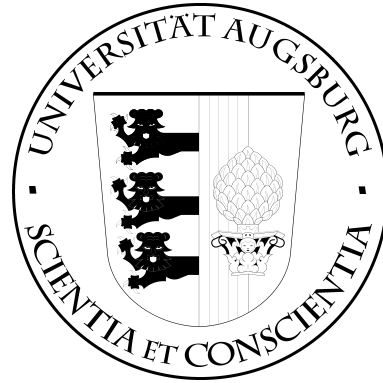


UNIVERSITÄT AUGSBURG

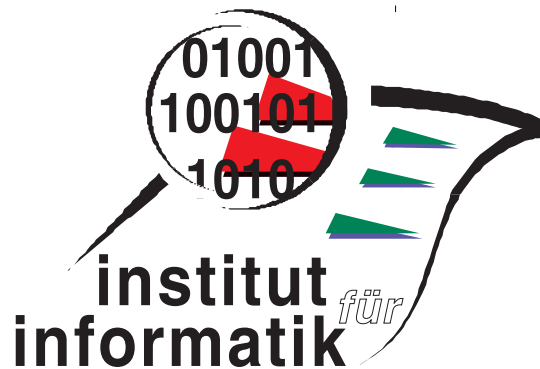


Liveness of a Mutex Algorithm in a Fair
Process Algebra

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

Report 2008-03

February 2008



INSTITUT FÜR INFORMATIK

D-86135 AUGSBURG

Copyright © F. Corradini, M.R. Di Berardini, W. Vogler
Institut für Informatik
Universität Augsburg
D-86135 Augsburg, Germany
<http://www.Informatik.Uni-Augsburg.DE>
— all rights reserved —

Liveness of a Mutex Algorithm in a Fair Process Algebra

F. Corradini, M.R. Di Berardini
Dipartimento di Matematica e Informatica
Università di Camerino

W. Vogler
Institut für Informatik
Universität Augsburg

{flavio.corradini,mariarita.diberardini}@unicam.it vogler@informatik.uni-augsburg.de

Abstract

In earlier work, we have shown that two variants of weak fairness can be expressed comparatively easily in the timed process algebra PAFAS. To demonstrate the usefulness of these results, we complement work by Walker [17] and study the liveness property of Dekker’s mutual exclusion algorithm within our process algebraic setting. We also present some results that allow to reduce the state space of the PAFAS process representing Dekker’s algorithm, and give some insight into the representation of fair behaviour in PAFAS.

1 Introduction

This paper was inspired by the work of Walker [17] who aimed at automatically verifying six mutual exclusion algorithms – including Dekker’s. Walker translated the algorithms into the process algebra CCS [14] and then verified with the Concurrency Workbench [2] that all of them satisfy the *safety property* that the two competing processes are never in their critical sections at the same time.

The *liveness property* that a requesting process will always eventually enter the critical section is more difficult to verify, since one has to assume some fairness, which is not so easy to do in a process algebraic setting; with respect to the verification of liveness, Walker was less successful.

Costa and Stirling [7, 8] have studied some notions of fairness in a process algebra. While their formalisation captures the intuition of fairness faithfully, it is technically involved and leads to processes with infinite state spaces – at least for processes that have an infinite computation, i.e. in all interesting cases. In [4, 5], we have defined fair runs in the spirit of Costa and Stirling and characterised them in the timed process algebra PAFAS as those runs that take infinitely long; here, processes that are finite state in a standard process algebra without time still have a finite transition system in the setting where fairness can be studied. The present paper complements the work by Walker, taking the liveness of Dekker’s algorithm as a case study to demonstrate how our approach to fairness can be used.

Attempting the verification of the liveness property, Walker used the following version in [17] – which could be expressed as a modal mu-calculus formula and checked with the Concurrency Workbench:

Whenever at some point in a run the process P_i requests the execution of its critical section, then in any continuation of that run from that point in which between them the processes execute an infinite number of critical sections, P_i performs its critical section at least once.

The fairness (or progress) assumed here is that infinitely often a critical section is entered. This assumption allows a run where one process enters its critical section repeatedly, while the other requests the execution of its critical section, but then – for no good reason at all – refuses to take the necessary steps to actually enter it. Thus, it is maybe not so surprising that four of the six mutual exclusion algorithms – including Dekker’s – fail to satisfy this property.

Walker then discusses how fairness could be assumed to enable a proof of liveness, but the ideas discussed could not be expressed for use of the Concurrency Workbench. Here, we will model Dekker’s algorithm in the CCS-type process algebra PAFAS and study whether all fair runs satisfy the liveness property.

Actually, we consider two versions of PAFAS. The first one is suitable for (weak) *fairness of actions*, i.e. in a fair run each enabled action must be performed or disabled eventually; if this action is a synchronisation, then the action is already disabled if one synchronisation partner synchronises its ‘part’ of the action with some other process. As a consequence, repeated accesses to a variable can block another access, and for this reason some fair runs of Dekker’s algorithm violate liveness; this is not so different from Walker’s result, but we can point to a realistic reason for the failure, namely the blocking of a variable. We provide two fair runs, one in which one process repeatedly enters its critical section while the other is stuck, and one where both processes are stuck.

It is equally realistic to assume that access to a variable cannot be blocked indefinitely. In the second version of PAFAS, we deal with (weak) *fairness of components*, i.e. in a fair run each enabled component must be performed or disabled eventually. Thus, if a process wants to read a binary variable, it will offer two read-actions (one for each value); if none of these is performed, then in every future state one or the other will be enabled, i.e. the process will be enabled indefinitely; fairness now implies that the process actually will read the variable eventually. Assuming fairness of components, we will show that Dekker’s algorithm indeed satisfies the liveness property.

In this proof, we have to take into account all possible derivatives reachable from *Dekker* along fair computations. In particular, we will consider those states where one process has just performed a request to enter a critical section, and show that from those states the respective process does eventually enter the critical section.

Modelling fairness involves a certain blow-up of the state space, so for a proof by hand the number of states we had to deal with was rather large. Consequently, to manage the proof, we had to rely on structural properties of the processes, which may be of interest independently of the main aims of this paper. Previously, we have characterised fair runs as those action sequences that arise from timed computations with infinitely many unit time steps by deleting these time steps. Our first result states that we can restrict attention to the subclass of such timed computations where each time step occurs as soon as possible and still cover all fair runs.

A considerable reduction of states comes from switching some components to “permanently lazy”, i.e. to require fairness only for the other components. In our case study, the “permanently lazy components” correspond to the variables; so this is a very realistic change, since it seems natural that only the processes are active, while a variable never forces to be read or to be written. In general, switching some components to permanently lazy gives an overapproximation for the fair runs, and it is clearly sufficient to prove a desired property for this possibly larger set of runs.

Finally, we take advantage of symmetries in the *Dekker* algorithm. The two processes that compete for the execution of their critical section, indeed, have a symmetric structure so that their derivatives follow a symmetric pattern. Thus, we check liveness of a generic fair-reachable derivative to deduce the same property of the symmetric one.

These observations have allowed a proof by hand. We believe, however, that they are not specific to this work but really add some general knowledge to the theory of PAFAS [6] useful to be embedded within an automatic tool for the verification based on fairness.

The rest of the paper is organised as follows. The next section recalls PAFAS, its functional and temporal operational semantics and the fairness notions we are interested in (actions and components). Section 3 describes Dekker’s mutual exclusion algorithm and its description within the PAFAS process algebra, and it introduces the liveness property we consider. Section 4 shows that fairness of actions is not suitable for liveness while Section 5 proves the main result of this paper, namely, that any computation from *Dekker* that is fair according to fairness of components satisfies liveness.

2 Fairness and PAFAS

In this section we recall PAFAS, its timed behaviour and the fairness notions we consider in the rest of the paper, namely fairness of actions and components. Instead of using the very involved direct formalisations of fairness in the spirit of [7, 8], we define the two types of fair traces on the basis of our characterisations with everlasting timed execution sequences in the two respective versions of PAFAS.

2.1 Fairness of Actions and PAFAS

PAFAS is a CCS-like process description language [14] (with *TCSP*-like parallel composition), where basic actions are atomic and instantaneous but have associated a time bound interpreted as a maximal time delay for their execution. As explained in [6], these upper time bounds (which are either 0 or 1, for simplicity) are suitable for evaluating the performance of asynchronous systems. Moreover, time bounds do not influence functionality (which actions are performed); so compared to CCS, also PAFAS treats the full functionality of asynchronous systems.

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, \dots and those of \mathbb{A}_τ are denoted by α, β, \dots . Actions in \mathbb{A}_τ 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 $\underline{\alpha}, \underline{\beta}, \dots$. 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, \dots .

$\Phi : \mathbb{A}_\tau \rightarrow \mathbb{A}_\tau$ 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 relabelling 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. (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 [6] for more details.) 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 .

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

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

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 guarded (see below).

The set $\tilde{\mathbb{P}}$ of (general) *(timed) process terms* is generated by the following grammar:

$$Q ::= P \mid \underline{\alpha}.P \mid Q + Q \mid Q \parallel_A Q \mid Q[\Phi] \mid \text{rec } x.Q$$

where $P \in \tilde{\mathbb{P}}_1$, $x \in \mathcal{X}$, $\alpha \in \mathbb{A}_\tau$, Φ is a general relabelling function and $A \subseteq \mathbb{A}$ possibly infinite. We assume that recursion is *guarded*, i.e. for $\text{rec } x.Q$ 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 *guarded* if each occurrence of a variable is guarded in this sense. A timed process term Q is *closed*, if every variable x in Q is bound by the corresponding

rec x -operator; such a Q in $\tilde{\mathbb{P}}$ and $\tilde{\mathbb{P}}_1$ is simply called *process* and *initial process* resp., and their sets are denoted by \mathbb{P} and \mathbb{P}_1 resp.¹

A brief description of the (PAFAS) operators now follows. The Nil-process nil 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 $a.b.nil + c.nil$. $Q_1 + Q_2$ models the choice between two conflicting processes Q_1 and Q_2 . $Q_1 \parallel_A Q_2$ 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 . $\text{rec } x.Q$ models a recursive definition.

Initial processes are just standard processes of a standard process algebra. 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.

We can now define the set of activated actions in a process term. Given a process term Q , $\mathcal{A}(Q, A)$ denotes the set of the activated (or enabled) actions of Q when the environment prevents the actions in A .

Definition 2.2 (*activated basic actions*) Let $Q \in \tilde{\mathbb{P}}$ and $A \subseteq \mathbb{A}$. The set $\mathcal{A}(Q, A)$ is defined by induction on Q .

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

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

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

$$\text{Par:} \quad \mathcal{A}(Q_1 \parallel_B Q_2, A) = \mathcal{A}(Q_1, A \cup A') \cup \mathcal{A}(Q_2, A \cup A'') \\ \text{where } A' = (\mathcal{A}(Q_1) \setminus \mathcal{A}(Q_2)) \cap B \text{ and } A'' = (\mathcal{A}(Q_2) \setminus \mathcal{A}(Q_1)) \cap B$$

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

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

The *activated actions* of Q are defined as $\mathcal{A}(Q, \emptyset)$ which we abbreviate to $\mathcal{A}(Q)$.

The set A represents the actions restricted upon. This is the reason why $\mathcal{A}(\alpha.P, A) = \mathcal{A}(\underline{\alpha}.P, A) = \emptyset$ if $\alpha \in A$ and $\mathcal{A}(\alpha.P, A) = \mathcal{A}(\underline{\alpha}.P, A) = \{\alpha\}$, if $\alpha \notin A$. A nondeterministic process can perform all the actions that its alternative components can perform minus the restricted ones. Parallel composition increases the prevented set: $\mathcal{A}(Q_1 \parallel_B Q_2, A)$ includes the actions that Q_1 can perform when we prevent all actions in A plus all actions in B that Q_2 cannot perform, and it includes the analogous actions of Q_2 . The other rules are as expected.

A significant subset of the activated actions is the set of urgent ones. These are activated actions that cannot let time pass.

Definition 2.3 (*urgent activated action*) Let $Q \in \tilde{\mathbb{P}}$ and $A \subseteq \mathbb{A}$. The set $\mathcal{U}(Q, A)$ is defined as in Definition 2.2 when $\mathcal{A}(_)$ is replaced by $\mathcal{U}(_)$ and the Pref-rule is replaced by the following one:

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

¹In [6], we prove that \mathbb{P}_1 processes do not have time-stops; i.e. every finite process run can be extended such that time grows unboundedly. This result was proven for a different operational semantics than that defined in this paper but a similar proof applies also in the current setting.

The *urgent activated actions* of Q are defined as $\mathcal{U}(Q, \emptyset)$ which we abbreviate to $\mathcal{U}(Q)$.

The operational semantics exploits two functions on process terms: $\text{clean}(_)$ and $\text{unmark}(_)$. Function $\text{clean}(_)$ 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 $\text{clean}(Q, A)$ denotes the set of actions that are not enabled or urgent due to restrictions of the environment. Function $\text{unmark}(_)$ simply removes all urgencies (inactive or not) in a process term $Q \in \tilde{\mathbb{P}}$. Both functions can be defined, as follows, by induction on the process structure.

Definition 2.4 (*cleaning inactive urgencies*) Given a process term $Q \in \tilde{\mathbb{P}}$ we define $\text{clean}(Q)$ as $\text{clean}(Q, \emptyset)$ where, for a set $A \subseteq \mathbb{A}$, $\text{clean}(Q, A)$ is defined as follows:

$$\begin{array}{ll}
\text{Nil, Var:} & \text{clean}(\text{nil}, A) = \text{nil}, \quad \text{clean}(x, A) = x \\
\\
\text{Pref:} & \text{clean}(\alpha.P, A) = \alpha.P \quad \text{clean}(\underline{\alpha}.P, A) = \begin{cases} \alpha.P & \text{if } \alpha \in A \\ \underline{\alpha}.P & \text{otherwise} \end{cases} \\
\\
\text{Sum:} & \text{clean}(Q_1 + Q_2, A) = \text{clean}(Q_1, A) + \text{clean}(Q_2, A) \\
\\
\text{Par:} & \text{clean}(Q_1 \parallel_B Q_2, A) = \text{clean}(Q_1, A \cup A') \parallel_B \text{clean}(Q_2, A \cup A'') \\
& \text{where } A' = (\mathcal{U}(Q_1) \setminus \mathcal{U}(Q_2)) \cap B \text{ and } A'' = (\mathcal{U}(Q_2) \setminus \mathcal{U}(Q_1)) \cap B \\
\\
\text{Rel:} & \text{clean}(Q[\Phi], A) = \text{clean}(Q, \Phi^{-1}(A))[\Phi] \\
\\
\text{Rec:} & \text{clean}(\text{rec } x.Q, A) = \text{rec } x. \text{clean}(Q, A)
\end{array}$$

Definition 2.5 (*cleaning all urgencies*) Let Q be a $\tilde{\mathbb{P}}$ term. Then $\text{unmark}(Q)$ is the term obtained by replacing each $\underline{\alpha}$ by α .

2.1.1 The functional behaviour of PAFAS process

The transitional semantics describing the functional behaviour of PAFAS processes indicates which basic actions they can perform. Timing can be disregarded: when an action is performed, one cannot see whether it was urgent or not, i.e. $\underline{\alpha}.P \xrightarrow{\alpha} P$; on the other hand, component $\alpha.P$ has to act *within* time 1, i.e. it can also act immediately, giving $\alpha.P \xrightarrow{\alpha} P$.

Definition 2.6 (*Functional operational semantics*) The following SOS-rules define the transition relations $\xrightarrow{\alpha} \subseteq (\tilde{\mathbb{P}} \times \tilde{\mathbb{P}})$ for $\alpha \in \mathbb{A}_\tau$, the *action transitions*.

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.

$$\begin{array}{c}
\text{PREF}_{a1} \frac{}{\alpha.P \xrightarrow{\alpha} P} \quad \text{PREF}_{a2} \frac{}{\underline{\alpha}.P \xrightarrow{\alpha} P} \quad \text{SUM}_a \frac{Q_1 \xrightarrow{\alpha} Q'}{Q_1 + Q_2 \xrightarrow{\alpha} Q'} \\
\\
\text{PAR}_{a1} \frac{\alpha \notin A, Q_1 \xrightarrow{\alpha} Q'_1}{Q_1 \parallel_A Q_2 \xrightarrow{\alpha} \text{clean}(Q'_1 \parallel_A Q_2)} \quad \text{PAR}_{a2} \frac{\alpha \in A, Q_1 \xrightarrow{\alpha} Q'_1, Q_2 \xrightarrow{\alpha} Q'_2}{Q_1 \parallel_A Q_2 \xrightarrow{\alpha} \text{clean}(Q'_1 \parallel_A Q'_2)} \\
\\
\text{REL}_a \frac{Q \xrightarrow{\alpha} Q'}{Q[\Phi] \xrightarrow{\Phi(\alpha)} Q'[\Phi]} \quad \text{REC}_a \frac{Q\{\text{rec } x.\text{unmark}(Q)/x\} \xrightarrow{\alpha} Q'}{\text{rec } x.Q \xrightarrow{\alpha} Q'}
\end{array}$$

Additionally, there are symmetric rules for PAR_{a1} and SUM_a for actions of Q_2 .

