_1', A_1) \parallel_B^u \mathsf{clean}(Q_2', A_2) \xrightarrow{X'}_r Q''$. Again, there exist X_1', X_2' such that $\mathsf{clean}(Q_1', A_1) \xrightarrow{X_1'}_r Q_1''$, $\mathsf{clean}(Q_2', A_2) \xrightarrow{X_2'}_r Q_2''$ with $X' \subseteq (B \cap (X_1' \cup X_2')) \cup ((X_1' \cap X_2') \backslash B)$ and $Q'' = \mathsf{clean}(Q_1'' \parallel_B^u Q_2'')$. If $\mathsf{clean}(Q_1', A_1) \xrightarrow{X_1'}_r Q_1''$ then $Q_1' \xrightarrow{X_1' \backslash A_1}_r Q_1''$ (by Proposition F.1). Moreover $A_2 \cap \mathcal{U}(Q_1') = \emptyset$ and Proposition E.1-1 imply $Q_1' \xrightarrow{(X_1' \backslash A_1) \cup A_2}_r Q_1''$. Similarly $Q_2' \xrightarrow{(X_2' \backslash A_2) \cup A_1}_r Q_2''$. By induction hypothesis $Q_1 \xrightarrow{X_1}_r Q_1' \xrightarrow{(X_1' \backslash A_1) \cup A_2}_r Q_1''$ and $Q_2 \xrightarrow{X_2}_r Q_2' \xrightarrow{(X_2' \backslash A_2) \cup A_1}_r Q_2''$ imply (i) $Q_1, Q_1' \not\xrightarrow{\mu}$ for any $\mu \in (X_1' \backslash A_1) \cup A_2 \cup \{\tau\}$ and (ii) $Q_2, Q_2' \not\xrightarrow{\mu}$ for any $\mu \in (X_2' \backslash A_2) \cup A_1 \cup \{\tau\}$.
   Now we prove that $Q' \not\xrightarrow{\mu}$ (and similarly Q) for any $\mu \in X' \cup \{\tau\}$. First $Q_1' \not\xrightarrow{\tau}$ and $Q_2' \not\xrightarrow{\tau}$ imply $Q' \not\xrightarrow{\tau}$. Let $\mu \in X'$ and consider the following subcases:
   - $\mu \in B$. Then $\mu \in X$ implies $\mu \in X_1' \cup X_2'$ and, hence, either $\mu \in X_1'$ or $\mu \in X_2'$. Assume $\mu \in X_1'$ (the case in which $\mu \in X_2$ can be proved similarly). If $\mu \in A_1$ then, by (ii), $Q_2' \not\xrightarrow{\mu}$. If $\mu \notin A_1$ and, hence, $\mu \in X_1' \backslash A_1$, then, by (i), $Q_1' \not\xrightarrow{\mu}$. In both cases $Q' \not\xrightarrow{\mu}$.
   - $\mu \notin B$. In this case $\mu \in X_1'$ and $\mu \in X_2'$. Moreover $\mu \notin A_1, A_2 \subseteq B$ Thus $\mu \in X_1' \backslash A_1$ and $\mu \in X_2' \backslash A_2$ imply, by (i) and (ii), $Q_1' \not\xrightarrow{\mu}$, $Q_2' \not\xrightarrow{\mu}$ and, hence, $Q' \not\xrightarrow{\mu}$.

   Again by induction hypothesis we have $Q_1' = Q_1''$ and $Q_2' = Q_2''$. Then also $Q' = Q''$.
2. Assume Q guarded and, hence, both Q_1 and Q_2 guarded. Now, assume $Q \not\xrightarrow{\mu}$ for any $\mu \in X \cup \{\tau\}$. By operational semantics we have that: (i) $Q_1 \not\xrightarrow{\mu}$ and $Q_2 \not\xrightarrow{\mu}$, for any $\mu \in (X \backslash B) \cup \{\tau\}$ and (ii) for any $\mu \in X \cap B$ either $Q_1 \not\xrightarrow{\mu}$ or $Q_2 \not\xrightarrow{\mu}$. Let $X_i' = \{\mu \in X \cap B \mid Q_i \not\xrightarrow{\mu}\} \subseteq X \cap B \subseteq B$ and $X_i = (X \backslash B) \cup X_i'$. Then, Q_i guarded and $Q_i \not\xrightarrow{\mu}$ for any $\mu \in X_i \cup \{\tau\}$ implies, by induction hypothesis, $Q_i \xrightarrow{X_i} Q_i'$ for $i = 1, 2$. Moreover, $B \cap (X_1 \cup X_2) = B \cap ((X \backslash B) \cup X_1' \cup X_2') = X_1' \cup X_2'$, $(X_1 \cap X_2) \backslash B = (X_1 \backslash B) \cup (X_2 \backslash B) = (X \backslash B) \cup (X \backslash B) = X \backslash B$ and, by (ii), $X \cap B = X_1' \cup X_2'$.
   Finally $(B \cap (X_1 \cup X_2)) \cup ((X_1 \cap X_2) \backslash B) = (X \cap B) \cup (X \backslash B) = X$ and, by operational rules, $Q \xrightarrow{X} \mathsf{clean}(Q_1' \parallel_B^u Q_2') = Q'$. Again by induction hypothesis we have that $Q_1' \not\xrightarrow{\mu}$ for any $\mu \in X_1 \cup \{\tau\}$ and $Q_2' \not\xrightarrow{\mu}$ for any $\mu \in X_2 \cup \{\tau\}$. Also in this case, $Q_1' \not\xrightarrow{\tau}$ and $Q_2' \not\xrightarrow{\tau}$ imply $Q' \not\xrightarrow{\tau}$. Now, let $\mu \in X$. If $\mu \in X \backslash B$ then $\mu \in X_1$ and $\mu \in X_2$ implies both $Q_1' \not\xrightarrow{\mu}$ and $Q_2' \not\xrightarrow{\mu}$ and, hence, $Q' \not\xrightarrow{\tau}$. If $\mu \in X \cap B = X_1' \cup X_2'$ we have either $\mu \in X_1' \subseteq X_1$ or $\mu \in X_2' \subseteq X_2$. Thus, either $Q_1' \not\xrightarrow{\mu}$ or $Q_2' \not\xrightarrow{\mu}$. Also in this case $Q' \not\xrightarrow{\tau}$.

Rel: $Q = Q_1[\Phi_u]$. Let $X, X' \subseteq \mathbb{A}$, $Y = \Phi^{-1}(X \cup \{\tau\})\backslash\{\tau\}$ and $Y' = \Phi^{-1}(X' \cup \{\tau\})\backslash\{\tau\}$. Then $\Phi(\tau) = \tau$ implies $\tau \in \Phi^{-1}(X \cup \{\tau\})$ and $\Phi^{-1}(X \cup \{\tau\}) = (\Phi^{-1}(X \cup \{\tau\})\backslash\{\tau\}) \cup \{\tau\} = Y \cup \{\tau\}$. Similarly, we have $\Phi^{-1}(X' \cup \{\tau\}) = Y' \cup \{\tau\}$.

1. By operational rules $Q \xrightarrow{X}_r Q' \xrightarrow{X'}_r Q''$ implies $Q_1 \xrightarrow{Y}_r Q'_1 \xrightarrow{Y'}_r Q''_1$, $Q' = Q'_1[\Phi_u]$ and $Q'' = Q''_1[\Phi_u]$. By induction hypothesis $Q_1, Q'_1 \overset{\underline{\mu'}}{\not\longrightarrow}$ for any $\mu' \in Y' \cup \{\tau\} = \Phi^{-1}(X' \cup \{\tau\})$ and, hence, $Q, Q' \overset{\underline{\mu}}{\not\longrightarrow}$ for any $\mu \in X' \cup \{\tau\}$. Again by induction hypothesis $Q'_1 = Q''_1$ and, hence, also $Q' = Q''$.
2. Q guarded implies Q_1 guarded. Now, assume $Q \overset{\underline{\mu}}{\not\longrightarrow}$ for any $\mu \in X \cup \{\tau\}$. Then $Q_1 \overset{\underline{\mu'}}{\not\longrightarrow}$ for any $\mu' \in \Phi^{-1}(X \cup \{\tau\}) = Y \cup \{\tau\}$. By induction hypothesis $Q_1 \xrightarrow{Y}_r Q'_1$ and, hence, $Q \xrightarrow{X}_r Q'_1[\Phi] = Q'$. Again by induction hypothesis we have that $Q'_1 \overset{\underline{\mu'}}{\not\longrightarrow}$ for any $\mu' \in \Phi^{-1}(X \cup \{\tau\})$ and, by operational semantics, $Q' \overset{\underline{\mu}}{\not\longrightarrow}$ for any $\mu \in X \cup \{\tau\}$.

Rec: $Q = \mathsf{rec}\, x_u.Q_1$

1. $Q \xrightarrow{X}_r \mathsf{rec}\, x_u.Q'_1 = Q' \xrightarrow{X'}_r \mathsf{rec}\, x_u.Q''_1 = Q''$ implies $Q_1 \xrightarrow{X}_r Q'_1 \xrightarrow{X'}_r Q''_1$. By induction hypothesis, $Q_1, Q'_1 \overset{\underline{\mu}}{\not\longrightarrow}$ for any $\mu \in X' \cup \{\tau\}$. Thus, x guarded in Q_1 and, hence, in Q'_1 imply, by Proposition A.4-3, $Q_1\{\!|\mathsf{rec} x_u.\mathsf{unmark}(Q_1)/x|\!\} \overset{\underline{\mu}}{\not\longrightarrow} Q'_1\{\!|\mathsf{rec} x_u.\mathsf{unmark}(Q'_1)/x|\!\} \overset{\underline{\mu}}{\not\longrightarrow}$ for any $\mu \in X' \cup \{\tau\}$. Finally, by operational semantics, $Q, Q' \overset{\underline{\mu}}{\not\longrightarrow}$ for any $\mu \in X \cup \{\tau\}$. Again by induction hypothesis, we also have $Q'_1 = Q''_1$ and, clearly, $Q' = Q''$.
2. Assume Q and, hence, Q_1 action-guarded. In this case $Q \overset{\underline{\mu}}{\not\longrightarrow}$ for any $\mu \in X \cup \{\tau\}$ implies $Q_1\{\!|\mathsf{rec}\, x_u.\mathsf{unmark}(Q_1)/x|\!\} \overset{\underline{\mu}}{\not\longrightarrow}$ and, since x is action-guarded in Q_1, also $Q_1 \overset{\underline{\mu}}{\not\longrightarrow}$ for any $\mu \in X \cup \{\tau\}$ (see Proposition A.4-3). Then, by induction hypothesis, $Q_1 \xrightarrow{X}_r Q'_1$ and, by operational semantics, $Q \xrightarrow{X}_r \mathsf{rec}\, x_u.Q'_1 = Q'$. Moreover, again by induction hypothesis, $Q'_1 \overset{\underline{\mu}}{\not\longrightarrow}$ for any $\mu \in X \cup \{\tau\}$. Then, since x action-guarded in Q_1 implies x action-guarded in Q'_1, by Proposition A.4-3 we have that $Q'_1\{\!|\mathsf{rec}\, x_u.\mathsf{unmark}(Q'_1)/x|\!\} \overset{\underline{\mu}}{\not\longrightarrow}$ for any $\mu \in X \cup \{\tau\}$. Finally, by operational rules, $Q' \overset{\underline{\mu}}{\not\longrightarrow}$ for any $\mu \in X \cup \{\tau\}$.

□

Proposition 5.12 Let $Q, Q', Q'' \in \mathsf{L}(\tilde{\mathbb{P}})$.