For an *initial* process P_0 , we say that a finite or infinite sequence $\alpha_0\alpha_1\dots$ of actions from \mathbb{A}_τ is a *trace* of P_0 , if there is a sequence $P_0 \xrightarrow{\alpha_0} P_1 \xrightarrow{\alpha_1} \dots$ of action transitions, possibly ending with a process P_n .

When in the rules for parallel composition one component changes, this changes the context for the other component such that some urgent actions might be no longer urgent or get disabled. For the necessary changes to the marking, `clean` is called upon as announced above. E.g. in $(\underline{a}.\text{nil} \parallel_{\emptyset} \underline{b}.\text{nil}) \parallel_{\{a,b\}} (\underline{a}.\text{nil} + \underline{b}.\text{nil} + \underline{c}.a.b.\text{nil}) \xrightarrow{c} (a.\text{nil} \parallel_{\emptyset} b.\text{nil}) \parallel_{\{a,b\}} (a.b.\text{nil})$, both actions a and b lose their urgency, since the third component only offers a new non-urgent synchronisation on a , and b is no longer enabled.

The use of `unmark` in rule REC_a has to be contrasted with the temporal behaviour defined next. Consider an initial process P ; after a time-step, the recursive term $\text{rec } x.P$ evolves to $\text{rec } x.\text{urgent}(P)$ (see Definition 2.7 below). Since occurrences of x in P are guarded, each x stands for a process which is not enabled yet and cannot have urgent actions; thus, these recursive calls in $\text{rec } x.\text{urgent}(P)$ refer to P and not to $\text{urgent}(P)$, which explains the substitution in rule REC_a of Definition 2.6, which in turn shows the use of `unmark`; cf. the example at the end of Subsection 2.1.2.

2.1.2 The temporal behaviour of PAFAS process

Now, we consider transitions corresponding to the passage of one unit of time. The function `urgent` marks all *enabled* actions of a process as urgent when a time step is performed. Before the next time step, all such actions must occur or get disabled.

Definition 2.7 (*time step, timed execution sequences*) For $P \in \tilde{\mathbb{P}}_1$, we write $P \xrightarrow{1} Q$ when $Q = \text{urgent}(P)$, where $\text{urgent}(P)$ (often also written \underline{P}) abbreviates $\text{urgent}(P, \emptyset)$ and $\text{urgent}(P, A)$ is defined as follows:

Nil, Var: $\text{urgent}(\text{nil}, A) = \text{nil}, \quad \text{urgent}(x, A) = x$

Pref: $\text{urgent}(\alpha.P, A) = \begin{cases} \underline{\alpha}.P & \text{if } \alpha \notin A \\ \alpha.P & \text{otherwise} \end{cases}$

Sum: $\text{urgent}(P_1 + P_2, A) = \text{urgent}(P_1, A) + \text{urgent}(P_2, A)$

Par: $\text{urgent}(P_1 \parallel_B P_2, A) = \text{urgent}(P_1, A \cup A') \parallel_B \text{urgent}(P_2, A \cup A'')$
where $A' = (\mathcal{A}(P_1) \setminus \mathcal{A}(P_2)) \cap B$ and $A'' = (\mathcal{A}(P_2) \setminus \mathcal{A}(P_1)) \cap B$

Rel: $\text{urgent}(P[\Phi], A) = \text{urgent}(P, \Phi^{-1}(A))[\Phi]$

Rec: $\text{urgent}(\text{rec } x.P, A) = \text{rec } x.\text{urgent}(P, A)$

For an initial process P_0 , we say that a sequence of transitions $\gamma = P_0 \xrightarrow{1} Q_0 \xrightarrow{\lambda_0} Q_1 \xrightarrow{\lambda_1} \dots$ with $\lambda_i \in \mathbb{A}_\tau \cup \{1\}$ is a *timed execution sequence* if it is an infinite sequence of action transitions and time steps (starting with a time step)². 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.

²Note that a maximal sequence of such transitions/steps is never finite, since for $\gamma = Q_0 \xrightarrow{\lambda_0} Q_1 \xrightarrow{\lambda_1} \dots \xrightarrow{\lambda_{n-1}} Q_n$, we have $Q_n \xrightarrow{\alpha}$ or $Q_n \xrightarrow{1}$ (see Proposition 3.13 in [5]).

Example 2.8 As an example for the definitions given so far, consider process $P = (R \parallel_{\emptyset} W) \parallel_{\{r,w\}} V$, where $V = \text{rec } x.(r.x + w.x)$ models a process variable, $R = \text{rec } x.r.x$ and $W = \text{rec } x.w.x$ model the activities of reading and writing the process variable, respectively. According to our operational semantics, we have $V \xrightarrow{1} \underline{V} = \text{rec } x.(\underline{r}.x + \underline{w}.x) \xrightarrow{r} V$; observe that, intuitively speaking, each occurrence of x is replaced by $V = \text{rec } x.\text{unmark}(\underline{r}.x + \underline{w}.x)$ before the second transition. Furthermore, we have:

$$P \xrightarrow{1} (\underline{R} \parallel_{\emptyset} \underline{W}) \parallel_{\{r,w\}} \underline{V} = ((\text{rec } x. \underline{r}.x) \parallel_{\emptyset} (\text{rec } x. \underline{w}.x)) \parallel_{\{r,w\}} \text{rec } x. (\underline{r}.x + \underline{w}.x) \xrightarrow{r} P$$

To explain the first transition, we notice that initial process P can synchronise either on action r or on action w . Both actions are activated so that they become urgent after one time unit. 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 loses its urgency since its synchronisation partner V offers a new, non-urgent synchronisation. At this stage, it is function `clean` in Definition 2.6 that operates this change.

2.1.3 Fairness of actions and timing

We can now define the (weakly) fair traces in terms of everlasting timed execution sequences: namely, a trace is fair (w.r.t. fairness of actions) if there exists a corresponding timed execution sequence with infinitely many time step (see full version of [5] for more details).

Definition 2.9 (*fair traces*) Let $P_0 \in \mathbb{P}_1$ and $\alpha_0, \alpha_1, \alpha_2, \dots \in \mathbb{A}_\tau$. A trace $\alpha_0 \alpha_1 \alpha_2 \dots$ of P_0 is *fair* (w.r.t. fairness of actions) if it can be obtained as the sequence of actions in a non-Zeno timed execution sequence. More in detail:

1. A finite trace $\alpha_0 \alpha_1 \alpha_2 \dots \alpha_{n-1}$ is fair if and only if there exists a timed execution sequence

$$P_{i_0} \xrightarrow{1} Q_{i_0} \xrightarrow{v_0} P_{i_1} \xrightarrow{1} Q_{i_1} \xrightarrow{v_1} P_{i_2} \dots P_{i_{m-1}} \xrightarrow{1} Q_{i_{m-1}} \xrightarrow{v_{m-1}} P_{i_m} \xrightarrow{1} Q_{i_m} \xrightarrow{1} Q_{i_m} \dots$$

where $P_{i_0} = P_0$ and $v_0 v_1 \dots v_{m-1} = \alpha_0 \alpha_1 \dots \alpha_{n-1}$;

2. an infinite trace $\alpha_0 \alpha_1 \alpha_2 \dots$ is fair if and only if there exists a timed execution sequence

$$P_{i_0} \xrightarrow{1} Q_{i_0} \xrightarrow{v_0} P_{i_1} \xrightarrow{1} Q_{i_1} \xrightarrow{v_1} P_{i_2} \dots P_{i_m} \xrightarrow{1} Q_{i_m} \xrightarrow{v_m} P_{i_{m+1}} \dots$$

where $P_{i_0} = P_0$ and $v_0 v_1 \dots v_m \dots = \alpha_0 \alpha_1 \dots \alpha_i \dots$

Recall that this is a characterisation for fair traces obtained in [5] on the basis of a more intuitive, but very complex definition of fair traces in the spirit of [7, 8].

In the following example we use the same process as in Example 2.8 to provide an intuitive idea on how (weak) fairness of actions works and on its relationship with timing.

Example 2.10 Consider again $P = (R \parallel_{\emptyset} W) \parallel_{\{r,w\}} V$ and a run by P consisting of infinitely many r 's. Such a computation 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 will be possible; this happens when r is performed, as we noted above:

$$P \xrightarrow{1} ((\text{rec } x. \underline{r}.x) \parallel_{\emptyset} (\text{rec } x. \underline{w}.x)) \parallel_{\{r,w\}} \text{rec } x. (\underline{r}.x + \underline{w}.x) \xrightarrow{r} P$$

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 can indefinitely block another process trying to write to it (and vice versa for repeated writing). This kind of computations will be prevented by fairness of components as we will discuss in Example 2.17.

2.2 Fairness of Components and PAFAS^C

In this section, we concentrate on weak *fairness of components*. Not surprisingly, the PAFAS timed operational semantics does not support a characterisation of fair behaviour w.r.t. components. But we have found a suitable variation of PAFAS and its semantics which allows us to characterise Costa and Stirling's fairness of components again in terms of a simple filtering of system executions. Conceptually, we proceed analogously to Section 2.1, but a number of technical changes are needed to define the new semantics.

The new operational semantics of processes we have arrived at can again be understood as the behaviour of timed processes with upper time bounds. We assume that for each parallel component this upper time bound is 1; hence, a component will perform some action within time 1 provided it is continually enabled. In other words, when time 1 passes, an enabled component becomes urgent and, before the next time step, it must perform an action (or get disabled).

Also in this case, we assume that time elapses in a discrete way. Thus, an action-prefixed process $a.P$ can again either do action a and become process P (as usual in CCS) or can let one unit of time pass and become $\underline{a}.P$; $\underline{a}.P$ cannot let time pass, but can only do a to become P . Since we associate time bounds with components in the present section, we also mark the other dynamic operator $+$ as urgent: a process $P + Q$ becomes $P \underline{+} Q$ after a time step. This variant of PAFAS is called PAFAS^C henceforth.

Definition 2.11 (*timed process terms*) Let $\tilde{\mathbb{P}}_1$ be the set of *initial timed process terms* as given in Definition 2.1. The set $\tilde{\mathbb{P}}_c$ of (*component-oriented*) *timed process terms* is generated by the following grammar:

$$Q ::= P \mid \underline{\alpha}.P \mid P \underline{+} P \mid Q \parallel_A Q \mid Q[\Phi] \mid \text{rec } x.Q$$

where $P \in \tilde{\mathbb{P}}_1$, $x \in \mathcal{X}$, $\alpha \in \mathbb{A}_\tau$, Φ is a general relabelling function, and $A \subseteq \mathbb{A}$ possibly infinite. Again, we assume that recursion is always guarded. The set of closed timed process terms in $\tilde{\mathbb{P}}_c$, simply called *processes* is denoted by \mathbb{P}_c .

For studying fairness, we are interested in the initial processes, and these coincide in PAFAS and in PAFAS^C; they are actually common CCS/TCSP-like processes. The additional terms of $\tilde{\mathbb{P}}_c$ turn up in evolutions of terms from $\tilde{\mathbb{P}}_1$ involving time steps, and here PAFAS and PAFAS^C differ.

We define function $\mathcal{A}(_)$ on process terms, that returns the *activated* (or enabled) actions of a process term. Given a process Q , $\mathcal{A}(Q)$ again abbreviates $\mathcal{A}(Q, \emptyset)$ and $\mathcal{A}(Q, A)$ denotes the set of actions that process Q can perform when the environment prevents the actions in $A \subseteq \mathbb{A}$.

Definition 2.12 (*activated basic actions*) Let $Q \in \tilde{\mathbb{P}}_c$ and $A \subseteq \mathbb{A}$. The set $\mathcal{A}(Q, A)$ can be defined as in Definition 2.2 when rule Sum is replaced as follows:

$$\text{Sum:} \quad \mathcal{A}(P_1 + P_2, A) = \mathcal{A}(P_1 \underline{+} P_2, A) = \mathcal{A}(P_1, A) \cup \mathcal{A}(P_2, A)$$

2.2.1 The operational behaviour of PAFAS^C processes

The transitional semantics describing the functional behaviour of PAFAS^C processes indicates which basic actions they can perform. The operational semantics exploits two functions on process terms: $\text{clean}(_)$ and $\text{unmark}(_)$ with a similar meaning to those defined in the corresponding Section 2.1.

Function $\text{clean}(_)$ removes all inactive urgencies in a process term $Q \in \tilde{\mathbb{P}}_c$. Indeed, when a process evolves, only whole components (and not single actions) may lose their urgency when their actions are no longer enabled due to changes of the context; the corresponding change of markings is performed by clean , where again set A in $\text{clean}(Q, A)$ denotes the set of actions that are not enabled due to restrictions of the environment. Function $\text{unmark}(_)$ simply removes all urgencies (inactive or not) in a process term $Q \in \tilde{\mathbb{P}}_c$.

Definition 2.13 (*cleaning inactive urgencies*) Given a process term $Q \in \tilde{\mathbb{P}}_c$ we define $\text{clean}(Q)$ as $\text{clean}(Q, \emptyset)$ where, for a set $A \subseteq \mathbb{A}$, $\text{clean}(Q, A)$ is defined as in Definition 2.4 where rules Sum and Par are replaced as follows:

$$\text{Sum: } \text{clean}(P_1 + P_2, A) = P_1 + P_2 \quad \text{clean}(P_1 \pm P_2, A) = \begin{cases} P_1 + P_2 & \text{if } \mathcal{A}(P_1) \cup \mathcal{A}(P_2) \subseteq A \\ P_1 \pm P_2 & \text{otherwise} \end{cases}$$

$$\text{Par: } \text{clean}(Q_1 \parallel_B Q_2, A) = \text{clean}(Q_1, A \cup A') \parallel_B \text{clean}(Q_2, A \cup A'')$$

where $A' = (\mathcal{A}(Q_1) \setminus \mathcal{A}(Q_2)) \cap B$ and $A'' = (\mathcal{A}(Q_2) \setminus \mathcal{A}(Q_1)) \cap B$

Definition 2.14 (*functional operational semantics*) The following SOS-rules define the transition relations $\vdash^{\alpha} \subseteq (\tilde{\mathbb{P}}_c \times \tilde{\mathbb{P}}_c)$ for $\alpha \in \mathbb{A}_\tau$, the *action transitions*; observe the use of \vdash instead of \rightarrow .

$$\begin{array}{c} \text{PREF}_{a1} \frac{}{\alpha.P \vdash^{\alpha} P} \quad \text{PREF}_{a2} \frac{}{\alpha.P \vdash^{\alpha} P} \\ \text{SUM}_{a1} \frac{P_1 \vdash^{\alpha} P'_1}{P_1 + P_2 \vdash^{\alpha} P'_1} \quad \text{SUM}_{a2} \frac{P_1 \vdash^{\alpha} P'_1}{P_1 \pm P_2 \vdash^{\alpha} P'_1} \\ \text{PAR}_{a1} \frac{\alpha \notin A, Q_1 \vdash^{\alpha} Q'_1}{Q_1 \parallel_A Q_2 \vdash^{\alpha} \text{clean}(Q'_1 \parallel_A Q_2)} \quad \text{PAR}_{a2} \frac{\alpha \in A, Q_1 \vdash^{\alpha} Q'_1, Q_2 \vdash^{\alpha} Q'_2}{Q_1 \parallel_A Q_2 \vdash^{\alpha} \text{clean}(Q'_1 \parallel_A Q'_2)} \\ \text{REL}_a \frac{Q \vdash^{\alpha} Q'}{Q[\Phi] \xrightarrow{\Phi(\alpha)} Q'[\Phi]} \quad \text{REC}_a \frac{Q\{\text{rec } x. \text{unmark}(Q)/x\} \vdash^{\alpha} Q'}{\text{rec } x. Q \vdash^{\alpha} Q'} \end{array}$$

Additionally, there are symmetric rules for PAR_{a1} , SUM_{a1} and SUM_{a2} for actions of P_2 .

For an *initial* process P_0 , we say that a finite or infinite sequence $\alpha_0 \alpha_1 \dots$ of actions from \mathbb{A}_τ is a *trace* of P_0 , if there is a sequence $P_0 \xrightarrow{\alpha_0} P_1 \xrightarrow{\alpha_1} \dots$ of action transitions, possibly ending with a process P_n .

Observe the following for SUM_{a2} : due to our syntax, P_1 in $P_1 \pm P_2$ is an initial process, i.e. has no components marked as urgent, and the same applies to P'_1 . Thus, $P_1 \pm P_2$ loses its urgency in a transition according to SUM_{a2} ; this corresponds to our intuition, since this atomic component (i.e. without parallel subcomponents) performs an action, which it had to perform urgently, and can afterwards wait with any further activity for time 1.

2.2.2 The temporal behaviour of PAFAS^c process

In addition to the purely functional transitions, we also consider transitions corresponding to the passage of one unit of time. The function **urgent** we exploit marks the *enabled* parallel components of a process as urgent; such a component can be identified with a dynamic operator (an action

or a choice), which gets underlined. This marking occurs when a time step is performed, because afterwards the marked components have to act in zero time – unless they are disabled. If such an urgent component acts, it should lose its urgency; and indeed, the marking vanishes with the dynamic operator. The next time step will only be possible, if no component is marked as urgent.