1. $Q \xrightarrow{1} Q' \xrightarrow{1} Q''$ implies $Q \overset{\underline{\mu}}{\not\longrightarrow}$ and $Q' \overset{\underline{\mu}}{\not\longrightarrow}$ for any $\mu \in \mathbb{A}_\tau$. Moreover $Q' = Q''$;
2. Q guarded and $Q \overset{\underline{\mu}}{\not\longrightarrow}$ for any $\mu \in \mathbb{A}_\tau$ implies $Q \xrightarrow{1} Q' \xrightarrow{1} Q'$

Proof:

1. If $Q \xrightarrow{1} Q' \xrightarrow{1} Q''$ then, by Lemma F.2-1 and $Q, Q' \overset{\underline{\mu}}{\not\longrightarrow}$ for any $\mu \in \mathbb{A}_\tau$ and $Q' = Q''$.
2. By Lemma F.2-2, Q guarded and $Q \overset{\underline{\mu}}{\not\longrightarrow}$ for any $\mu \in \mathbb{A}_\tau$ implies that $Q \xrightarrow{1} Q'$ and $Q' \overset{\underline{\mu}}{\not\longrightarrow}$ for any $\mu \in \mathbb{A}_\tau$. Moreover since Q guarded implies also Q' guarded, again by Lemma F.2-2, there exists Q'' such that $Q' \xrightarrow{1} Q''$. By Item 1, $Q' = Q''$ and the statement follows.

□

G A Proof of Proposition 6.4

Proposition 6.4 Let $Q \in \mathsf{L}(\tilde{\mathbb{P}})$ and $P \in \mathsf{L}(\tilde{\mathbb{P}}_1)$ such that $P = \mathsf{unmark}(Q)$. Then:

1. $\mathsf{LE}(Q, A) = \mathsf{LE}(P, A)$ for every A;
2. $Q \xrightarrow{\alpha} Q'$ implies $P \xrightarrow{\alpha} P'$ and $P' = \mathsf{unmark}(Q')$ for some P'. Moreover $\mathsf{UE}(Q', A) \subseteq \mathsf{UE}(Q, A)$ and, whenever Q' is clean and $\mathsf{UE}(Q') = \emptyset$, we have $Q' = P'$;
3. $P \xrightarrow{\alpha} P'$ implies $Q \xrightarrow{\alpha} Q'$ and $P' = \mathsf{unmark}(Q')$ for some Q'.

Proof: Let us first consider Item 2. We prove this items indirectly by providing a proof of the following two statements:

2.1 $Q \overset{\alpha}{\rightsquigarrow} Q'$ implies $P \overset{\alpha}{\rightsquigarrow} P'$ for some P'. Notice that if $Q \overset{\alpha}{\rightsquigarrow} Q'$ and $P \overset{\alpha}{\rightsquigarrow} P'$ then $Q = Q'$ and $P' = P$ trivially implies $P' = \mathsf{unmark}(Q')$. Moreover, $\mathsf{UE}(Q', A) = \mathsf{UE}(Q, A)$ and whenever $\mathsf{UE}(Q') = \emptyset$ and Q' is clean then—by Propositions 5.11-3. and 2.10-2.—we have $\mathcal{U}(Q') = \emptyset$, $Q' \in \tilde{\mathbb{P}}_1$ and, finally, $P' = \mathsf{unmark}(Q') = Q'$.

2.2 $Q \overset{\alpha}{\mapsto} Q'$ implies $P \overset{\alpha}{\mapsto} P'$ and $P' = \mathsf{unmark}(Q')$ for some P'. Moreover $\mathsf{UE}(Q', A) \subseteq \mathsf{UE}(Q, A)$ and $\mathsf{UE}(Q') = \emptyset$ imples $Q' = P'$ (in this case the addition assumption that Q' is clean is not needed).

We prove statement 2.1 and Item 1 by induction on Q, while statement 2.2 will be proven by induction of length of derivation $Q \overset{\mu}{\mapsto} Q'$. The proof for Item 3 is similar to the one for 2 and hence omitted. We proceed by case analysis on the structure of Q.

Nil, Var: $Q = \mathsf{nil}_u$, $Q = x_u$. In both cases $P = \mathsf{unmark}(Q)$ implies $P = Q$ and items 1. and 2. hold trivially.

Pref: $Q = \mu_u.P_1$ with $\mu \in \{\beta, \underline{\beta}\}$. In both case $P = \mathsf{unmark}(Q)$ implies $P = \beta_u.P_1$. Then:

1. $\mathsf{LE}(P, A) = \mathsf{LE}(Q, A) = \{\langle u \rangle\}$ if $\beta \notin A$, $\mathsf{LE}(P, A) = \mathsf{LE}(Q, A) = \emptyset$ otherwise.

2.1 This case is not possible since $Q \overset{\alpha}{\not\rightsquigarrow}$.

2.2 $Q \overset{\alpha}{\mapsto} P_1$ implies $\alpha = \beta$ and hence $P \overset{\beta}{\mapsto} P_1$; since $P_1 \in \mathsf{L}(\tilde{\mathbb{P}}_1)$, it is also $P_1 = \mathsf{unmark}(P_1)$. Moreover we have that $\mathsf{UE}(P_1, A) = \emptyset \subseteq \mathsf{UE}(Q, A)$, $\mathsf{UE}(P_1) = \emptyset$ and, sure, $P_1 = P_1$.

Read: $Q = \mu_{u1} \rhd_u Q_1$ with $\mu \in \{\beta, \underline{\beta}\}$. In both case $P = \mathsf{unmark}(Q)$ implies $P = \beta_{u1} \rhd_u P_1$ where $P_1 = \mathsf{unmark}(Q_1)$. Then:

1. $\mathsf{LE}(P, A) = \{\langle u1 \rangle\} \cup \mathsf{LE}(P_1, A) = \{\langle u1 \rangle\} \cup \mathsf{LE}(Q_1, A) = \mathsf{LE}(Q, A)$ if $\beta \notin A$, $\mathsf{LE}(P, A) = \mathsf{LE}(P_1, A) = \mathsf{LE}(Q_1, A)\mathsf{LE}(Q, A)$ otherwise.