Definition 2.15 (*time step, timed execution sequence*) For $P \in \tilde{\mathbb{P}}_1$, we write $P \xrightarrow{1} Q$ when $Q = \text{urgent}(P)$, where $\text{urgent}(P)$ abbreviates $\text{urgent}(P, \emptyset)$ and $\text{urgent}(P, A)$ is defined as in Definition 2.7 but rule Sum is replaced as follows:

$$\text{Sum: } \text{urgent}(P_1 + P_2, A) = \begin{cases} P_1 \underline{\pm} P_2 & \text{if } (\mathcal{A}(P_1) \cup \mathcal{A}(P_2)) \setminus A \neq \emptyset \\ P_1 + P_2 & \text{otherwise} \end{cases}$$

As in the corresponding Section 2.1.2, we define *timed execution sequences* to be infinite sequences of action transitions and time steps starting with some $P \xrightarrow{1} Q_0$ (again a maximal sequence of such transitions/steps starting is never finite) and the property *non-Zeno*.

Example 2.16 Again we provide an example for the use of the various definitions and for this we use process P already introduced in Example 2.8. According to the transitional rules in Definition 2.14 and 2.15 we have:

$$\begin{aligned} P &\xrightarrow{1} (\underline{R} \parallel_{\emptyset} \underline{W}) \parallel_{\{r,w\}} \underline{V} = ((\text{rec } x. \underline{r}.x) \parallel_{\emptyset} (\text{rec } x. \underline{w}.x)) \parallel_{\{r,w\}} \text{rec } x. (r.x \underline{\pm} w.x) \\ &\xrightarrow{r} (R \parallel_{\emptyset} \underline{W}) \parallel_{\{r,w\}} V \\ &\xrightarrow{r} (R \parallel_{\emptyset} \underline{W}) \parallel_{\{r,w\}} V \end{aligned}$$

After one time unit, all the enabled parallel components become urgent in process P . In the second transition, \underline{R} and \underline{V} synchronize on action r . In the case of fairness of components, component \underline{W} remains enabled and thus urgent, and it will indeed remain urgent until it synchronizes on the action w ; cf. Definition 2.13, where the Par-case refers to activated actions of the context – and not to urgent ones as in Definition 2.4. We refer the reader to [4] for a detailed description of the various definitions.

2.2.3 Fairness of components and timing

As in the corresponding Section 2.1.3, we can now define (weak) *fairness w.r.t. components* in terms of non-Zeno timed execution sequences. In fact, *fair traces* (w.r.t. fairness of components) can be defined just as in Definition 2.9 by replacing each action transition $\xrightarrow{\alpha}$ and time step $\xrightarrow{1}$ with its counterpart in the component-oriented timed operational semantics, i.e. $\xrightarrow{\alpha}$ and $\xrightarrow{1}$. To keep things short, we do not report here the formal definition.

Example 2.17 In the case of fairness of components, the infinite computation of r 's in Example 2.10 is not fair intuitively because component W is enabled at every stage but it never performs w .

As already shown above, also in the component case this intuition corresponds to our formal definition in terms of timed runs, where components play the role of actions. Whenever P lets time pass evolving to $(\underline{R} \parallel_{\emptyset} \underline{W}) \parallel_{\{r,w\}} \underline{V}$, then a new time step is only possible when components R , W and V evolve by performing actions r and w . As a consequence, in any fair trace of P an infinite sequence of readings cannot indefinitely block the other process wishing to write the variable.

We conclude this section stating some properties of functions $\text{urgent}(_)$ and $\text{unmark}(_)$ which will be useful in the rest of the paper. A detailed proof of the statements can be found in [4].

Lemma 2.18 Let $Q \in \tilde{\mathbb{P}}_c$ and $P = \text{unmark}(Q) \in \tilde{\mathbb{P}}_1$. Then:

1. $Q \xrightarrow{\alpha} Q'$ implies $P \xrightarrow{\alpha} P'$ with $P' = \text{unmark}(Q')$;
2. $P \xrightarrow{\alpha} P'$ implies $Q \xrightarrow{\alpha} Q'$ with $P' = \text{unmark}(Q')$.

3 Dekker's Algorithm and its Liveness Property

In this section we briefly describe Dekker's mutex algorithm. There are two processes P_1 and 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, but taken to be 1 here.

Informally, the b variables are "request" variables and the variable k is a "turn" variable. Variable b_i is *true* if P_i is requesting entry to its critical section; variable k is i if it is P_i 's turn to enter its critical section. Only P_i writes the variable b_i , but both processes read it.