2.1 $Q \overset{\alpha}{\rightsquigarrow} Q'$ if either $\alpha = \beta$ and $Q' = Q$ or $Q_1 \overset{\alpha}{\rightsquigarrow} Q_1'$. In the former case $P \overset{\alpha}{\rightsquigarrow} P$ and the statement follows easily. In the latter one, by induction hypothesis, $P_1 \overset{\alpha}{\rightsquigarrow} P_1'$ with $P_1' = \mathsf{unmark}(Q_1')$. Thus, $P \overset{\alpha}{\rightsquigarrow} \beta_{u1} \rhd_u P_1' = P'$, with $P' = \beta_{u1} \rhd_u \mathsf{unmark}(Q_1') = \mathsf{unmark}(Q')$.

2.2 $Q \overset{\alpha}{\mapsto} P_1$ implies $\alpha = \beta$ and hence $P \overset{\beta}{\mapsto} P_1$; since $P_1 \in \mathsf{L}(\tilde{\mathbb{P}}_1)$, it is also $P_1 = \mathsf{unmark}(P_1)$. Moreover we have that $\mathsf{UE}(P_1, A) = \emptyset \subseteq \mathsf{UE}(Q, A)$, $\mathsf{UE}(P_1) = \emptyset$ and, sure, $P_1 = P_1$.

Sum: $Q = Q_1 +_u Q_2$. In this case, $P = \mathsf{unmark}(Q)$ implies $P_i = \mathsf{unmark}(Q_i)$, for $i = 1, 2$, and $P = P_1 +_u P_2$.

1. By induction hypothesis $\mathsf{LE}(P, A) = \mathsf{LE}(P_1, A) \cup \mathsf{LE}(P_2, A) = \mathsf{LE}(Q_1, A) \cup \mathsf{LE}(Q_2, A) = \mathsf{LE}(Q, A)$, for every A.

2.1 By operational rules, $Q \overset{\alpha}{\rightsquigarrow} Q'$ implies either (i) $Q_1 \overset{\alpha}{\mapsto} Q_1'$ and $Q' = Q_1' + Q_2$ or (ii) $Q_2 \overset{\alpha}{\mapsto} Q_2'$ and $Q' = Q_1 + Q_2'$. We only prove the former case (the latter one is similar). By induction hypothesis, it is $P_1 \overset{\alpha}{\mapsto} P_1'$ with $P_1' = \mathsf{unmark}(Q_1')$. Thus, $P \overset{\alpha}{\mapsto} P_1' + P_2 = P'$ with $P' = \mathsf{unmark}(Q_1') + \mathsf{unmark}(Q_2) = \mathsf{unmark}(Q_1' + Q_2) = \mathsf{unmark}(Q')$.

2.2 By operational rules, $Q \overset{\alpha}{\mapsto} Q'$ if either (i) $Q_1 \overset{\alpha}{\mapsto} Q'$ or (ii) $Q_2 \overset{\alpha}{\mapsto} Q'$. Consider the (i)-case (the other one is similar). By induction hypothesis, $Q_1 \overset{\alpha}{\mapsto} Q'$ implies $P_1 \overset{\alpha}{\mapsto} P'$ (i.e., again by operational rules $P \xrightarrow{\alpha} P'$) and $P' = \mathsf{unmark}(Q')$ for some P'. Moreover, again by induction hypothesis, $\mathsf{UE}(Q', A) \subseteq \mathsf{UE}(Q_1, A) \subseteq \mathsf{UE}(Q, A)$ and $\mathsf{UE}(Q') = \emptyset$ implies $Q' = P'$.

Par: $Q = Q_1 \|_B^u Q_2$. By definition $\mathsf{unmark}(Q) = \mathsf{unmark}(Q_1) \|_B^u \mathsf{unmark}(Q_2)$. Thus $P = P_1 \|_B^u P_2$, where $P_i = \mathsf{unmark}(Q_i)$ for every $i = 1, 2$. Then:

1. By induction hypothesis, $\mathsf{LE}(P_i, A \cup B) \cup \mathsf{LE}(Q_i, A \cup B)$ for $i = 1, 2$, and, for each $\alpha \in B \backslash A$, $\mathsf{LE}(P_1, \mathbb{A}_\tau \backslash \{\alpha\}) \times \mathsf{LE}(P_2, \mathbb{A}_\tau \backslash \{\alpha\}) = \mathsf{LE}(Q_1, \mathbb{A}_\tau \backslash \{\alpha\}) \times \mathsf{LE}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\})$. By Definition 5.7 we can conclude that $\mathsf{LE}(P, A) = \mathsf{LE}(Q, A)$.

2.1 Assume $Q \overset{\alpha}{\rightsquigarrow} Q'$ and consider the following three possible cases:

i. $\alpha \in B$ and $Q_i \overset{\alpha}{\rightsquigarrow} Q_i'$ for $i = 1, 2$ and $Q' = Q_1' \|_B^u Q_2'$. By induction hypothesis $P_i \overset{\alpha}{\rightsquigarrow} P_i'$ with $P_i' = \mathsf{unmark}(Q_i')$, for every i. Then: $P \overset{\alpha}{\rightsquigarrow} P_1' \|_B^u P_2' = P'$ and $P' = \mathsf{unmark}(Q')$.

ii. $\alpha \notin B$ and $Q_1 \overset{\alpha}{\rightsquigarrow} Q_1'$ and $Q' = Q_1' \|_B^u Q_2$. By induction hypothesis $P_1 \overset{\alpha}{\rightsquigarrow} P_1'$ with $P_1' = \mathsf{unmark}(Q_1')$. Then: $P \overset{\alpha}{\rightsquigarrow} P_1' \|_B^u P_2 = P'$ and $P' = \mathsf{unmark}(Q')$

iii. $\alpha \notin B$ and $Q_2 \overset{\alpha}{\rightsquigarrow} Q_2'$ and $Q' = Q_1 \|_B^u Q_2'$. Similar to the previous one.