The i th 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⟩;
     $b_i = \text{true}$ ;
    while  $b_j$  do
      if  $k = j$  then begin
         $b_i := \text{false}$ ;
        while  $k = j$  do skip;
         $b_i := \text{true}$ ;
      end;
    ⟨critical section⟩;
     $k := j$ ;
     $b_i := \text{false}$ ;
  end;
end;

```

3.1 Translating the Algorithm into PAFAS Processes

In our translation of the algorithm into PAFAS, we use essentially the same coding as given by Walker in [17] – with some trivial changes due to the different language we are using (PAFAS instead of CCS). Each program variable is represented as a family of processes. Thus, for instance, the process $B_1(\text{false})$ denotes the variable b_1 with value *false*. The *sort* of the process $B_1(\text{false})$ is the set $\{b_1rt, b_1rf, b_1wf, b_1wt\}$ where b_1rf and b_1rt represent the actions of reading the values *false* and *true* from b_1 , b_1wf and b_1wt represent, respectively, the writing of the values *false* and *true* into b_1 . In the following we denote with $\mathbb{B} = \{\text{false}, \text{true}\}$ the set of boolean values and with $\mathbb{K} = \{1, 2\}$ the set of values that the variable k may take.

Definition 3.1 (*program variables*) Let $i \in \{1, 2\}$. We define the processes representing program variables as follows:

$$\begin{aligned}
B_i(\text{false}) &= b_irtf.B_i(\text{false}) + (b_irtf.B_i(\text{false}) + b_irtf.B_i(\text{true})) \\
B_i(\text{true}) &= b_irtf.B_i(\text{true}) + (b_irtf.B_i(\text{false}) + b_irtf.B_i(\text{true})) \\
K(1) &= kr1.K(1) + (kw1.K(1) + kw2.K(2)) \\
K(2) &= kr2.K(2) + (kw1.K(1) + kw2.K(2))
\end{aligned}$$

Let $B = \{b_irtf, b_irtf, b_irtf, b_irtf \mid i \in \{1, 2\}\} \cup \{kr1, kr2, kw1, kw2\}$ 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 $b_1, b_2 \in \mathbb{B}$ and $k \in \mathbb{K}$, we define

$$PV(b_1, b_2, k) = (B_1(b_1) \parallel B_2(b_2)) \parallel K(k),$$

where we use \parallel as a shorthand for \parallel_{\emptyset} .

The next definition introduces the PAFAS version of Dekker’s algorithm.

Definition 3.2 (*the algorithm*) The processes P_1 and P_2 are represented by the following PAFAS processes; the actions \mathbf{req}_i and \mathbf{cs}_i have been added to indicate the request to enter and the execution of the critical section by the process P_i .

$$\begin{array}{ll}
P_1 = \mathbf{req}_1.b_1wt.P_{11} + \tau.P_1 & P_2 = \mathbf{req}_2.b_2wt.P_{21} + \tau.P_2 \\
P_{11} = b_2rf.P_{14} + b_2rt.P_{12} & P_{21} = b_1rf.P_{24} + b_1rt.P_{22} \\
P_{12} = kr1.P_{11} + kr2.b_1wf.P_{13} & P_{22} = kr2.P_{21} + kr1.b_2wf.P_{23} \\
P_{13} = kr1.b_1wt.P_{11} + kr2.P_{13} & P_{23} = kr2.b_2wt.P_{21} + kr1.P_{23} \\
P_{14} = \mathbf{cs}_1.kw2.b_1wf.P_1 & P_{24} = \mathbf{cs}_2.kw1.b_2wf.P_2
\end{array}$$

Finally, the algorithm can be defined as $Dekker = ((P_1 \parallel P_2) \parallel_B \text{PV}(false, false, 1))[\Phi_B]$. The sort of $Dekker$ is the set $\mathbb{A}_d = \{\mathbf{req}_i, \mathbf{cs}_i \mid i = 1, 2\}$.

The definition of Dekker in PAFAS closely follows the one in CCS given by Walker in [17]. Only a couple of changes have been implemented, and neither of them has an influence on the validity of the safety property. They all concern the definitions of P_1 (and P_2 resp.).

In [17], $P_1 = b_1wt.\mathbf{req}_1.P_{11}$, i.e. process P_1 first sets its request variable to true and then performs the request. Walker regarded \mathbf{req}_1 as a *probe*, which just indicates that requesting has started. But there is a problem: in order to request access, P_1 must first set its request variable, which is also used by the other process; therefore, the other process might block the request. This can really happen even under a reasonable fairness assumption as we will demonstrate below. Such a behaviour would give the wrong impression that the liveness property is satisfied – P_1 just never requested; to avoid such situations, requesting must be completely under the control of the respective process, and to ensure this we have swapped the write and the request actions.

Furthermore, we have added a $\tau.P_1$ -summand; without it, any of the two forms of weak fairness would force P_1 to request. This is not desirable, since in reality P_1 can decide to stay idle. In fact, with forced requesting the simple strategy of granting access to the critical section in turns would solve the mutex problem easily. In order to consider the real, much more difficult problem, one has to model the situation in which process P_1 decides to idle, and that we have done.

The third change is motivated by economy: for our approach, it suffices to have one action \mathbf{cs}_1 to denote performance of the critical section instead of two actions for entering and exiting as in [17].

3.2 Liveness Property of Dekker’s Algorithm

As discussed in the introduction, a mutex algorithm satisfies its liveness property if whenever at any point in any computation a process P_i requests the execution of its critical section, then, in any continuation of that computation, there is a point at which P_i will perform its critical section. We can expect this property to hold only under some fairness assumption; so for the formal property we want to check, we replace ‘computation’ by ‘fair trace’ (in one of our two interpretations). In other words, a mutex algorithm satisfies its *liveness property* if any occurrence of \mathbf{req}_i in a fair trace is eventually followed by \mathbf{cs}_i , $i = 1, 2$.

Due to our definition of fair trace, this amounts to checking that each non-Zeno timed execution sequence is live according to the following definition.

Definition 3.3 (*live*) Let $P_0 \in \mathbb{P}_1$, $\lambda_0, \lambda_1, \dots \in (\mathbb{A}_d \cup \{\tau\} \cup \{1\})$. A *timed execution sequence* γ from P_0 with $\gamma = P_0 \xrightarrow{1} Q_0 \xrightarrow{\lambda_0} Q_1 \xrightarrow{\lambda_1} \dots$ ($\gamma = P_0 \xrightarrow{1} Q_0 \xrightarrow{\lambda_0} Q_1 \xrightarrow{\lambda_1} \dots$) is *not live* if there exists $j \in \mathbb{N}_0$ such that $\lambda_j = \mathbf{req}_i$ and \mathbf{cs}_i is not performed in the execution sequence $Q_{j+1} \xrightarrow{\lambda_{j+1}} Q_{j+2} \xrightarrow{\lambda_{j+2}} \dots$ ($Q_{j+1} \xrightarrow{\lambda_{j+1}} Q_{j+2} \xrightarrow{\lambda_{j+2}} \dots$ respectively). Otherwise, we say that γ is live.

4 Fairness of Actions and Liveness

This section shows that fairness of actions is not sufficiently strong to ensure the liveness property. For this purpose, we present two fair traces with respect to fairness of actions, which violate the liveness property, i.e. two non-Zeno timed execution sequences in PAFAS (cf. Section 2.1) which are not live. First of all, we need to describe how program variables and the processes P_1 and P_2 evolve by letting one time unit pass.

Definition 4.1 (*urgent program variables*) According to Definitions 3.1 and 2.7, *urgent program variables* can be defined as follows:

$$\begin{aligned} \underline{B}_i(\text{false}) &= \underline{b}_i \text{rf}.B_i(\text{false}) + (\underline{b}_i \text{wf}.B_i(\text{false}) + \underline{b}_i \text{wt}.B_i(\text{true})) \\ \underline{B}_i(\text{true}) &= \underline{b}_i \text{rt}.B_i(\text{true}) + (\underline{b}_i \text{wf}.B_i(\text{false}) + \underline{b}_i \text{wt}.B_i(\text{true})) \end{aligned}$$

$$\begin{aligned} \underline{K}(1) &= \underline{kr}1.K(1) + (\underline{kw}1.K(1) + \underline{kw}2.K(2)) \\ \underline{K}(2) &= \underline{kr}2.K(2) + (\underline{kw}1.K(1) + \underline{kw}2.K(2)) \end{aligned}$$

We further introduce $\underline{\mathbb{B}} = \{\text{false}, \text{true}\}$ and $\underline{\mathbb{K}} = \{1, 2\}$. Then, given $b'_1, b'_2 \in \underline{\mathbb{B}} \cup \underline{\mathbb{B}}$ and $k' \in \underline{\mathbb{K}} \cup \underline{\mathbb{K}}$, we define $\text{PV}(b'_1, b'_2, k') = ((B_1 \parallel B_2) \parallel K)$, where:

$$B_i = \begin{cases} B_i(b) & \text{if } b'_i = b \in \underline{\mathbb{B}} \\ \underline{B}_i(b) & \text{if } b'_i = \underline{b} \in \underline{\mathbb{B}} \end{cases} \quad K = \begin{cases} K(k) & \text{if } k' = k \in \underline{\mathbb{K}} \\ \underline{K}(k) & \text{if } k' = \underline{k} \in \underline{\mathbb{K}} \end{cases}$$

As an example, we have that $\text{PV}(\underline{\text{true}}, \underline{\text{false}}, \underline{2}) = (\underline{B}_1(\text{true}) \parallel B_2(\text{false})) \parallel \underline{K}(2)$.

Similarly, the urgent versions of processes P_1 and P_2 can be defined as follows:

Definition 4.2 (*urgent processes*)

$$\begin{aligned} \underline{P}_1 &= \underline{\text{req}}_1.b_1 \text{wt}.P_{11} + \tau.P_1 & \underline{P}_2 &= \underline{\text{req}}_2.b_2 \text{wt}.P_{21} + \tau.P_2 \\ \underline{P}_{11} &= \underline{b}_2 \text{rf}.P_{14} + \underline{b}_2 \text{rt}.P_{12} & \underline{P}_{21} &= \underline{b}_1 \text{rf}.P_{24} + \underline{b}_1 \text{rt}.P_{22} \\ \underline{P}_{12} &= \underline{kr}1.P_{11} + \underline{kr}2.b_1 \text{wf}.P_{13} & \underline{P}_{22} &= \underline{kr}2.P_{21} + \underline{kr}1.b_2 \text{wf}.P_{23} \\ \underline{P}_{13} &= \underline{kr}1.b_1 \text{wt}.P_{11} + \underline{kr}2.P_{13} & \underline{P}_{23} &= \underline{kr}2.b_2 \text{wt}.P_{21} + \underline{kr}1.P_{23} \\ \underline{P}_{14} &= \underline{\text{cs}}_1.kw2.b_1 \text{wf}.P_1 & \underline{P}_{24} &= \underline{\text{cs}}_2.kw1.b_2 \text{wf}.P_2 \end{aligned}$$

As a consequence of the above definitions (and by the action-oriented operational semantics) we have that *Dekker* can let one time unit pass evolving into $\underline{\text{Dekker}}$, where:

$$\underline{\text{Dekker}} = ((\underline{P}_1 \parallel \underline{P}_2) \parallel_B \text{PV}(\underline{\text{false}}, \underline{\text{false}}, 1))[\Phi_B]$$

Our first example shows how an infinite τ -loop can result in the starvation of both processes.

Example 4.3 Let us consider the following timed computation from *Dekker*:

$$\begin{aligned} \text{Dekker} \xrightarrow{1} \underline{\text{Dekker}} &= ((\underline{P}_1 \parallel \underline{P}_2) \parallel_B \text{PV}(\underline{\text{false}}, \underline{\text{false}}, 1))[\Phi_B] && \xrightarrow{\underline{\text{req}}_1} \xrightarrow{\underline{\text{req}}_2} \\ &((b_1 \text{wt}.P_{11} \parallel b_2 \text{wt}.P_{21}) \parallel_B \text{PV}(\underline{\text{false}}, \underline{\text{false}}, 1))[\Phi_B] && \xrightarrow{\tau} \\ &((P_{11} \parallel b_2 \text{wt}.P_{21}) \parallel_B \text{PV}(\text{true}, \underline{\text{false}}, 1))[\Phi_B] && \xrightarrow{\tau} \\ &((P_{11} \parallel P_{21}) \parallel_B \text{PV}(\text{true}, \text{true}, 1))[\Phi_B] && \xrightarrow{\tau} \\ &((P_{11} \parallel P_{22}) \parallel_B \text{PV}(\text{true}, \text{true}, 1))[\Phi_B] && \xrightarrow{\tau} \end{aligned}$$

$$\begin{aligned}
P_0 &= ((P_{11} \parallel b_2 wf.P_{23}) \parallel_B PV(true, true, 1))[\Phi_B] \xrightarrow{1} \\
Q_0 &= ((\underline{P}_{11} \parallel \underline{b_2 wf.P_{23}}) \parallel_B PV(true, \underline{true}, 1))[\Phi_B] \xrightarrow{\tau} \\
&((P_{12} \parallel b_2 wf.P_{23}) \parallel_B PV(true, true, 1))[\Phi_B] \xrightarrow{\tau} \\
P_0 &= ((P_{11} \parallel b_2 wf.P_{23}) \parallel_B PV(true, true, 1))[\Phi_B]
\end{aligned}$$

Repeating the last three transitions, we get a non-Zeno timed execution sequence that is not live, i.e. *Dekker* can perform a fair trace

$$Dekker \xrightarrow{\text{req}_1 \text{ req}_2 \tau^4} P_0 \xrightarrow{\tau^2} P_0 \xrightarrow{\tau^2} P_0 \dots$$

that violates liveness since no process will ever enter its critical section. More intuitively speaking, once in P_0 , repeated reading of variables b_2 and k blocks indefinitely P_2 which will never set its request variable b_2 to false. We have already indicated in Example 2.8 how fairness of actions allows computations along which repeated accesses to a variable can block another access. On the other hand, P_1 cannot enter its critical section and, hence, cannot proceed until the value of b_2 is true. As a consequence, both processes are stuck; this undesirable behaviour would not be detected with the liveness definition of [17], since no process enters its critical section.

The next example shows a different kind of computation which also causes a violation of liveness; along such a computation, one process is stuck while the other repeatedly executes its critical section.

Example 4.4 Now, consider the following timed computation:

$$\begin{aligned}
Dekker \xrightarrow{1} \underline{Dekker} &= ((\underline{P}_1 \parallel \underline{P}_2) \parallel_B PV(false, false, 1))[\Phi_B] && \xrightarrow{\text{req}_1} \xrightarrow{\text{req}_2} \\
P_0 &= ((b_1 wt.P_{11} \parallel b_2 wt.P_{21}) \parallel_B PV(false, false, 1))[\Phi_B] && \xrightarrow{1} \\
&((\underline{b_1 wt.P_{11}} \parallel \underline{b_2 wt.P_{21}}) \parallel_B PV(\underline{false}, \underline{false}, 1))[\Phi_B] && \xrightarrow{\tau} \\
&((\underline{b_1 wt.P_{11}} \parallel P_{21}) \parallel_B PV(\underline{false}, true, 1))[\Phi_B] && \xrightarrow{\tau} \\
&((b_1 wt.P_{11} \parallel P_{24}) \parallel_B PV(false, true, 1))[\Phi_B] && \xrightarrow{\text{cs}_2} \\
&((b_1 wt.P_{11} \parallel kw1.b_2 wf.P) \parallel_B PV(false, true, 1))[\Phi_B] && \xrightarrow{\tau} \\
&((b_1 wt.P_{11} \parallel b_2 wf.P_2) \parallel_B PV(false, true, 1))[\Phi_B] && \xrightarrow{\tau} \\
&((b_1 wt.P_{11} \parallel P_2) \parallel_B PV(false, false, 1))[\Phi_B] && \xrightarrow{\text{req}_2} \\
&((b_1 wt.P_{11} \parallel b_2 wt.P_{21}) \parallel_B PV(false, false, 1))[\Phi_B] = P_0
\end{aligned}$$

Again, the trace performed in

$$Dekker \xrightarrow{\text{req}_1 \text{ req}_2} P_0 \xrightarrow{\tau^2 \text{ cs}_2 \tau \tau \text{ req}_2} P_0 \dots$$

is fair but violates liveness since P_1 never enters its critical section. Here, P_2 repeatedly executes its critical section, again preventing P_1 to set its request variable b_1 to *true*. As a consequence, P_1

cannot enter its critical section even though the value of turn variable k is 1. In other words, writing a value to a variable (b_1wt) is blocked by repeated reading of the variable (b_1rt); accordingly, P_1 could be prevented from requesting in the modelling of Walker.

5 Fairness of Components and Liveness

This section is the core of the paper. It proves that any fair trace of *Dekker* according to fairness of components satisfies the liveness property, i.e. that any non-Zeno timed execution sequence of *Dekker* in PAFAS^C is live in the sense of Definition 3.3. We will present three ideas that help to reduce the number of states we have to deal with.

5.1 Permanently Lazy Components

The state space of a process in PAFAS^C is considerably larger than in an untimed process algebra because process components switch from lazy to urgent. In principle, for a process with 5 components like *Dekker*, this could make the state space 32-fold in size. We can achieve a considerable reduction, if we prevent this by declaring some components as permanently lazy, as we describe it in this subsection. As an application, we will regard the three program variables as one component of *Dekker* for technical reasons, and declare it as permanently lazy; this results in a process denoted by *Dekker*[PV].

Technically, this declaration means that a non-Zeno timed execution sequence of the original process can be simulated by one of the new process, but not vice versa. This way, instead of proving that all non-Zeno timed execution sequences of *Dekker* are live it is sufficient to prove that all non-Zeno timed execution sequences of *Dekker*[PV] are live.

Although this is only a sufficient condition, it should better be satisfied in our example since there is a good intuitive reason behind it. Since fairness is required for all components, a program variable can intuitively speaking enforce to be read or written – provided there is always some component that could do so. But our intuition for variables is that they are passive, that we really only want fairness towards P_1 and P_2 . As it turns out, assuming this kind of fairness is indeed enough.

We now extend PAFAS^C with a new operator, which intuitively speaking can only be applied to a top-level component.

Definition 5.1 (*permanently lazy processes*) Given $P \in \tilde{\mathbb{P}}_1$, we define the *permanently lazy version* of P , written $[P]$, to be the process with the same syntactical structure of P (and, hence, the same functional behaviour) but which permanently ignores the passage of time. The timed operational semantics of $[P]$ can be defined by the following rules:

$$\text{ACT}_L \frac{P \xrightarrow{\alpha} P'}{[P] \xrightarrow{\alpha} [P']} \quad \text{TIME}_L \frac{}{[P] \xrightarrow{1} [P]}$$

The set $\tilde{\mathbb{P}}_{\ell_1}$ of *initial processes with one permanently lazy top-level component* is generated by the following grammar:

$$S ::= P \parallel_A [P] \mid S[\Phi]$$

where $P \in \tilde{\mathbb{P}}_1$, $A \subseteq \mathbb{A}$ (possibly infinite) and Φ is a general relabelling function.

Similarly, the set $\tilde{\mathbb{P}}_{\ell}$ of (*general*) *processes with one permanently lazy top-level component* is generated by the following grammar:

$$R ::= Q \parallel_A [P] \mid R[\Phi]$$

where $Q \in \tilde{\mathbb{P}}_c$, $P \in \tilde{\mathbb{P}}_1$, $A \subseteq \mathbb{A}$ (possibly infinite) and Φ is a general relabelling function.

Next, we want to define the operational semantics for processes with one permanently lazy top-level component. As in PAFAS^C, the respective SOS-rules exploit a function $\text{clean}(_)$ on process terms which, as in the previous sections, removes all inactive urgencies, and we also use a function $\text{unmark}(_)$ which removes all urgencies (inactive or not). Both these functions can be defined by induction on the process structure as follows:

Definition 5.2 (*cleaning urgencies*) Let R be a $\tilde{\mathbb{P}}_\ell$ term. Then $\text{clean}(R)$ and $\text{unmark}(R)$ are defined by induction on R as follows:

$$\begin{array}{l} \text{Par:} \quad \text{clean}(Q \parallel_A [P]) = \text{clean}(Q, A_1) \parallel_A [P] \quad \text{where } A_1 = (\mathcal{A}(Q) \setminus \mathcal{A}(P)) \cap A \\ \text{Rel:} \quad \text{clean}(R[\Phi]) = \text{clean}(R)[\Phi] \end{array}$$

$$\begin{array}{l} \text{Par:} \quad \text{unmark}(Q \parallel_A [P]) = \text{unmark}(Q) \parallel_A [P] \\ \text{Rel:} \quad \text{unmark}(R[\Phi]) = \text{unmark}(R)[\Phi] \end{array}$$

The functional behaviour of $\tilde{\mathbb{P}}_\ell$ terms can be defined as follows:

Definition 5.3 (*Functional operational semantics*) The following SOS-rules define the transition relations $\vdash^\alpha \subseteq (\tilde{\mathbb{P}}_\ell \times \tilde{\mathbb{P}}_\ell)$ for $\alpha \in \mathbb{A}_\tau$, the *action transitions*.

$$\begin{array}{l} \text{LAZYPAR}_{a1} \frac{\alpha \notin A, Q \vdash^\alpha Q'}{Q \parallel_A [P] \vdash^\alpha \text{clean}(Q' \parallel_A [P])} \quad \text{LAZYPAR}_{a2} \frac{\alpha \notin A, P \vdash^\alpha P'}{Q \parallel_A [P] \vdash^\alpha \text{clean}(Q \parallel_A [P'])} \\ \text{LAZYSYNCH} \frac{\alpha \in A, Q \vdash^\alpha Q', P \vdash^\alpha P'}{Q \parallel_A [P] \vdash^\alpha \text{clean}(Q' \parallel_A [P'])} \quad \text{LAZYREL}_a \frac{R \vdash^\alpha R'}{R[\Phi] \vdash^\alpha R'[\Phi]} \end{array}$$

Definition 5.4 (*time step*) For $S \in \tilde{\mathbb{P}}_{\ell 1}$, we write that $S \xrightarrow{1} R$ when $R = \text{urgent}(S)$ where function $\text{urgent}(S)$ is defined as follows:

$$\begin{array}{l} \text{Par:} \quad \text{urgent}(P_1 \parallel_A [P_2]) = \text{urgent}(P_1, A_1) \parallel_A [P_2] \quad \text{where } A_1 = (\mathcal{A}(P_1) \setminus \mathcal{A}(P_2)) \cap A \\ \text{Rel:} \quad \text{urgent}(S[\Phi]) = \text{urgent}(S)[\Phi] \end{array}$$

Definition 5.5 Let $Q \in \tilde{\mathbb{P}}$ and $R \in \tilde{\mathbb{P}}_\ell$. We write that $Q \prec R$ if either $Q = Q_1 \parallel_A Q_2$ and $R = Q_1 \parallel_A [\text{unmark}(Q_2)]$ or $Q = Q_1[\Phi]$ and $R = R_1[\Phi]$ with $Q_1 \prec R_1$.

Let us denote by $\text{Dekker}[\text{PV}]$ the process $((P_1 \parallel_B P_2) \parallel [\text{PV}(\text{false}, \text{false}, 1)])(\Phi_B)$, which is *Dekker* with permanently lazy process variables. We prove that any non-Zeno timed execution sequence of *Dekker* can be simulated by a corresponding non-Zeno timed execution sequence of $\text{Dekker}[\text{PV}]$ (see Proposition 5.9). This allows us to consider, for our calculation, $\text{Dekker}[\text{PV}]$ (instead of *Dekker*).

Lemma 5.6 Let $Q \in \tilde{\mathbb{P}}$ and $R, R' \in \tilde{\mathbb{P}}_\ell$. Then:

1. $Q \prec R$ and $Q \in \tilde{\mathbb{P}}_1$ imply $R \in \tilde{\mathbb{P}}_{\ell 1}$;
2. $Q \prec R$ and $Q \prec R'$ imply $R = R'$.

Proposition 5.7 Let $Q \in \tilde{\mathbb{P}}$ and $R \in \tilde{\mathbb{P}}_\ell$ with $Q \prec R$. Then:

1. $Q \vdash^\alpha Q'$ implies $R \vdash^\alpha R'$ and $Q' \prec R'$;

2. $R \xrightarrow{\alpha} R'$ implies $Q \xrightarrow{\alpha} Q'$ and $Q' \prec R'$.

Proof: We only prove Item 1 (Item 2 can be proved similarly). By Definition 5.5 we have to consider two possible cases:

Par: $Q = Q_1 \parallel_A Q_2$ and $R = Q_1 \parallel_A [P_2]$ with $P_2 = \text{unmark}(Q_2)$. Assume that $Q \xrightarrow{\alpha} Q'$ and consider the following possible cases:

- $\alpha \notin A$, $Q_1 \xrightarrow{\alpha} Q'_1$ and $Q' = \text{clean}(Q'_1 \parallel_A Q_2) = \text{clean}(Q'_1, A_1) \parallel_A \text{clean}(Q_2, A_2)$ with $A_1 = (\mathcal{A}(Q'_1) \setminus \mathcal{A}(Q_2)) \cap A = (\mathcal{A}(Q'_1) \setminus \mathcal{A}(P_2)) \cap A$ (since $\mathcal{A}(Q_2) = \mathcal{A}(\text{unmark}(Q_2)) = \mathcal{A}(P_2)$) and $A_2 = (\mathcal{A}(Q_2) \setminus \mathcal{A}(Q'_1)) \cap A$. On the other hand $\alpha \notin A$ and $Q_1 \xrightarrow{\alpha} Q'_1$ imply, by rule LAZYPAR_{a1}, $R \xrightarrow{\alpha} \text{clean}(Q'_1 \parallel_A [P_2]) = \text{clean}(Q'_1, A_1) \parallel_A [P_2] = R'$. Finally, $\text{unmark}(\text{clean}(Q_2, A_2)) = \text{unmark}(Q_2) = P_2$ implies $Q' \prec R'$.
- $\alpha \notin A$, $Q_2 \xrightarrow{\alpha} Q'_2$ and $Q' = \text{clean}(Q_1 \parallel_A Q'_2) = \text{clean}(Q_1, A_1) \parallel_A \text{clean}(Q'_2, A_2)$ with $A_1 = (\mathcal{A}(Q_1) \setminus \mathcal{A}(Q'_2)) \cap A$ and $A_2 = (\mathcal{A}(Q'_2) \setminus \mathcal{A}(Q_1)) \cap A$. Moreover, by Lemma 2.18, $Q_2 \xrightarrow{\alpha} Q'_2$ implies $P_2 = \text{unmark}(Q_2) \xrightarrow{\alpha} \text{unmark}(Q'_2) = P'_2$. Thus, $R \xrightarrow{\alpha} \text{clean}(Q_1 \parallel_A [P'_2]) = \text{clean}(Q_1, A_1) \parallel_A [P'_2] = R'$ (since $(\mathcal{A}(Q_1) \setminus \mathcal{A}(P'_2)) \cap A = (\mathcal{A}(Q_1) \setminus \mathcal{A}(Q'_2)) \cap A = A_1$). Finally, $\text{unmark}(\text{clean}(Q'_2, A_2)) = \text{unmark}(Q'_2) = P'_2$ implies $Q' \prec R'$.
- $\alpha \in A$, $Q_1 \xrightarrow{\alpha} Q'_1$, $Q_2 \xrightarrow{\alpha} Q'_2$ and $Q' = \text{clean}(Q'_1 \parallel_A Q'_2)$. A mix of the proofs of the above two cases.

Rel: $Q = Q_1[\Phi]$ and $R = R_1[\Phi]$ with $Q_1 \prec R_1$. In this case $Q \xrightarrow{\alpha} Q'$ if there exists $\beta \in \Phi^{-1}(\alpha)$ such that $Q_1 \xrightarrow{\beta} Q'_1$ and $Q' = Q'_1[\Phi]$. By induction hypothesis we have that $R_1 \xrightarrow{\beta} R'_1$ with $Q'_1 \prec R'_1$. Thus, $R \xrightarrow{\alpha} R'_1[\Phi] = R'$ and $Q' \prec R'$.

□

Proposition 5.8 Let $P \in \tilde{\mathbb{P}}_1$ and $S \in \tilde{\mathbb{P}}_{\ell 1}$ with $P \prec S$. Then $\text{urgent}(P) \prec \text{urgent}(S)$.

Proof: Again we have the following two possible cases:

Par: $P = P_1 \parallel_A P_2$ and $S = P_1 \parallel_A [P_2]$. $\text{urgent}(P_1 \parallel_A P_2) = \text{urgent}(P_1, A_1) \parallel_A \text{urgent}(P_2, A_2) = Q_1 \parallel_B Q_2$ where $A_1 = (\mathcal{A}(P_1) \setminus \mathcal{A}(P_2)) \cap A$ and $A_2 = (\mathcal{A}(P_2) \setminus \mathcal{A}(P_1)) \cap A$. On the other hand, $\text{urgent}(P_1 \parallel_A [P_2]) = \text{urgent}(P_1, A_1) \parallel_A [P_2] = Q_1 \parallel_A [P_2] = R$. Finally, $\text{unmark}(Q_2) = \text{unmark}(\text{urgent}(P_2, A_2)) = P_2$ implies $\text{urgent}(P) \prec \text{urgent}(S)$.

Rel: $P = P_1[\Phi]$ and $S = S_1[\Phi]$ with $P_1 \prec S_1$. By induction hypothesis, we have also $\text{urgent}(P) = \text{urgent}(P_1)[\Phi] \prec \text{urgent}(S_1)[\Phi] = \text{urgent}(S)$.

□

By iterative applications of Propositions 5.7 and 5.8 we can prove the following statement:

Proposition 5.9 Let $P \in \tilde{\mathbb{P}}_1$, $S \in \tilde{\mathbb{P}}_{\ell 1}$ with $P \prec S$ and $v \in (\mathbb{A}_\tau)^*$. Then $P \xrightarrow{1} Q \xrightarrow{v} P' \in \tilde{\mathbb{P}}_1$ implies $S \xrightarrow{1} R \xrightarrow{v} S'$ with $P' \prec S'$ (and hence S can simulate each non-Zeno timed execution sequence of P).

5.2 F-Steps

We can group the transitions of a non-Zeno timed execution sequence into infinitely many steps of the form $S \xrightarrow{1} R \xrightarrow{v} S'$, where $v \in (\mathbb{A}_\tau)^*$ and S' is the next process to perform a time step. Such a step is minimal in a sense, if S' is the first process in the transition sequence $R \xrightarrow{v} S'$ that could perform a time step, i.e. the first initial process. We call such minimal steps f-steps and the processes reachable by them fair-reachable. We will show in this subsection that we only have to consider timed execution sequences built from infinitely many such f-steps.

Definition 5.10 (*f-executions*) A transition sequence $S \xrightarrow{1} R \xrightarrow{v} S'$ with $S, S' \in \mathbb{P}_{\ell_1}$ and $v \in (\mathbb{A}_\tau)^*$ is an *f-step* if S' is the only initial process in the transition sequence $R \xrightarrow{v} S'$ (allowing $R = S'$ if v is the empty sequence). An *f-execution* from $S_0 \in \mathbb{P}_{\ell_1}$ is any infinite sequence of f-steps of the form:

$$\gamma = S_0 \xrightarrow{1} R_0 \xrightarrow{v_0} S_1 \xrightarrow{1} R_1 \xrightarrow{v_1} S_2 \dots$$

We call the processes S_1, S_2, \dots *fair-reachable* from S_0 .

According to this definition, f-executions are special non-Zeno timed execution sequences. To show that checking them for liveness suffices, we need the following lemmas.

Lemma 5.11 Let $P_0, P_1, P_2 \in \mathbb{P}_1$, v_0 and $v_1 \in (\mathbb{A}_\tau)^*$. Then: $P_0 \xrightarrow{v_0} P_1 \xrightarrow{1} Q_1 \xrightarrow{v_1} P_2$ implies $P_0 \xrightarrow{1} Q_0 \xrightarrow{v_0 v_1} P_2$.

Proof: This can be proven by collecting some results stated in [4]. □

Lemma 5.12 Let $S_0, S_1, S_2 \in \mathbb{P}_{\ell_1}$, v_0 and $v_1 \in (\mathbb{A}_\tau)^*$. Then: $S_0 \xrightarrow{v_0} S_1 \xrightarrow{1} R_1 \xrightarrow{v_1} S_2$ implies $S_0 \xrightarrow{1} R_0 \xrightarrow{v_0 v_1} S_2$.

Proof: Let $S_0 \in \mathbb{P}_{\ell_1}$ and let P_0 be the \mathbb{P}_1 -process obtained from S_0 by removing laziness from its top-level permanent lazy component. By Definition 5.5, it is $P_0 \prec S_0$. Now, assume that $S_0 \xrightarrow{v_0} S_1 \xrightarrow{1} R_1 \xrightarrow{v_1} S_2$. Then $P_0 \prec S_0$, Propositions 5.7-2 and 5.9 imply $P_0 \xrightarrow{v_0} P_1 \xrightarrow{1} Q_1 \xrightarrow{v_1} P_2$ with $P_i \prec S_i$, for $i \in \{1, 2\}$. By Lemma 5.11 and (iterative applications of) Proposition 5.7-1, we also have that $P_0 \xrightarrow{1} Q_0 \xrightarrow{v_0 v_1} P_2$ and $S_0 \xrightarrow{1} R_0 \xrightarrow{v_0 v_1} S'_2$ with $P_2 \prec S'_2$. Finally, $P_2 \prec S'_2$ and $P_2 \prec S_2$ imply $S'_2 = S_2$ (see Lemma 5.6-2). □

Lemma 5.13 Let $S_0, S_1, S_2 \in \mathbb{P}_{\ell_1}$, $v_1 \in (\mathbb{A}_\tau)^*$ and $\delta_0, \delta_1 \in (\mathbb{A}_\tau \cup \{1\})^*$. If $S_0 \xrightarrow{\delta_0} S_1 \xrightarrow{v_1} S_2 \xrightarrow{1} R_2 \xrightarrow{\delta_1}$ is a non-Zeno timed execution sequence, then $S_0 \xrightarrow{\delta_0} S_1 \xrightarrow{1} R_1 \xrightarrow{v_1 \delta_1}$ is also a non-Zeno timed execution sequence.

Proof: Directly from Lemma 5.12, since in δ_1 we have infinitely many time-steps and, hence, “along δ_1 ” there will be an initial process playing the rôle of S_2 in Lemma 5.12. □

Proposition 5.14 For each non-Zeno timed execution sequence from $S_0 \in \mathbb{P}_{\ell_1}$

$$\gamma = S_0 \xrightarrow{1} R_0 \xrightarrow{v_0} S_1 \xrightarrow{1} R_1 \xrightarrow{v_1} S_2 \dots$$

there exists a corresponding f-execution

$$\gamma' = S'_0 \xrightarrow{1} R'_0 \xrightarrow{v'_0} S'_1 \xrightarrow{1} R'_1 \xrightarrow{v'_1} S'_2 \dots$$

where $S'_0 = S_0$, $v_0 v_1 \dots = v'_0 v'_1 \dots$ and each step $S'_i \xrightarrow{1} R'_i \xrightarrow{v'_i} S'_{i+1}$ is minimal.

Proof: We use Lemma 5.13 repeatedly to move each time step to the first possible position. Observe that γ either deadlocks (ends with some process S_k performing $(\dashv\rightarrow)^\omega$) or always performs one or more actions between two consecutive time-steps, so the limit contains all actions. \square

By definition of fair-traces, Propositions 5.9 and 5.14, in order to check the liveness property of Dekker's algorithm for all fair traces, we only have to check that all f-executions of $Dekker[PV]$ are live. The initial processes we have to consider for this check must all be fair-reachable from $Dekker[PV]$.

5.3 Symmetry of Fair-Reachable Processes

Half of the processes which are fair-reachable from $Dekker[PV]$ are denoted by D_1, \dots, D_{47} . Their formal definition is provided in Tables 1 and 2 while a detailed description of the way in which they have been obtained can be found in the appendix³. We also consider all possible symmetries and use S_y to denote the process which is symmetric to D_y with respect to the local state of P_1 and P_2 and the value of the variables b_1 , b_2 and k . Namely, for each $y \in [1, 47]$, $S_y = \mathcal{S}(D_y)$, where the function $\mathcal{S}(_)$ on processes is given below. Moreover, $\mathcal{S}(S_y) = D_y$ for any y .

Definition 5.15 (*symmetric processes*)

Let $P_1, P_{11}, \dots, P_{14}, P_2, P_{21}, \dots, P_{24}$ be processes as given in Definition 3.2. Let moreover $x \in [1, 4]$ and $\{i, j\} = \{1, 2\}$. Then:

$$\begin{aligned} \mathcal{S}(P_i) &= P_j & \mathcal{S}(P_{ix}) &= P_{jx} \\ \mathcal{S}(b_i wt.P_{i1}) &= b_j wt.P_{j1} & \mathcal{S}(b_i wf.P_{i3}) &= b_j wf.P_{j3} \\ \mathcal{S}(kwj.b_i wf.P_i) &= kwi.b_j wf.P_j & \mathcal{S}(b_i wf.P_i) &= b_j wf.P_j \end{aligned}$$