2.2 Assume $Q \overset{\alpha}{\mapsto} Q'$ and consider the following three possible cases:

i. $\alpha \in B$ and $Q_i \overset{\alpha}{\mapsto} Q_i'$ for $i = 1, 2$ and $Q' = \mathsf{clean}(Q_1' \|_B^u Q_2')$. By induction hypothesis $P_i \overset{\alpha}{\mapsto} P_i'$ with $P_i' = \mathsf{unmark}(Q_i')$, $\mathsf{UE}(Q_i', A) \subseteq \mathsf{UE}(Q_i, A)$ and $\mathsf{UE}(Q_i') = \emptyset$ implies $Q_i' = P_i'$, for every i and A. Then: $P \overset{\alpha}{\mapsto} \mathsf{clean}(P_1' \|_B^u P_2') = P_1' \|_B^u P_2' = P'$ and $P' = \mathsf{unmark}(Q')$. Moreover, by Lemma A.11, it is $\mathsf{UE}(Q', A) = \mathsf{UE}(Q_1' \|_B^u Q_2', A) = \mathsf{UE}(Q_1', A \cup B) \cup \mathsf{UE}(Q_2', A \cup B) \cup \bigcup_{\alpha \in B \backslash A} (\mathsf{UE}(Q_1', \mathbb{A}_\tau \backslash \{\alpha\}) \times \mathsf{UE}(Q_2', \mathbb{A}_\tau \backslash \{\alpha\}) \subseteq \mathsf{UE}(Q_1, A \cup B) \cup \mathsf{UE}(Q_2, A \cup B) \cup \bigcup_{\alpha \in B \backslash A} (\mathsf{UE}(Q_1, \mathbb{A}_\tau \backslash \{\alpha\}) \times \mathsf{UE}(Q_2, \mathbb{A}_\tau \backslash \{\alpha\}) = \mathsf{UE}(Q, A)$.

   Now assume $\mathsf{UE}(Q') = \mathsf{UE}(Q_1' \|_B^u Q_2', A) = \emptyset$. By Propositions 5.11-3 and A.5-3, we also have $\mathcal{U}(Q_1' \|_B^u Q_2') = \emptyset$ and $Q' = \mathsf{clean}(Q_1' \|_B^u Q_2') = \mathsf{unmark}(Q_1' \|_B^u Q_2') = P_1' \|_B^u P_2' = P'$.

ii. $\alpha \notin B$ and $Q_1 \overset{\alpha}{\mapsto} Q_1'$ and $Q' = \mathsf{clean}(Q_1' \|_B^u Q_2)$. By induction hypothesis $P_1 \overset{\alpha}{\mapsto} P_1'$ with $P_1' = \mathsf{unmark}(Q_1')$, $\mathsf{UE}(Q_1', A) \subseteq \mathsf{UE}(Q_1, A)$ for every A and $\mathsf{UE}(Q_1') = \emptyset$ implies $Q_1' = P_1'$. Then: $P \overset{\alpha}{\mapsto} \mathsf{clean}(P_1' \|_B^u P_2) = P_1' \|_B^u P_2 = P'$ and $P' = \mathsf{unmark}(Q')$. Again by Lemma A.11, it is $\mathsf{UE}(Q', A) = \mathsf{UE}(Q_1' \|_B^u Q_2, A)$ and as in the previous case we can prove that $\mathsf{UE}(Q_1' \|_B^u Q_2, A) \subseteq \mathsf{UE}(Q, A)$.

   Now assume $\mathsf{UE}(Q') = \mathsf{UE}(Q_1' \|_B^u Q_2, A) = \emptyset$. By Propositions 5.11-3 and A.5-3, we also have $\mathcal{U}(Q_1' \|_B^u Q_2) = \emptyset$ and $Q' = \mathsf{clean}(Q_1' \|_B^u Q_2) = \mathsf{unmark}(Q_1' \|_B^u Q_2) = P_1' \|_B^u P_2 = P'$.

iii. $\alpha \notin B$ and $Q_2 \overset{\alpha}{\mapsto} Q_2'$ and $Q' = \mathsf{clean}(Q_1 \|_B^u Q_2')$. Similar to the previous one.

Rel: $Q = Q_1[\Phi_u]$. In this case $P = \mathsf{unmark}(Q)$ implies $P = P_1[\Phi_u]$ with $P_1 = \mathsf{unmark}(Q_1)$. Then:

1. By induction hypothesis $\mathsf{LE}(P, A) = \mathsf{LE}(P_1, \Phi^{-1}(A)) = \mathsf{LE}(Q_1, \Phi^{-1}(A)) = \mathsf{LE}(Q, A)$.

2.1 By operational rules, $Q \overset{\alpha}{\rightsquigarrow} Q_1'[\Phi_u] = Q'$ if there exists $\beta \in \Phi^{-1}(\alpha)$ such that $Q_1 \overset{\beta}{\rightsquigarrow} Q_1'$. By induction hypothesis $P_1 \overset{\beta}{\rightsquigarrow} P_1'$ and $P_1' = \mathsf{unmark}(Q_1')$ which implies $P \overset{\alpha}{\rightsquigarrow} P_1'[\Phi_u] = P'$ and $P' = \mathsf{unmark}(Q_1')[\Phi_u] = \mathsf{unmark}(Q')$.