Now, let $b_1, b_2 \in \mathbb{B}$, $k \in \mathbb{K}$ and $S = ((S_1 \parallel S_2) \parallel_B [PV(b_1, b_2, k)])[\Phi]$ be action-reachable from $Dekker[PV]$. We can define the symmetric process of S as follows:

$$\mathcal{S}(S) = ((\mathcal{S}(S_2) \parallel \mathcal{S}(S_1)) \parallel_B [PV(b_2, b_1, (k \bmod 2) + 1)])[\Phi_B]$$

We say that two processes S and S' action-reachable from $Dekker[PV]$ are symmetric, written $S \approx S'$, if either $S' = \mathcal{S}(S)$ or $S = \mathcal{S}(S')$.

Definition 5.16 (*symmetric sequences of actions*)

Let $v \in (\mathbb{A}_d \cup \{\tau\})^*$. Then, the string $\mathcal{S}(v)$ can be defined, by induction on the length of v , as follows:

$$\mathcal{S}(v) = \begin{cases} \varepsilon & \text{if } v = \varepsilon \\ \tau \mathcal{S}(v') & \text{if } v = \tau v' \\ \mathbf{req}_j \mathcal{S}(v') & \text{if } v = \mathbf{req}_i v' \text{ and } j = (i \bmod 2) + 1 \\ \mathbf{cs}_j \mathcal{S}(v') & \text{if } v = \mathbf{cs}_i v' \text{ and } j = (i \bmod 2) + 1 \end{cases}$$

The following proposition simply states that symmetric processes have symmetric behaviors, i.e. they perform symmetric f-steps and then evolve into processes which are again symmetric.

Proposition 5.17 Let $S \approx S'$ and $v \in (\mathbb{A}_d \cup \{\tau\})^*$. Then: $S' \dashv\rightarrow R' \xrightarrow{v} S'_0 \in \tilde{\mathbb{P}}_{\ell_1}$ implies $S \dashv\rightarrow R \xrightarrow{\mathcal{S}(v)} S_0 \in \tilde{\mathbb{P}}_{\ell_1}$ with $S_0 \approx S'_0$, and one is an f-step if and only if the other one is.

³There are 192 non-initial processes reached during the execution of f-steps from any $D_y \in \mathbb{D}$; they are listed in the appendix.

$D_1 = ((b_1 wt.P_{11} \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(false, false, 1)])[\Phi_B]$
$D_2 = ((b_1 wt.P_{11} \parallel P_2) \parallel_B [\text{PV}(false, false, 1)])[\Phi_B]$
$D_3 = ((P_1 \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(false, false, 1)])[\Phi_B]$
$D_4 = ((P_{11} \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B]$
$D_5 = ((P_{11} \parallel P_2) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B]$
$D_6 = ((b_1 wt.P_{11} \parallel P_{21}) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B]$
$D_7 = ((P_1 \parallel P_{21}) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B]$
$D_8 = ((P_{14} \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B]$
$D_9 = ((P_{14} \parallel P_2) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B]$
$D_{10} = ((b_1 wt.P_{11} \parallel P_{24}) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B]$
$D_{11} = ((P_1 \parallel P_{24}) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B]$
$D_{12} = ((kw2.b_1 wf.P_1 \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B]$
$D_{13} = ((kw2.b_1 wf.P_1 \parallel P_2) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B]$
$D_{14} = ((b_1 wt.P_{11} \parallel kw1.b_2 wf.P_2) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B]$
$D_{15} = ((P_1 \parallel kw1.b_2 wf.P_2) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B]$
$D_{16} = ((b_1 wf.P_1 \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B]$
$D_{17} = ((b_1 wf.P_1 \parallel P_2) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B]$
$D_{18} = ((P_{11} \parallel P_{21}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B]$
$D_{19} = ((P_{14} \parallel P_{21}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B]$
$D_{20} = ((P_{11} \parallel P_{24}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B]$
$D_{21} = ((kw2.b_1 wf.P_1 \parallel P_{21}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B]$
$D_{22} = ((P_{11} \parallel kw1.b_2 wf.P_2) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B]$
$D_{23} = ((b_1 wf.P_1 \parallel P_{21}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B]$

Table 1: Fair-Reachable Processes

Let $D_0 = Dekker[\text{PV}]$, $\mathbb{D} = \{D_0, \dots, D_{47}\}$ and $\mathbb{S} = \{S_0, \dots, S_{47}\}$. The following proposition states the main result of this section, i.e. all processes which are fair-reachable from $Dekker[\text{PV}]$ belong to the set $\mathbb{D} \cup \mathbb{S}$.

Proposition 5.18 Let $S \in \mathbb{D} \cup \mathbb{S}$ and $v \in (\mathbb{A}_d \cup \{\tau\})^*$. $S \xrightarrow{1} R \xrightarrow{v} S' \in \tilde{\mathbb{P}}_{\ell_1}$ implies $S' \in \mathbb{D} \cup \mathbb{S}$.

Proof: The proof of the case in which $S \in \mathbb{D}$ can be found in the appendix (see Lemma A.1). If $S \in \mathbb{S}$ the statement comes directly from Proposition 5.17. \square

5.4 Progressing Processes

We can distinguish terms in $\mathbb{D} \cup \mathbb{S}$ depending on how many processes are waiting to perform their critical section or, in other words, depending on how many actions cs_i are still pending. We say that the action cs_i is *pending* for a given $S \in \mathbb{D} \cup \mathbb{S}$ if there exist sequences of basic actions $v, w \in (\mathbb{A}_d \cup \{\tau\})^*$ such that $Dekker[\text{PV}] \xrightarrow{v \text{ req}_i, w} S$ and $\text{cs}_i \notin w$. Each process may have at most two pending actions and so we are interested in the set \mathbb{R}_1 (\mathbb{R}_2) of fair-reachable states with only cs_1 (cs_2 , respectively) pending and the set $\mathbb{R}_{1,2}$ of fair-reachable states with both cs_1 and cs_2 pending.

We may check if a given fair-reachable process S belongs to \mathbb{R}_1 , \mathbb{R}_2 or $\mathbb{R}_{1,2}$ by considering its syntactical structure and, in particular, the local states of P_1 and P_2 in S . As an example, consider $D_{14} = ((b_1 wt.P_{11} \parallel kw1.b_2 wf.P_2) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B]$: P_1 has requested to perform its critical section but the corresponding action cs_1 has not been performed yet; P_2 is exiting its critical section and both actions req_2 and cs_2 have already been executed. Thus, $D_{14} \in \mathbb{R}_1$. Similarly, we have that $D_{12} = ((kw2.b_1 wf.P_1 \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] \in \mathbb{R}_2$ while

$D_{24} = ((P_{12} \parallel P_{21}) \parallel_B [\text{PV}(\text{true}, \text{true}, 1)])[\Phi_B]$
$D_{25} = ((P_{12} \parallel P_{22}) \parallel_B [\text{PV}(\text{true}, \text{true}, 1)])[\Phi_B]$
$D_{26} = ((P_{12} \parallel b_2wf.P_{23}) \parallel_B [\text{PV}(\text{true}, \text{true}, 1)])[\Phi_B]$
$D_{27} = ((P_{14} \parallel P_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B]$
$D_{28} = ((P_{11} \parallel P_{22}) \parallel_B [\text{PV}(\text{true}, \text{true}, 1)])[\Phi_B]$
$D_{29} = ((kw2.b_1wf.P_1 \parallel P_{22}) \parallel_B [\text{PV}(\text{true}, \text{true}, 1)])[\Phi_B]$
$D_{30} = ((kw2.b_1wf.P_1 \parallel b_2wf.P_{23}) \parallel_B [\text{PV}(\text{true}, \text{true}, 1)])[\Phi_B]$
$D_{31} = ((kw2.b_1wf.P_1 \parallel P_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B]$
$D_{32} = ((P_{12} \parallel kw1.b_2wf.P_1) \parallel_B [\text{PV}(\text{true}, \text{true}, 1)])[\Phi_B]$
$D_{33} = ((b_1wf.P_1 \parallel P_{22}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B]$
$D_{34} = ((b_1wf.P_1 \parallel b_2wf.P_{23}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B]$
$D_{35} = ((b_1wf.P_1 \parallel P_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])[\Phi_B]$
$D_{36} = ((P_1 \parallel P_{22}) \parallel_B [\text{PV}(\text{false}, \text{true}, 2)])[\Phi_B]$
$D_{37} = ((P_{11} \parallel b_2wf.P_{23}) \parallel_B [\text{PV}(\text{true}, \text{true}, 1)])[\Phi_B]$
$D_{38} = ((P_{11} \parallel P_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B]$
$D_{39} = ((P_{12} \parallel P_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B]$
$D_{40} = ((P_1 \parallel P_{23}) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])[\Phi_B]$
$D_{41} = ((b_1wt.P_{11} \parallel P_{23}) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])[\Phi_B]$
$D_{42} = ((P_{11} \parallel P_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])[\Phi_B]$
$D_{43} = ((P_{12} \parallel P_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])[\Phi_B]$
$D_{44} = ((b_1wf.P_{13} \parallel P_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])[\Phi_B]$
$D_{45} = ((P_{13} \parallel P_{23}) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])[\Phi_B]$
$D_{46} = (b_1wf.P_{13} \parallel b_2wt.P_{21}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)][\Phi_B]$
$D_{47} = (P_{13} \parallel b_2wt.P_{21}) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)][\Phi_B]$

Table 2: Fair-Reachable Processes

$D_1 = ((b_1wt.P_{11} \parallel b_2wt.P_{21}) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])[\Phi_B] \in \mathbf{R}_{1,2}$. In detail, we have:

$$\mathbf{D}_1 = \mathbf{D} \cap \mathbf{R}_1 = \{D_2, D_5, D_9, D_{14}, D_{22}, D_{32}\}$$

$$\mathbf{D}_2 = \mathbf{D} \cap \mathbf{R}_2 = \{D_3, D_7, D_{11}, D_{12}, D_{16}, D_{21}, D_{23}, D_{29}, D_{30}, D_{31}, D_{33}, \dots, D_{36}, D_{40}\}$$

$$\mathbf{D}_{1,2} = \mathbf{D} \cap \mathbf{R}_{1,2} = \{D_1, D_4, D_6, D_8, D_{10}, D_{18}, D_{19}, D_{20}, D_{24}, \dots, D_{28}, D_{37}, D_{38}, D_{39}, D_{41}, \dots, D_{47}\}$$

Processes D_0, D_{13}, D_{15} and D_{17} have no pending sections. Furthermore, since $S \in \mathbf{R}_1$, $S \in \mathbf{R}_2$ and $S \in \mathbf{R}_{1,2}$ imply $\mathcal{S}(S) \in \mathbf{R}_2$, $\mathcal{S}(S) \in \mathbf{R}_1$ and $\mathcal{S}(S) \in \mathbf{R}_{1,2}$, respectively, we also have: $\mathbf{S}_1 = \mathbf{S} \cap \mathbf{R}_1 = \mathcal{S}(\mathbf{D}_2)$, $\mathbf{S}_2 = \mathbf{S} \cap \mathbf{R}_2 = \mathcal{S}(\mathbf{D}_1)$ and $\mathbf{S}_{1,2} = \mathbf{S} \cap \mathbf{R}_{1,2} = \mathcal{S}(\mathbf{D}_{1,2})$. S_0, S_{13}, S_{15} and S_{17} have no pending actions.

Definition 5.19 (*progressing processes*)

Let $S \in \mathbf{D} \cup \mathbf{S}$ and $v = \alpha_1 \dots \alpha_n \in (\mathbb{A}_d \cup \{\tau\})^*$:

- We say that the string v contains the action cs_i ($i \in \{1, 2\}$), and write that $\text{cs}_i \in v$ if $\alpha_j = \text{cs}_i$ for some $j \in [1, n]$, and $\text{cs}_1, \text{cs}_2 \in v$ if both $\text{cs}_1 \in v$ and $\text{cs}_2 \in v$.
- We say that S implies the execution of the action cs_i , written $S \triangleright \text{cs}_i$, if each f-execution from S contains the action cs_i . We write $S \triangleright \text{cs}_1, \text{cs}_2$ to denote that both $S \triangleright \text{cs}_1$ and $S \triangleright \text{cs}_2$. Finally, given $A \subseteq \mathbf{D} \cup \mathbf{S}$ we write $A \triangleright \text{cs}_i$ ($A \triangleright \text{cs}_1, \text{cs}_2$) to denote that $S \triangleright \text{cs}_i$ ($S \triangleright \text{cs}_1, \text{cs}_2$, resp.) for each $S \in A$.
- We say that $S \in \mathbf{R}_1$ (symmetrically for $S \in \mathbf{R}_2$ and $S \in \mathbf{R}_{1,2}$) is *making progress* (*progressing*) if $S \triangleright \text{cs}_1$ ($S \triangleright \text{cs}_2$, $S \triangleright \text{cs}_1, \text{cs}_2$, respectively).

Proposition 5.20 Let $S, S' \in \mathbf{D} \cup \mathbf{S}$ with $S \approx S'$. Then:

1. $S \triangleright cs_i$ implies $S' \triangleright cs_j$ with $\{i, j\} = \{1, 2\}$;
2. if S is making progress, then also S' is making progress.

Proof: Item 1 follows from iterative application of Proposition 5.17 and from the fact that $cs_i \in v$ implies $cs_j \in \mathcal{S}(v)$ for $\{i, j\} = \{1, 2\}$. Moreover, since $S \in R_i$ ($S \in R_{1,2}$) implies $S' \in R_j$ with $\{i, j\} = \{1, 2\}$ ($S' \in R_{1,2}$ respectively), Item 2 follows directly from Item 1. \square

We now sketch how to prove that all processes in $D \cup S$ are making progress; more details can be found in Appendix B. In principle, we construct for each process $D \in D$ the transition system of those processes and transitions encountered in f-steps from D ; then, we read off from this transition system that the actions cs_i pending for D are indeed performed in every f-execution from D . The basis for the latter is the following simple observation: if for each f-step from some D_y of the form $D_y \xrightarrow{1} R \xrightarrow{v} S \in \tilde{\mathbb{P}}_{\ell 1}$ we have either $cs_i \in v$ or $S \triangleright cs_i$, then we have $D_y \triangleright cs_i$.

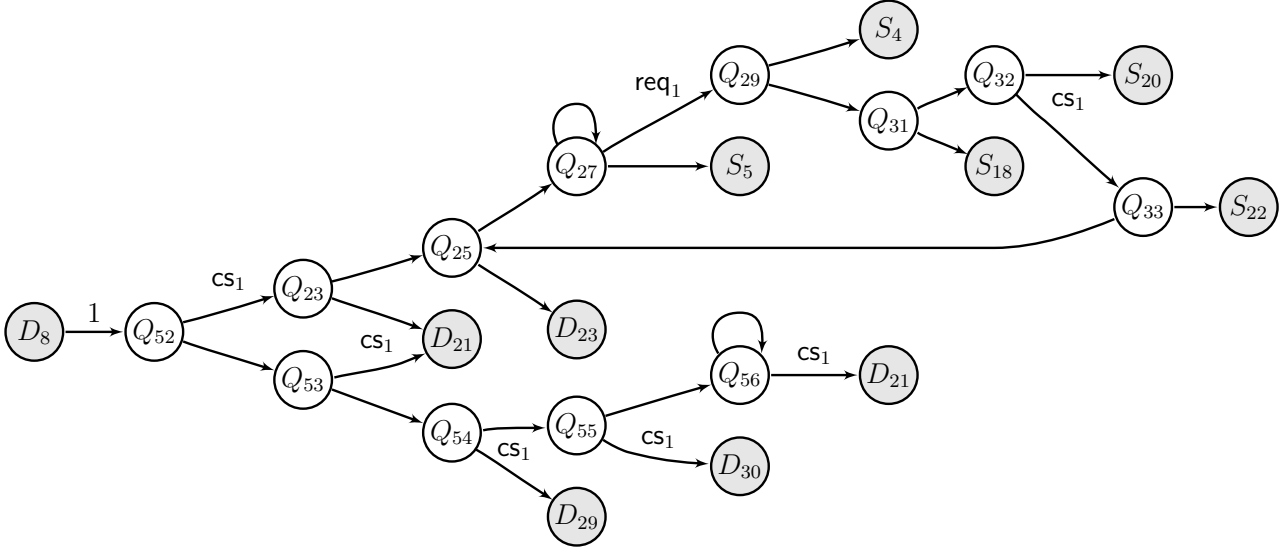


Figure 1: Fair-Steps of D_8

Figure 1 shows the respective transition system for D_8 ; to increase readability, we omit τ -actions in the transition labels. Applying the above observation, we see that $cs_1 \in v$ for each suitable v , and thus we have $D_8 \triangleright cs_1$. Note that, if only cs_1 were pending for D_8 , we would not have to follow the transitions from Q_{23} since cs_1 was already performed when Q_{23} is reached; thus, we would not generate Q_{25} and all subsequent processes. Below, we will indicate states where we stopped building the transition system by an arrow.

Collecting statements like $D_8 \triangleright cs_1$, we can also deal with more complicated cases. As an example, assume that $D_8, D_9, D_{24}, D_{25}, D_{26}$ and D_{27} have already been proven to imply the execution of the action cs_1 ; then we can prove that $D_5 = ((P_{11} \parallel P_2) \parallel_B [PV(true, false, 1)])[\Phi_B]$ is making progress; cf. the respective transition system for D_5 in Figure 2.

Starting from D_5 , an f-step either runs through Q_9 and is thus clearly not relevant, or it leads to $D_8, D_9, D_{24}, D_{25}, D_{26}$ or to D_{27} . Since $D_8, D_9, D_{24}, D_{25}, D_{26}$ and D_{27} imply the execution of the action cs_1 , we can conclude that D_5 implies the execution of cs_1 and, since $D_5 \in R_1$, that this fair-reachable process is making progress.

Proposition 5.21 All processes in $D \cup S$ are making progress.

From this proposition, we can conclude:

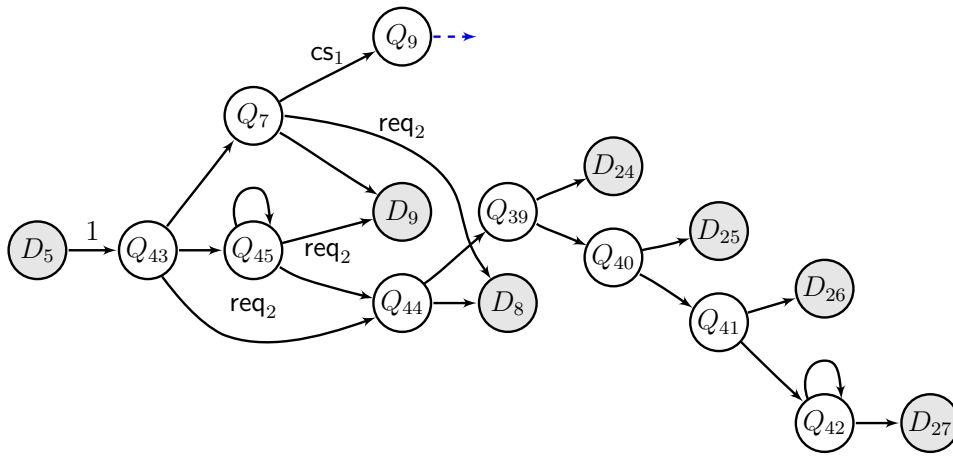


Figure 2: Fair-Steps of D_8

Proposition 5.22 Each f-execution from $Dekker[PV]$ is live.

Proof: Assume, by contradiction, that there exists an f-execution from $Dekker[PV]$

$$Dekker[PV] = P_0 \xrightarrow{1} Q_0 \xrightarrow{v_0} P_1 \xrightarrow{1} Q_1 \xrightarrow{v_1} P_2 \dots$$

which is not live. By Definition 3.3, there exists $j \in \mathbb{N}_0$ such that

$$P_0 \xrightarrow{1} Q_0 \xrightarrow{v_0} P_1 \xrightarrow{1} Q_1 \xrightarrow{v_1} P_2 \dots \xrightarrow{v_{j-1}} P_j \xrightarrow{1} Q_j \xrightarrow{w_1} Q' \xrightarrow{\text{req}_i} Q' \xrightarrow{w_2} P_{j+1} \xrightarrow{1} Q_{j+1} \xrightarrow{v_{j+1}} P_{j+2} \dots$$

where $v_j = w_1 \text{req}_i w_2$ and cs_i does not occur in $w_2 v_{j+1} v_{j+2} \dots$. Thus, cs_i is pending for P_{j+1} and it does not occur in the f-execution $P_{j+1} \xrightarrow{1} Q_{j+1} \xrightarrow{v_{j+1}} P_{j+2} \dots$. This implies that $P_{j+1} \in \mathcal{D} \cup \mathcal{S}$ is not making progress, contradicting Lemma 5.21. \square

As an immediate consequence of the relationships between fair traces of $Dekker$ and f-executions of $Dekker[PV]$, we can state the main result of this section:

Theorem 5.23 Each fair trace of $Dekker$ satisfies the liveness property.

References

- [1] S. Brookes. Traces, Pomsets, Fairness and Full Abstractions for Communicating Processes. Proc. of *Concur'02*, Lect. Notes Comp. Sci. 2421, pp. 466-482, Springer, 2002.
- [2] R. Cleaveland, J. Parrow, and B. Steffen. The concurrency workbench: a semantics-based tool for the verification of concurrent systems. In *Proceedings of ACM Transaction on Programming Languages and Systems*, **15**, 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 Components in System Computations. *Theoretical Computer Science*, **356**(3), pp. 291-324, 2006.
- [5] F. Corradini, M.R. Di Berardini, and W. Vogler. Fairness of Actions in System Computations. *Acta Informatica*, **43**, pp. 73-130, 2006.

- [6] F. Corradini, W. Vogler, and L. Jenner. Comparing the Worst-Case Efficiency of Asynchronous Systems with PAFAS. *Acta Informatica* **38**, pp. 735-792, 2002.
- [7] G. Costa, C. Stirling. A Fair Calculus of Communicating Systems. *Acta Informatica* **21**, pp. 417-441, 1984.
- [8] G. Costa, C. Stirling. Weak and Strong Fairness in CCS. *Information and Computation* **73**, pp. 207-244, 1987.
- [9] E.A. Emmerson. Temporal and Modal Logic. *Handbook of Theoretical Computer Science* Vol. B, 1990.
- [10] N. Francez. *Fairness*. Springer, 1986.
- [11] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.
- [12] G. Lüttgen and W. Vogler. Bisimulation on speed: Worst-case efficiency. *Information and Computation*, 191(2):105-144, 2004.
- [13] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.
- [14] R. Milner. *Communication and Concurrency*. International series in computer science, Prentice Hall International, 1989.
- [15] J.L. Peterson, A. Silberschatz . *Operating Systems Concepts*. Addison Wiley, 1985.
- [16] W. Vogler. Efficiency of asynchronous systems, read arcs, and the MUTEX-problem. *Theoret. Comput. Sci.* 275, pp. 589-631, 2002.
- [17] D.J. Walker. Automated Analysis of Mutual Exclusion algorithms using CCS. *Formal Aspects of Computing* **1**, pp. 273-292, 1989.

A List of Fair-Reachable Processes

This appendix section is devoted to identify all processes which belong to \mathcal{D} . We proceed as follows. First of all, we include in \mathcal{D} all processes that are reachable from $Dekker[PV]$ by performing one f-step. Those processes are ranged over by D_0, \dots, D_{13} and hence, at first, $\mathcal{D} = \{D_0, \dots, D_{13}\}$. Then, for any given $D_y \in \mathcal{D}$, we consider all processes that can be reached from D_y by executing another f-step. Any of such P is added to \mathcal{D} if and only if neither P nor its symmetric $\mathcal{S}(P)$ are already in \mathcal{D} and, in this case, we set $S = D_{y'}$, where y' is the number of element currently in \mathcal{D} .

We will recursively apply this procedure until it is not possible to obtain new processes any more. At the end, we have all processes we are interested in. Moreover, as a trivial consequence, the following statement (needed in the proof of Proposition 5.18) follows “by construction”:

Lemma A.1 Let $S \in \mathcal{D}$ and let $S \xrightarrow{1} R \xrightarrow{v} S' \in \tilde{\mathbb{P}}_{\ell_1}$ be an f-step from S . Then: $S' \in \mathcal{D}$.

All non initial processes reached along f-execution sequences from $Dekker[PV]$ are ranged over by Q_0, \dots, Q_{191} . Their formal definition is again provided in the next pages. To this aim, we need to define how processes P_1 and P_2 and their derivatives evolve by letting one time unit pass. Recall that, we assume the component-oriented timed operational semantics defined in Section 2.2; thus, as stated by Definition 2.15, we simply mark the enabled parallel components as urgent.

Definition A.2 (*urgent processes*) The urgent versions of processes P_1 and P_2 according to Definition 2.15 are defined as follows:

$$\begin{array}{ll}
 \underline{P}_1 = b_1 wt.req_1.P_{11} \underline{\pm} \tau.P_1 & \underline{P}_2 = b_2 wt.req_2.P_{21} \underline{\pm} \tau.P_2 \\
 \underline{P}_{11} = b_2 rf.P_{14} \underline{\pm} b_2 rt.P_{12} & \underline{P}_{21} = b_1 rf.P_{24} \underline{\pm} b_1 rt.P_{22} \\
 \underline{P}_{12} = kr1.P_{11} \underline{\pm} kr2.b_1 wf.P_{13} & \underline{P}_{22} = kr2.P_{21} \underline{\pm} kr1.b_2 wf.P_{23} \\
 \underline{P}_{13} = kr1.b_1 wt.P_{11} \underline{\pm} kr2.P_{13} & \underline{P}_{23} = kr2.b_2 wt.P_{21} \underline{\pm} kr1.P_{23} \\
 \underline{P}_{14} = \underline{cs}_1.kw2.b_1 wf.P_1 & \underline{P}_{24} = \underline{cs}_2.kw1.b_2 wf.P_2
 \end{array}$$

$$\text{Dekker[PV]} \xrightarrow{1} ((\underline{P}_1 \parallel \underline{P}_2) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])[\Phi_B] = Q_0$$

$$Q_0 \xrightarrow{\text{req}_1} ((b_1 \text{wt}.P_{11} \parallel \underline{P}_2) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])[\Phi_B] = Q_1$$

$$Q_0 \xrightarrow{\tau} ((P_1 \parallel \underline{P}_2) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])[\Phi_B] = Q_2$$

$$Q_0 \xrightarrow{\text{req}_2} ((\underline{P}_1 \parallel b_2 \text{wt}.P_{21}) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])[\Phi_B] = Q_3$$

$$Q_0 \xrightarrow{\tau} ((\underline{P}_1 \parallel P_2) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])[\Phi_B] = Q_4$$

$$Q_1 \xrightarrow{\tau} ((P_{11} \parallel \underline{P}_2) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = Q_5$$

$$Q_1 \xrightarrow{\text{req}_2} ((b_1 \text{wt}.P_{11} \parallel b_2 \text{wt}.P_{21}) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])[\Phi_B] = D_1$$

$$Q_1 \xrightarrow{\tau} ((b_1 \text{wt}.P_{11} \parallel P_2) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])[\Phi_B] = D_2$$

$$Q_2 \xrightarrow{\text{req}_1} ((b_1 \text{wt}.P_{11} \parallel \underline{P}_2) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])[\Phi_B] = Q_1$$

$$Q_2 \xrightarrow{\tau} ((P_1 \parallel \underline{P}_2) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])[\Phi_B] = Q_2$$

$$Q_2 \xrightarrow{\text{req}_2} ((P_1 \parallel b_1 \text{wt}.P_{21}) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])[\Phi_B] = D_3$$

$$Q_2 \xrightarrow{\tau} ((P_1 \parallel P_2) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])[\Phi_B] = \text{Dekker}$$

$$Q_3 \xrightarrow{\text{req}_1} ((b_1 \text{wt}.P_{11} \parallel b_2 \text{wt}.P_{21}) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])[\Phi_B] = D_1$$

$$Q_3 \xrightarrow{\tau} ((P_1 \parallel b_2 \text{wt}.P_{21}) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])[\Phi_B] = D_3$$

$$Q_3 \xrightarrow{\tau} ((\underline{P}_1 \parallel P_{21}) \parallel_B [\text{PV}(\text{false}, \text{true}, 1)])[\Phi_B] = Q_6$$

$$Q_4 \xrightarrow{\text{req}_1} ((b_1 \text{wt}.P_{11} \parallel P_2) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])[\Phi_B] = D_2$$

$$Q_4 \xrightarrow{\tau} ((P_1 \parallel P_2) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])[\Phi_B] = \text{Dekker}$$

$$Q_4 \xrightarrow{\text{req}_2} ((\underline{P}_1 \parallel b_2 \text{wt}.P_{21}) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])[\Phi_B] = Q_3$$

$$Q_4 \xrightarrow{\tau} ((\underline{P}_1 \parallel P_2) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])[\Phi_B] = Q_4$$

$$Q_5 \xrightarrow{\tau} ((P_{14} \parallel \underline{P}_2) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = Q_7$$

$$Q_5 \xrightarrow{\text{req}_2} ((P_{11} \parallel b_2 \text{wt}.P_{21}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = D_4$$

$$Q_5 \xrightarrow{\tau} ((P_{11} \parallel P_2) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = D_5$$

$$Q_6 \xrightarrow{\text{req}_1} ((b_1 \text{wt}.P_{11} \parallel P_{21}) \parallel_B [\text{PV}(\text{false}, \text{true}, 1)])[\Phi_B] = D_6$$

$$Q_6 \xrightarrow{\tau} ((P_1 \parallel P_{21}) \parallel_B [\text{PV}(\text{false}, \text{true}, 1)])[\Phi_B] = D_7$$

$$Q_6 \xrightarrow{\tau} ((\underline{P}_1 \parallel P_{24}) \parallel_B [\text{PV}(\text{false}, \text{true}, 1)])[\Phi_B] = Q_8$$

$$Q_7 \xrightarrow{\text{cs}_1} ((kw2.b_1 \text{wf}.P_1 \parallel \underline{P}_2) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = Q_9$$

$$Q_7 \xrightarrow{\text{req}_2} ((P_{14} \parallel b_2 \text{wt}.P_{21}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = D_8$$

$$Q_7 \xrightarrow{\tau} ((P_{14} \parallel P_2) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = D_9$$

$$Q_8 \xrightarrow{\text{req}_1} ((b_1 \text{wt}.P_{11} \parallel P_{24}) \parallel_B [\text{PV}(\text{false}, \text{true}, 1)])[\Phi_B] = D_{10}$$

$$Q_8 \xrightarrow{\tau} ((P_1 \parallel P_{24}) \parallel_B [\text{PV}(\text{false}, \text{true}, 1)])[\Phi_B] = D_{11}$$

$$Q_8 \xrightarrow{\text{cs}_2} ((\underline{P}_1 \parallel kw1.b_2 \text{wf}.P_2) \parallel_B [\text{PV}(\text{false}, \text{true}, 1)])[\Phi_B] = Q_{10}$$

$$Q_9 \xrightarrow{\tau} ((b_1 \text{wf}.P_1 \parallel \underline{P}_2) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])[\Phi_B] = Q_{11}$$

$$Q_9 \xrightarrow{\text{req}_2} ((kw2.b_1 \text{wf}.P_1 \parallel b_2 \text{wt}.P_{21}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = D_{12}$$

$$Q_9 \xrightarrow{\tau} ((kw2.b_1 \text{wf}.P_1 \parallel P_2) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = D_{13}$$

$$\begin{aligned}
Q_{10} &\xrightarrow{\text{req}_1} ((b_1 \text{ wt}.P_{11} \parallel kw1.b_2 \text{ wf}.P_2) \parallel_B [\text{PV}(\text{false}, \text{true}, 1)])(\Phi_B) = D_{14} \\
Q_{10} &\xrightarrow{\tau} ((P_1 \parallel kw1.b_2 \text{ wf}.P_2) \parallel_B [\text{PV}(\text{false}, \text{true}, 1)])(\Phi_B) = D_{15} \\
Q_{10} &\xrightarrow{\tau} ((\underline{P}_1 \parallel b_2 \text{ wf}.P_2) \parallel_B [\text{PV}(\text{false}, \text{true}, 1)])(\Phi_B) = Q_{12} \\
\\
Q_{11} &\xrightarrow{\tau} ((P_1 \parallel \underline{P}_2) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])(\Phi_B) = Q_{13} \\
Q_{11} &\xrightarrow{\text{req}_2} ((b_1 \text{ wf}.P_1 \parallel b_2 \text{ wt}.P_{21}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])(\Phi_B) = D_{16} \\
Q_{11} &\xrightarrow{\tau} ((b_1 \text{ wf}.P_1 \parallel P_2) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])(\Phi_B) = D_{17} \\
\\
Q_{12} &\xrightarrow{\tau} ((b_1 \text{ wt}.P_{11} \parallel b_2 \text{ wf}.P_2) \parallel_B [\text{PV}(\text{false}, \text{true}, 1)])(\Phi_B) = S_{16} \\
Q_{12} &\xrightarrow{\tau} ((P_1 \parallel b_2 \text{ wf}.P_2) \parallel_B [\text{PV}(\text{false}, \text{true}, 1)])(\Phi_B) = S_{17} \\
Q_{12} &\xrightarrow{\tau} ((\underline{P}_1 \parallel P_2) \parallel_B [\text{PV}(\text{false}, \text{false}, 1)])(\Phi_B) = Q_4 \\
\\
Q_{13} &\xrightarrow{\text{req}_1} ((b_1 \text{ wt}.P_{11} \parallel \underline{P}_2) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])(\Phi_B) = Q_{14} \\
Q_{13} &\xrightarrow{\tau} ((P_1 \parallel \underline{P}_2) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])(\Phi_B) = Q_{13} \\
Q_{13} &\xrightarrow{\text{req}_2} ((P_1 \parallel b_2 \text{ wt}.P_{21}) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])(\Phi_B) = S_2 \\
Q_{13} &\xrightarrow{\tau} ((P_1 \parallel P_2) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])(\Phi_B) = \mathcal{S}(\text{Dekker}) \\
\\
Q_{14} &\xrightarrow{\tau} ((P_{11} \parallel \underline{P}_2) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])(\Phi_B) = Q_{15} \\
Q_{14} &\xrightarrow{\text{req}_2} ((b_1 \text{ wt}.P_{11} \parallel b_2 \text{ wt}.P_{21}) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])(\Phi_B) = S_1 \\
Q_{14} &\xrightarrow{\tau} ((b_1 \text{ wt}.P_{11} \parallel P_2) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])(\Phi_B) = S_3 \\
\\
Q_{15} &\xrightarrow{\tau} ((P_{14} \parallel \underline{P}_2) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])(\Phi_B) = Q_{16} \\
Q_{15} &\xrightarrow{\text{req}_2} ((P_{11} \parallel b_2 \text{ wt}.P_{21}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])(\Phi_B) = S_6 \\
Q_{15} &\xrightarrow{\tau} ((P_{11} \parallel P_2) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])(\Phi_B) = S_7 \\
\\
Q_{16} &\xrightarrow{\text{cs}_1} ((kw2.b_1 \text{ wf}.P_1 \parallel \underline{P}_2) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])(\Phi_B) = Q_{17} \\
Q_{16} &\xrightarrow{\text{req}_2} ((P_{14} \parallel b_2 \text{ wt}.P_{21}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])(\Phi_B) = S_{10} \\
Q_{16} &\xrightarrow{\tau} ((P_{14} \parallel P_2) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])(\Phi_B) = S_{11} \\
\\
Q_{17} &\xrightarrow{\tau} ((b_1 \text{ wf}.P_1 \parallel \underline{P}_2) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])(\Phi_B) = Q_{11} \\
Q_{17} &\xrightarrow{\text{req}_2} ((kw2.b_1 \text{ wf}.P_1 \parallel b_2 \text{ wt}.P_{21}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])(\Phi_B) = S_{14} \\
Q_{17} &\xrightarrow{\tau} ((kw2.b_1 \text{ wf}.P_1 \parallel P_2) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])(\Phi_B) = S_{15}
\end{aligned}$$

$$\mathbf{D}_1 \xrightarrow{1} ((\underline{b_1 wt. P_{11}} \parallel \underline{b_2 wt. P_{21}}) \parallel_B [\mathbf{PV}(false, false, 1)])[\Phi_B] = \mathbf{Q}_{18}$$

$$Q_{18} \xrightarrow{\tau} ((P_{11} \parallel \underline{b_2 wt. P_{21}}) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = Q_{19}$$

$$Q_{18} \xrightarrow{\tau} ((\underline{b_1 wt. P_{11}} \parallel P_{21}) \parallel_B [\mathbf{PV}(false, true, 1)])[\Phi_B] = Q_{20}$$

$$Q_{19} \xrightarrow{\tau} ((P_{14} \parallel \underline{b_2 wt. P_{21}}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = Q_{21}$$

$$Q_{19} \xrightarrow{\tau} ((P_{11} \parallel P_{21}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = D_{18}$$

$$Q_{20} \xrightarrow{\tau} ((P_{11} \parallel P_{21}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = D_{18}$$

$$Q_{20} \xrightarrow{\tau} ((\underline{b_1 wt. P_{11}} \parallel P_{24}) \parallel_B [\mathbf{PV}(false, true, 1)])[\Phi_B] = Q_{22}$$

$$Q_{21} \xrightarrow{cs_1} ((kw2.b_1 wf.P_1 \parallel \underline{b_2 wt. P_{21}}) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = Q_{23}$$

$$Q_{21} \xrightarrow{\tau} ((P_{14} \parallel P_{21}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = D_{19}$$

$$Q_{22} \xrightarrow{\tau} ((P_{11} \parallel P_{24}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = D_{20}$$

$$Q_{22} \xrightarrow{cs_2} ((\underline{b_1 wt. P_{11}} \parallel kw1.b_2 wf.P_2) \parallel_B [\mathbf{PV}(false, true, 1)])[\Phi_B] = Q_{24}$$

$$Q_{23} \xrightarrow{\tau} ((b_1 wf.P_1 \parallel \underline{b_2 wt. P_{21}}) \parallel_B [\mathbf{PV}(true, false, 2)])[\Phi_B] = Q_{25}$$

$$Q_{23} \xrightarrow{\tau} ((kw2.b_1 wf.P_1 \parallel P_{21}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = D_{21}$$

$$Q_{24} \xrightarrow{\tau} ((P_{11} \parallel kw1.b_2 wf.P_2) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = D_{22}$$

$$Q_{24} \xrightarrow{\tau} ((\underline{b_1 wt. P_{11}} \parallel b_2 wf.P_2) \parallel_B [\mathbf{PV}(false, true, 1)])[\Phi_B] = Q_{26}$$

$$Q_{25} \xrightarrow{\tau} ((P_1 \parallel \underline{b_2 wt. P_{21}}) \parallel_B [\mathbf{PV}(false, false, 2)])[\Phi_B] = Q_{27}$$

$$Q_{25} \xrightarrow{\tau} ((b_1 wf.P_1 \parallel P_{21}) \parallel_B [\mathbf{PV}(true, true, 2)])[\Phi_B] = D_{23}$$

$$Q_{26} \xrightarrow{\tau} ((P_{11} \parallel b_2 wf.P_2) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = S_{23}$$

$$Q_{26} \xrightarrow{\tau} ((\underline{b_1 wt. P_{11}} \parallel P_2) \parallel_B [\mathbf{PV}(false, false, 1)])[\Phi_B] = Q_{28}$$