2.2 By operational rules, $Q \overset{\alpha}{\mapsto} Q_1'[\Phi_u] = Q'$ if there exists $\beta \in \Phi^{-1}(\alpha)$ such that $Q_1 \overset{\beta}{\mapsto} Q_1'$. By induction hypothesis $P_1 \overset{\beta}{\mapsto} P_1'$ and $P_1' = \mathsf{unmark}(Q_1')$ which implies $P \overset{\alpha}{\mapsto} P_1'[\Phi_u] = P'$ and $P' = \mathsf{unmark}(Q_1')[\Phi_u] = \mathsf{unmark}(Q')$. Again by induction hypothesis $\mathsf{UE}(Q', A) = \mathsf{UE}(Q_1', \Phi^{-1}(A)) \subseteq \mathsf{UE}(Q_1, \Phi^{-1}(A)) = \mathsf{UE}(Q, A)$ and $\mathsf{UE}(Q') = \mathsf{UE}(Q_1') = \emptyset$ implies $Q_1' = P_1'$ and, hence, $Q' = P'$.

Rec: $Q = \mathsf{rec}\, x_u.Q_1$. In this case $P = \mathsf{unmark}(Q)$ implies $P = \mathsf{rec}\, x_u.P_1$ with $P_1 = \mathsf{unmark}(Q_1)$.

1. By induction hypothesis, $\mathsf{LE}(P, A) = \mathsf{LE}(P_1, A) = \mathsf{LE}(Q_1, A) = \mathsf{LE}(Q, A)$.

2.1 $Q \overset{\alpha}{\rightsquigarrow} \mathsf{rec}\, x.Q_1' = Q'$ if $Q_1 \overset{\alpha}{\rightsquigarrow} Q_1'$. By induction hypothesis we have that $P_1 \overset{\alpha}{\rightsquigarrow} P_1'$ and $P_1' = \mathsf{unmark}(Q_1')$. Thus, $P \overset{\alpha}{\rightsquigarrow} \mathsf{rec}\, x.P_1' = P'$ and $P' = \mathsf{rec}\, x.\mathsf{unmark}(Q_1') = \mathsf{unmark}(Q')$.

2.2 Let $R = Q_1\{\!|\mathsf{rec}\, x_u.\mathsf{unmark}(Q_1)/x|\!\} = Q_1\{\!|\mathsf{rec}\, x_u.P_1/x|\!\}$ and $S = \mathsf{unmark}(R)$. x action-guarded in Q_1 and Propositions A.4-2 imply $\mathsf{UE}(R, A) = \mathsf{UE}(Q_1, A) = \mathsf{UE}(Q, A)$; moreover, by Proposition A.7-2, x-action-guarded in Q_1 also implies $S = \mathsf{unmark}(R) = \mathsf{unmark}(Q_1)\{\!|\mathsf{rec}\, x_u.P_1/x|\!\} = P_1\{\!|\mathsf{rec}\, x_u.P_1/x|\!\}$.

Now assume $Q \overset{\alpha}{\mapsto} Q'$ and hence, by operational rules, that $R \overset{\alpha}{\mapsto} Q'$. By induction hypothesis $S = \mathsf{unmark}(R) \overset{\alpha}{\mapsto} P'$—by operational rule this also implies $P \overset{\alpha}{\mapsto} P'$—for some P' with $P' = \mathsf{unmark}(Q')$, $\mathsf{UE}(Q', A) \subseteq \mathcal{U}(R, A) = \mathsf{UE}(Q, A)$ for every A and $\mathsf{UE}(Q') = \emptyset$ implies $Q' = P'$.

□

H Proof of Theorem 8.2

We start with two lemmas:

Lemma H.1 Let $P, P', P'' \in \tilde{\mathbb{P}}_1$, $Q' \in \tilde{\mathbb{P}}$, $v, w \in \mathbb{A}_\tau$ such that $P \xrightarrow{v} P' \xrightarrow{1} Q' \xrightarrow{w} P''$. Then there exists $Q \in \tilde{\mathbb{P}}$ such that $P \xrightarrow{1} Q \xrightarrow{vw} P''$.

Proof: Let $S \in \mathsf{L}(P)$. By repeated applications of Proposition A.3-(ii), if $P \xrightarrow{v} P'$ (in the unlabelled PAFAS$_r$) then $S \xrightarrow{v} S'$ for some S' with $P' = \mathsf{R}(S')$ (and hence for some $S' \in \mathsf{L}(P')$).

Again due to repeated applications of Proposition A.3-(ii), if $P' \xrightarrow{1} Q' \xrightarrow{w} P''$ then there exists $R' \in \mathsf{L}(Q')$ and $S'' \in \mathsf{L}(P'') \subseteq \mathsf{LAB}(\tilde{\mathbb{P}}_1)$ such that $S' \xrightarrow{1} R' \xrightarrow{w} S''$. Moreover, since $S'' \in \mathsf{LAB}(\tilde{\mathbb{P}}_1)$ implies $S'' \xrightarrow{1}$, by Proposition B.5-2, we can conclude that $S' \xrightarrow{w}_{\mathsf{LE}(S')} S''$.

Now, $S \xrightarrow{v} S'$ and $S' \xrightarrow{w}_{\mathsf{LE}(S')} S''$ imply $S \xrightarrow{vw}_{\mathsf{LE}(S)} S''$ (this is a trivial consequence of the definition of B-steps) and, by Proposition B.5-1, $S \xrightarrow{1} R \xrightarrow{vw}_{\mathsf{LE}(S)} S''$, Finally, by Proposition A.3-(i) implies that $P \xrightarrow{1} Q \xrightarrow{vw}_{\mathsf{LE}(S)} P''$ where $P = \mathsf{R}(S)$, $Q = \mathsf{R}(R)$ and $P'' = \mathsf{R}(S'')$ □

The next lemma is a trivial consequence of the previous one.

Lemma H.2 Let $S, S', S'' \in \tilde{\mathbb{S}}_1$, $Q' \in \tilde{\mathbb{S}}$ with S read-proper, $v, w \in \mathbb{A}_\tau$ such that $S \xrightarrow{v} S' \xrightarrow{1} Q' \xrightarrow{w} S''$. Then there exists $Q \in \tilde{\mathbb{S}}$ such that $S \xrightarrow{1} Q \xrightarrow{vw} S''$.

Proof: Assume that $S \xrightarrow{v} S' \xrightarrow{1} Q' \xrightarrow{w} S''$. Then, by Theorem 4.3-1, we also have $[\![S]\!] \xrightarrow{v} [\![S']\!] \xrightarrow{1} [\![Q']\!] \xrightarrow{w} [\![S'']\!]$. Now, by Lemma H.1, there is $Q'' \in \tilde{\mathbb{P}}$ such that $[\![S]\!] \xrightarrow{1} Q'' \xrightarrow{vw} [\![S'']\!]$. By Theorem 4.3-2, we have finally that there is $Q \in \tilde{\mathbb{S}}$ with $[\![Q]\!] = Q''$ such that $S \xrightarrow{1} Q \xrightarrow{vw} S''$. □

Now we can prove Theorem 8.2:

Theorem 8.2 *Dekker* is live iff *Dekker$_{io}$* does not have catastrophic cycles.

Proof: Assume to the contrary that *Dekker* $\in \mathbb{S}_1$ and, hence, $[\![Dekker]\!] \in \mathbb{P}_1$ are not live. According to Theorems 6.8 and 6.9, there is a timed execution sequence with infinitely many steps along which a process (by symmetry, we can assume it is $\mathtt{P}_2$) requests to enter its critical section (it performs $\mathtt{req}_2$), but P_2 never performs cs_2 afterwards. In the following, we will prove that this timed execution corresponds to a catastrophic cycle in the rRTS of $[\![Dekker_{io}]\!]$. Since – by Theorem 4.3 – we have $\mathsf{RTS}(Dekker_{io}) = \mathsf{RTS}([\![Dekker_{io}]\!])$ as well as $\mathsf{rRTS}(Dekker_{io}) = \mathsf{rRTS}([\![Dekker_{io}]\!])$, the above result allows us to conclude the proof.

Let $\gamma = [\![Dekker]\!] \xrightarrow{1} Q \xrightarrow{w_0} R_0 \xrightarrow{1} Q_0 \xrightarrow{w_1} R_1 \xrightarrow{1} Q_1 \xrightarrow{w_2} R_2 \xrightarrow{1} \ldots$ be a computation as above; we call it a *witness*. Assume that the last $\mathtt{req}_2$ is an action of w_i (for some $i \geq 0$) and, hence, that the action cs_2 never occurs in w_j, for any $j > i$.

Let $w = w_0 w_1 \ldots w_i$. Since $P \xrightarrow{1} Q$ implies $P = \mathsf{unmark}(Q)$ for a generic $P \in \tilde{\mathbb{P}}_1$, repeated application of Proposition 6.4-2 (at first to $[\![Dekker]\!]$ and Q) shows that $[\![Dekker]\!]$ can perform w and that the process reached is $\mathsf{unmark}(R_i) = R_i$ (this follows since $R_i \in \tilde{\mathbb{P}}_1$ by Propositions 2.10 and 5.11); thus $\gamma' = [\![Dekker]\!] \xrightarrow{w} R_i \xrightarrow{1} Q_i \xrightarrow{w_{i+1}} R_{i+1} \xrightarrow{1} Q_{i+1} \xrightarrow{w_{i+2}} R_{i+2} \xrightarrow{1} \ldots$ is also a timed execution sequence where cs_2 is pending indefinitely.

Now, assume that one or more occurrences of the τ in conflict with $\mathtt{req}_2$ are performed before the last $\mathtt{req}_2$. Since all processes in this part of γ' are initial, such τ's do not change the process; moreover, after the last $\mathtt{req}_2$, there are no such τ's anyway. Thus, we can remove all such internal steps and obtain a timed execution sequence $\gamma'' = [\![Dekker]\!] \xrightarrow{w'} R_i \xrightarrow{1} Q_i \xrightarrow{w_{i+1}} R_{i+1} \xrightarrow{1} Q_{i+1} \xrightarrow{w_{i+2}} R_{i+2} \xrightarrow{1} \ldots$ where again cs_2 is pending indefinitely (w' is w where the τ's in conflict with $\mathtt{req}_2$ have been removed).

Starting from R_i, it may either happen that both processes $\mathtt{P}_1$ and $\mathtt{P}_2$ get stuck or only process $\mathtt{P}_2$ gets stuck while $\mathtt{P}_1$ repeatedly enters and exits its critical section. In both cases, there is a set of states that are repeatedly entered along the computation γ''. Without loss of generality, we may assume that there exists j, k with $i \leq j \leq k$ such that $R_j = R_{k+1}$. Hence, $\gamma'' = [\![\mathit{Dekker}]\!] \xrightarrow{w'} R_i \xrightarrow{1} Q_i \xrightarrow{w_{i+1}} R_{i+1} \xrightarrow{1} \dots \xrightarrow{w_j} R_j \xrightarrow{1} Q_j \xrightarrow{w_{j+1}} R_{j+1} \xrightarrow{1} \dots \xrightarrow{w_k} R_k \xrightarrow{1} Q_k \xrightarrow{w_{k+1}} R_j \xrightarrow{1} \dots$

Now, we change all $\mathtt{req}_1$ and $\mathtt{cs}_1$ occurring in γ'' into τ; we also change all $\mathtt{req}_2$ and $\mathtt{cs}_2$ into in and out, respectively. This gives a timed execution sequence of $[\![\mathit{Dekker}_{io}]\!]$ since the hidden and renamed actions are not synchronised or relabelled. In our case, we obtain a computation of the form $[\![\mathit{Dekker}_{io}]\!] \xrightarrow{v''} R'_i \xrightarrow{1} Q'_i \xrightarrow{v_{i+1}} R'_{i+1} \xrightarrow{1} \dots \xrightarrow{v_j} R'_j \xrightarrow{1} Q'_j \xrightarrow{v_{j+1}} R'_{j+1} \xrightarrow{1} \dots \xrightarrow{v_k} R'_k \xrightarrow{1} Q'_k \xrightarrow{v_{k+1}} R'_j \xrightarrow{1} \dots$ with $v_{i+1}, \dots, v_j, v_{j+1}, \dots, v_k, v_{k+1} \in \{\tau\}^*$ (recall that $\mathtt{cs}_2$ (out) and, hence, also $\mathtt{req}_2$ (in) never occur in w_j (v_j, resp.) for each $j \geq i$). This computation demonstrates that $[\![\mathit{Dekker}_{io}]\!]$ has a catastrophic cycle.

Now, we prove the reverse implication. If a process reachable from Dekker_{io} has a pending out (corresponding to $\mathtt{cs}_2$), then it cannot perform in (corresponding to $\mathtt{req}_2$) as we can read off from the structure of P_2; thus, if such a process can refuse out, it can also refuse $\{\mathit{in}, \mathit{out}\}$. Thus, in the reduced refusal transition system of Dekker_{io}, all time steps are full. Hence, if there exists a catastrophic cycle, this can be translated back to a computation of Dekker with infinitely many time steps (all full) where either (i) $\mathtt{P}_2$ is stuck after performing $\mathtt{cs}_2$ in $\{\mathit{kw1}.b_2\mathit{wf}.P_2, \underline{\mathit{kw1}}.b_2\mathit{wf}.P_2\}$ or in $\{b_2\mathit{wf}.P_2, \underline{b_2\mathit{wf}}.P_2\}$ or (ii) there is a $\mathtt{req}_2$ not followed by $\mathtt{cs}_2$. In case (ii), we obtain a computation that is almost a witness ("almost" because it might not start with a time step); but with Lemma H.2, we can transform the start $S \xrightarrow{v} S' \xrightarrow{1} Q' \xrightarrow{w} S''$ to $S \xrightarrow{1} Q \xrightarrow{vw} S''$ and are done.

We now prove that case (i) is not possible. First, assume that $\mathtt{P}_2$ is stuck in $\{b_2\mathit{wf}.P_2, \underline{b_2\mathit{wf}}.P_2\}$, and consider a state where $\mathtt{P}_2$ enters $\{b_2\mathit{wf}.P_2, \underline{b_2\mathit{wf}}.P_2\}$ for the last time. After the next time step, it is in state $\underline{b_2\mathit{wf}}.P_2$, i.e. Dekker_{io} is in a state $((Q_1 \parallel \underline{b_2\mathit{wf}}.P_2) \parallel_B (\mathsf{B}'_1(x) \parallel \mathsf{B}'_2(\mathit{true}) \parallel \mathsf{K}'(y)))[\Phi']$ where Φ' is the suitable (hiding and) renaming and Q_1, $\mathsf{B}'_1(x)$, $\mathsf{B}'_2(\mathit{true})$ and $\mathsf{K}'(1)$ are the states of the other component processes. In particular, we have either $\mathsf{B}'_2(\mathit{true}) = \{\underline{b_2\mathit{rt}}, b_2\mathit{wt}\} \triangleright \underline{b_2\mathit{wf}}.\mathsf{B}_2(\mathit{false})$ or $\mathsf{B}'_2(\mathit{true}) = \{b_2\mathit{rt}, b_2\mathit{wt}\} \triangleright \underline{b_2\mathit{wf}}.\mathsf{B}_2(\mathit{false})$ (depending on the enabledness and hence urgency of $b_2\mathit{rt}$ in Q_1). Now, since the process $\mathtt{P}_1$ can read but not write the variable b_2, the internal action corresponding to the writing of the value false on b_2 is always enabled and urgent in all the states we can reach from Q via a sequence of internal actions due to moves of $\mathtt{P}_1$; thus, no further time step is possible, a contradiction.

Second, we assume that $\mathtt{P}_2$ is stuck in $\{\mathit{kw1}.b_2\mathit{wf}.P_2, \underline{\mathit{kw1}}.b_2\mathit{wf}.P_2\}$; analogously, there is a state $Q = ((Q_1 \parallel \underline{\mathit{kw1}}.b_2\mathit{wf}.P_2) \parallel_B ((\mathsf{B}'_1(x) \parallel \mathsf{B}'_2(\mathit{true}) \parallel \mathsf{K}'(y)))[\Phi']$ on the computation such that $\mathtt{P}_2$ never performs an action again. We distinguish two further subcases:

$y = 2$ We reach a state like Q where $y = 2$. Then, either $\mathsf{K}'(y) = \{\mathit{kr2}, \mathit{kw2}\} \triangleright \underline{\mathit{kw1}}.\mathsf{K}(1)$ or $\mathsf{K}'(y) = \{\underline{\mathit{kr2}}, \mathit{kw2}\} \triangleright \underline{\mathit{kw1}}.\mathsf{K}(1)$ or $\mathsf{K}'(y) = \{\mathit{kr2}, \underline{\mathit{kw2}}\} \triangleright \underline{\mathit{kw1}}.\mathsf{K}(1)$. This case is similar to the previous one, since the process $\mathtt{P}_1$ never writes value 1 into the variable k; thus, $\mathit{kw1}$ remains urgent, and we again derive a contradiction.

$y = 1$ We do not reach a suitable state with $y = 2$. Then, either $\mathsf{K}'(y) = \{\underline{\mathit{kr1}}, \underline{\mathit{kw1}}\} \triangleright \mathit{kw2}.\mathsf{K}(2)$ (if the action $\mathit{kr1}$ is enabled in Q_1) or $\mathsf{K}'((1) = \{\mathit{kr1}, \underline{\mathit{kw1}}\} \triangleright \mathit{kw2}.\mathsf{K}(2)$; it is not possible that Q_1 enables $\mathit{kw2}$, since then $Q_1 \in \{\mathit{kw2}.b_1\mathit{wf}.P_1, \underline{\mathit{kw2}}.b_1\mathit{wf}.P_1\}$ and we would get to a state like Q with $y = 2$ before the next time step. Thus, again $\mathit{kw1}$ remains urgent, giving a contradiction.

□