$$Q_{27} \xrightarrow{req_1} ((b_1 wt.P_{11} \parallel \underline{b_2 wt. P_{21}}) \parallel_B [\mathbf{PV}(false, false, 2)])[\Phi_B] = Q_{29}$$

$$Q_{27} \xrightarrow{\tau} ((P_1 \parallel \underline{b_2 wt. P_{21}}) \parallel_B [\mathbf{PV}(false, false, 2)])[\Phi_B] = Q_{27}$$

$$Q_{27} \xrightarrow{\tau} ((P_1 \parallel P_{21}) \parallel_B [\mathbf{PV}(false, true, 2)])[\Phi_B] = S_5$$

$$Q_{28} \xrightarrow{\tau} ((P_{11} \parallel P_2) \parallel_B [\mathbf{PV}(false, false, 1)])[\Phi_B] = D_5$$

$$Q_{28} \xrightarrow{req_2} ((\underline{b_1 wt. P_{11}} \parallel \underline{b_2 wt. P_{21}}) \parallel_B [\mathbf{PV}(false, false, 1)])[\Phi_B] = Q_{30}$$

$$Q_{28} \xrightarrow{\tau} ((\underline{b_1 wt. P_{11}} \parallel P_2) \parallel_B [\mathbf{PV}(false, false, 1)])[\Phi_B] = Q_{28}$$

$$Q_{29} \xrightarrow{\tau} ((P_{11} \parallel \underline{b_2 wt. P_{21}}) \parallel_B [\mathbf{PV}(true, false, 2)])[\Phi_B] = Q_{31}$$

$$Q_{29} \xrightarrow{\tau} ((b_1 wt.P_{11} \parallel P_{21}) \parallel_B [\mathbf{PV}(false, true, 2)])[\Phi_B] = S_4$$

$$Q_{30} \xrightarrow{\tau} ((P_{11} \parallel \underline{b_2 wt. P_{21}}) \parallel_B [\mathbf{PV}(false, false, 1)])[\Phi_B] = D_4$$

$$Q_{30} \xrightarrow{\tau} ((\underline{b_1 wt. P_{11}} \parallel P_{21}) \parallel_B [\mathbf{PV}(false, true, 1)])[\Phi_B] = Q_{20}$$

$$Q_{31} \xrightarrow{\tau} ((P_{14} \parallel \underline{b_2 wt. P_{21}}) \parallel_B [\mathbf{PV}(true, false, 2)])[\Phi_B] = Q_{32}$$

$$Q_{31} \xrightarrow{\tau} ((P_{11} \parallel P_{21}) \parallel_B [\mathbf{PV}(true, true, 2)])[\Phi_B] = S_{18}$$

$$Q_{32} \xrightarrow{cs_1} ((kw2.b_1wf.P_1 \parallel \underline{b_2wt}.P_{21}) \parallel_B [PV(true, false, 2)])[\Phi_B] = Q_{33}$$

$$Q_{32} \xrightarrow{\tau} ((P_{14} \parallel P_{21}) \parallel_B [PV(true, true, 2)])[\Phi_B] = S_{20}$$

$$Q_{33} \xrightarrow{\tau} ((b_1wf.P_1 \parallel \underline{b_2wt}.P_{21}) \parallel_B [PV(true, false, 2)])[\Phi_B] = Q_{25}$$

$$Q_{33} \xrightarrow{\tau} ((kw2.b_1wf.P_1 \parallel P_{21}) \parallel_B [PV(true, true, 2)])[\Phi_B] = S_{22}$$

$$D_2 \xrightarrow{1} ((\underline{b_1wt}.P_{11} \parallel \underline{P_2}) \parallel_B [PV(false, false, 1)])[\Phi_B] = Q_{34}$$

$$Q_{34} \xrightarrow{\tau} ((P_{11} \parallel \underline{P_2}) \parallel_B [PV(false, false, 1)])[\Phi_B] = Q_5$$

$$Q_{34} \xrightarrow{req_2} ((\underline{b_1wt}.P_{11} \parallel b_2wt.P_2) \parallel_B [PV(false, false, 1)])[\Phi_B] = Q_{30}$$

$$Q_{34} \xrightarrow{\tau} ((\underline{b_1wt}.P_{11} \parallel P_2) \parallel_B [PV(false, false, 1)])[\Phi_B] = Q_{28}$$

$$D_3 \xrightarrow{1} ((\underline{P_1} \parallel \underline{b_2wt}.P_{21}) \parallel_B [PV(false, false, 1)])[\Phi_B] = Q_{35}$$

$$Q_{35} \xrightarrow{req_1} ((b_1wt.P_{11} \parallel b_2wt.P_{21}) \parallel_B [PV(false, false, 1)])[\Phi_B] = Q_{36}$$

$$Q_{35} \xrightarrow{\tau} ((P_1 \parallel \underline{b_2wt}.P_{21}) \parallel_B [PV(false, false, 1)])[\Phi_B] = Q_{37}$$

$$Q_{35} \xrightarrow{\tau} ((\underline{P_1} \parallel P_{21}) \parallel_B [PV(false, true, 1)])[\Phi_B] = Q_6$$

$$Q_{36} \xrightarrow{\tau} ((P_{11} \parallel \underline{b_2wt}.P_{21}) \parallel_B [PV(true, false, 1)])[\Phi_B] = Q_{19}$$

$$Q_{36} \xrightarrow{\tau} ((b_1wt.P_{11} \parallel P_{21}) \parallel_B [PV(false, true, 1)])[\Phi_B] = D_6$$

$$Q_{37} \xrightarrow{req_1} ((b_1wt.P_{11} \parallel \underline{b_2wt}.P_{21}) \parallel_B [PV(false, false, 1)])[\Phi_B] = Q_{36}$$

$$Q_{37} \xrightarrow{\tau} ((P_1 \parallel \underline{b_2wt}.P_{21}) \parallel_B [PV(false, false, 1)])[\Phi_B] = Q_{37}$$

$$Q_{37} \xrightarrow{\tau} ((P_1 \parallel P_{21}) \parallel_B [PV(false, true, 1)])[\Phi_B] = D_7$$

$$D_4 \xrightarrow{1} ((\underline{P_{11}} \parallel \underline{b_2wt}.P_{21}) \parallel_B [PV(true, false, 1)])[\Phi_B] = Q_{38}$$

$$Q_{38} \xrightarrow{\tau} ((P_{14} \parallel \underline{b_2wt}.P_{21}) \parallel_B [PV(true, false, 1)])[\Phi_B] = Q_{21}$$

$$Q_{38} \xrightarrow{\tau} ((\underline{P_{11}} \parallel P_{21}) \parallel_B [PV(true, true, 1)])[\Phi_B] = Q_{39}$$

$$Q_{39} \xrightarrow{\tau} ((P_{12} \parallel P_{21}) \parallel_B [PV(true, true, 1)])[\Phi_B] = D_{24}$$

$$Q_{39} \xrightarrow{\tau} ((\underline{P_{11}} \parallel P_{22}) \parallel_B [PV(true, true, 1)])[\Phi_B] = Q_{40}$$

$$Q_{40} \xrightarrow{\tau} ((P_{12} \parallel P_{22}) \parallel_B [PV(true, true, 1)])[\Phi_B] = D_{25}$$

$$Q_{40} \xrightarrow{\tau} ((\underline{P_{11}} \parallel b_2wf.P_{23}) \parallel_B [PV(true, true, 1)])[\Phi_B] = Q_{41}$$

$$Q_{41} \xrightarrow{\tau} ((P_{12} \parallel b_2wf.P_{23}) \parallel_B [PV(true, true, 1)])[\Phi_B] = D_{26}$$

$$Q_{41} \xrightarrow{\tau} ((\underline{P_{11}} \parallel P_{23}) \parallel_B [PV(true, false, 1)])[\Phi_B] = Q_{42}$$

$$Q_{42} \xrightarrow{\tau} ((P_{14} \parallel P_{23}) \parallel_B [PV(true, false, 1)])[\Phi_B] = D_{27}$$

$$Q_{42} \xrightarrow{\tau} ((\underline{P_{11}} \parallel P_{23}) \parallel_B [PV(true, false, 1)])[\Phi_B] = Q_{42}$$

$$\mathbf{D}_5 \xrightarrow{\mathbf{1}} ((\underline{P}_{11} \parallel \underline{P}_2) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = \mathbf{Q}_{43}$$

$$Q_{43} \xrightarrow{\tau} ((P_{14} \parallel \underline{P}_2) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = Q_7$$

$$Q_{43} \xrightarrow{\text{req}_2} ((\underline{P}_{11} \parallel b_2wt.P_{21}) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = Q_{44}$$

$$Q_{43} \xrightarrow{\tau} ((\underline{P}_{11} \parallel P_2) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = Q_{45}$$

$$Q_{44} \xrightarrow{\tau} ((P_{14} \parallel b_2wt.P_{21}) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = D_8$$

$$Q_{44} \xrightarrow{\tau} ((\underline{P}_{11} \parallel P_{21}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = Q_{39}$$

$$Q_{45} \xrightarrow{\tau} ((P_{14} \parallel P_2) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = D_9$$

$$Q_{45} \xrightarrow{\text{req}_2} ((\underline{P}_{11} \parallel b_2wt.P_{21}) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = Q_{44}$$

$$Q_{45} \xrightarrow{\tau} ((\underline{P}_{11} \parallel P_2) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = Q_{45}$$

$$\mathbf{D}_6 \xrightarrow{\mathbf{1}} ((\underline{b_1wt.P}_{11} \parallel \underline{P}_{21}) \parallel_B [\mathbf{PV}(false, true, 1)])[\Phi_B] = \mathbf{Q}_{46}$$

$$Q_{46} \xrightarrow{\tau} ((P_{11} \parallel \underline{P}_{21}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = Q_{47}$$

$$Q_{46} \xrightarrow{\tau} ((\underline{b_1wt.P}_{11} \parallel P_{24}) \parallel_B [\mathbf{PV}(false, true, 1)])[\Phi_B] = Q_{22}$$

$$Q_{47} \xrightarrow{\tau} ((P_{12} \parallel \underline{P}_{21}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = Q_{48}$$

$$Q_{47} \xrightarrow{\tau} ((P_{11} \parallel P_{22}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = D_{28}$$

$$Q_{48} \xrightarrow{\tau} ((P_{11} \parallel \underline{P}_{21}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = Q_{47}$$

$$Q_{48} \xrightarrow{\tau} ((P_{12} \parallel P_{22}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = D_{25}$$

$$\mathbf{D}_7 \xrightarrow{\mathbf{1}} ((\underline{P}_1 \parallel \underline{P}_{21}) \parallel_B [\mathbf{PV}(false, true, 1)])[\Phi_B] = \mathbf{Q}_{49}$$

$$Q_{49} \xrightarrow{\text{req}_1} ((b_1wt.P_{11} \parallel \underline{P}_{21}) \parallel_B [\mathbf{PV}(false, true, 1)])[\Phi_B] = Q_{50}$$

$$Q_{49} \xrightarrow{\tau} ((P_1 \parallel \underline{P}_{21}) \parallel_B [\mathbf{PV}(false, true, 1)])[\Phi_B] = Q_{51}$$

$$Q_{49} \xrightarrow{\tau} ((\underline{P}_1 \parallel P_{24}) \parallel_B [\mathbf{PV}(false, true, 1)])[\Phi_B] = Q_8$$

$$Q_{50} \xrightarrow{\tau} ((P_{11} \parallel \underline{P}_{21}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = Q_{47}$$

$$Q_{50} \xrightarrow{\tau} ((b_1wt.P_{11} \parallel P_{24}) \parallel_B [\mathbf{PV}(false, true, 1)])[\Phi_B] = D_{10}$$

$$Q_{51} \xrightarrow{\text{req}_1} ((b_1wt.P_{11} \parallel \underline{P}_{21}) \parallel_B [\mathbf{PV}(false, true, 1)])[\Phi_B] = Q_{50}$$

$$Q_{51} \xrightarrow{\tau} ((P_1 \parallel \underline{P}_{21}) \parallel_B [\mathbf{PV}(false, true, 1)])[\Phi_B] = Q_{51}$$

$$Q_{51} \xrightarrow{\tau} ((P_1 \parallel P_{24}) \parallel_B [\mathbf{PV}(false, true, 1)])[\Phi_B] = D_{11}$$

$$\mathbf{D}_8 \xrightarrow{\mathbf{1}} ((\underline{P}_{14} \parallel \underline{b_2wt.P}_{21}) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = \mathbf{Q}_{52}$$

$$Q_{52} \xrightarrow{\text{cs}_1} ((kw2.b_1wf.P_1 \parallel \underline{b_2wt.P}_{21}) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = Q_{23}$$

$$Q_{52} \xrightarrow{\tau} ((\underline{P}_{14} \parallel P_{21}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = Q_{53}$$

$$Q_{53} \xrightarrow{\text{cs}_1} ((kw2.b_1wf.P_1 \parallel P_{21}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = D_{21}$$

$$Q_{53} \xrightarrow{\tau} ((\underline{P}_{14} \parallel P_{22}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = Q_{54}$$

$$Q_{54} \xrightarrow{\text{cs}_1} ((kw2.b_1wf.P_1 \parallel P_{22}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = D_{29}$$

$$Q_{54} \xrightarrow{\tau} ((\underline{P}_{14} \parallel b_2wf.P_{23}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{55}$$

$$Q_{55} \xrightarrow{\text{cs}_1} ((kw2.b_1wf.P_1 \parallel b_2wf.P_{23}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = D_{30}$$

$$Q_{55} \xrightarrow{\tau} ((\underline{P}_{14} \parallel P_{23}) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = Q_{56}$$

$$Q_{56} \xrightarrow{\text{cs}_1} ((kw2.b_1wf.P_1 \parallel P_{23}) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = D_{31}$$

$$Q_{56} \xrightarrow{\tau} ((\underline{P}_{14} \parallel P_{23}) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = Q_{56}$$

$$\mathbf{D}_9 \xrightarrow{\mathbf{1}} ((\underline{\mathbf{P}}_{14} \parallel \underline{\mathbf{P}}_2) \parallel_{\mathbf{B}} [\text{PV}(true, false, 1)])[\Phi_{\mathbf{B}}] = \mathbf{Q}_{57}$$

$$Q_{57} \xrightarrow{\text{cs}_1} ((kw2.b_1wf.P_1 \parallel \underline{P}_2) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = Q_9$$

$$Q_{57} \xrightarrow{\text{req}_2} ((\underline{P}_{14} \parallel b_2wt.P_{21}) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = Q_{58}$$

$$Q_{57} \xrightarrow{\tau} ((\underline{P}_{14} \parallel P_2) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = Q_{59}$$

$$Q_{58} \xrightarrow{\text{cs}_1} ((kw2.b_1wf.P_1 \parallel b_2wt.P_{21}) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = D_{12}$$

$$Q_{58} \xrightarrow{\tau} ((\underline{P}_{14} \parallel P_{21}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{53}$$

$$Q_{59} \xrightarrow{\text{cs}_1} ((kw2.b_1wf.P_1 \parallel P_2) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = D_{13}$$

$$Q_{59} \xrightarrow{\text{req}_2} ((\underline{P}_{14} \parallel b_2wt.P_{21}) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = Q_{58}$$

$$Q_{59} \xrightarrow{\tau} ((\underline{P}_{14} \parallel P_2) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = Q_{59}$$

$$\mathbf{D}_{10} \xrightarrow{\mathbf{1}} ((\underline{b_1wt.P}_{11} \parallel \underline{\mathbf{P}}_{24}) \parallel_{\mathbf{B}} [\text{PV}(false, true, 1)])[\Phi_{\mathbf{B}}] = \mathbf{Q}_{60}$$

$$Q_{60} \xrightarrow{\tau} ((P_{11} \parallel \underline{P}_{24}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{61}$$

$$Q_{60} \xrightarrow{\text{cs}_2} ((\underline{b_1wt.P}_{11} \parallel kw1.b_2wf.P_2) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B] = Q_{24}$$

$$Q_{61} \xrightarrow{\tau} ((P_{12} \parallel \underline{P}_{24}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{62}$$

$$Q_{61} \xrightarrow{\text{cs}_2} ((P_{11} \parallel kw1.b_2wf.P_2) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = D_{22}$$

$$Q_{62} \xrightarrow{\tau} ((P_{11} \parallel \underline{P}_{24}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{61}$$

$$Q_{62} \xrightarrow{\text{cs}_2} ((P_{12} \parallel kw1.b_2wf.P_2) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = D_{32}$$

$$\mathbf{D}_{11} \xrightarrow{\mathbf{1}} ((\underline{\mathbf{P}}_1 \parallel \underline{\mathbf{P}}_{24}) \parallel_{\mathbf{B}} [\text{PV}(false, true, 1)])[\Phi_{\mathbf{B}}] = \mathbf{Q}_{63}$$

$$Q_{63} \xrightarrow{\text{req}_1} ((b_1wt.P_{11} \parallel \underline{P}_{24}) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B] = Q_{64}$$

$$Q_{63} \xrightarrow{\tau} ((P_1 \parallel \underline{P}_{24}) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B] = Q_{65}$$

$$Q_{63} \xrightarrow{\text{cs}_2} ((\underline{P}_1 \parallel kw1.b_2wf.P_2) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B] = Q_{10}$$

$$Q_{64} \xrightarrow{\tau} ((P_{11} \parallel \underline{P}_{24}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{61}$$

$$Q_{64} \xrightarrow{\text{cs}_2} ((b_1wt.P_{11} \parallel kw1.b_2wf.P_2) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B] = D_{14}$$

$$Q_{65} \xrightarrow{\text{req}_1} ((b_1wt.P_{11} \parallel \underline{P}_{24}) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B] = Q_{64}$$

$$Q_{65} \xrightarrow{\tau} ((P_1 \parallel \underline{P}_{24}) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B] = Q_{65}$$

$$Q_{65} \xrightarrow{\text{cs}_1} ((P_1 \parallel kw1.b_2wf.P_2) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B] = D_{15}$$

$$\mathbf{D}_{12} \xrightarrow{1} ((\underline{kw2}.b_1 wf.P_1 \parallel \underline{b_2 wt}.P_{21}) \parallel_B [PV(true, false, 1)])[\Phi_B] = \mathbf{Q}_{66}$$

$$Q_{66} \xrightarrow{\tau} ((b_1 wf.P_1 \parallel \underline{b_2 wt}.P_{21}) \parallel_B [PV(true, false, 2)])[\Phi_B] = Q_{25}$$

$$Q_{66} \xrightarrow{\tau} ((\underline{kw2}.b_1 wf.P_1 \parallel P_{21}) \parallel_B [PV(true, true, 1)])[\Phi_B] = Q_{67}$$

$$Q_{67} \xrightarrow{\tau} ((b_1 wf.P_1 \parallel P_{21}) \parallel_B [PV(true, true, 2)])[\Phi_B] = D_{23}$$

$$Q_{67} \xrightarrow{\tau} ((\underline{kw2}.b_1 wf.P_1 \parallel P_{22}) \parallel_B [PV(true, true, 1)])[\Phi_B] = Q_{68}$$

$$Q_{68} \xrightarrow{\tau} ((b_1 wf.P_1 \parallel P_{22}) \parallel_B [PV(true, true, 2)])[\Phi_B] = D_{33}$$

$$Q_{68} \xrightarrow{\tau} ((\underline{kw2}.b_1 wf.P_1 \parallel b_2 wf.P_{23}) \parallel_B [PV(true, true, 1)])[\Phi_B] = Q_{69}$$

$$Q_{69} \xrightarrow{\tau} ((b_1 wf.P_1 \parallel b_2 wf.P_{23}) \parallel_B [PV(true, true, 2)])[\Phi_B] = D_{34}$$

$$Q_{69} \xrightarrow{\tau} ((\underline{kw2}.b_1 wf.P_1 \parallel P_{23}) \parallel_B [PV(true, false, 1)])[\Phi_B] = Q_{70}$$

$$Q_{70} \xrightarrow{\tau} ((b_1 wf.P_1 \parallel P_{23}) \parallel_B [PV(true, false, 2)])[\Phi_B] = D_{35}$$

$$Q_{70} \xrightarrow{\tau} ((\underline{kw2}.b_1 wf.P_1 \parallel P_{23}) \parallel_B [PV(true, false, 1)])[\Phi_B] = Q_{70}$$

$$\mathbf{D}_{13} \xrightarrow{1} ((\underline{kw2}.b_1 wf.P_1 \parallel \underline{P}_2) \parallel_B [PV(true, false, 1)])[\Phi_B] = \mathbf{Q}_{71}$$

$$Q_{71} \xrightarrow{\tau} ((b_1 wf.P_1 \parallel \underline{P}_2) \parallel_B [PV(true, false, 2)])[\Phi_B] = Q_{11}$$

$$Q_{71} \xrightarrow{\text{req}_2} ((\underline{kw2}.b_1 wf.P_1 \parallel b_2 wt.P_{21}) \parallel_B [PV(true, false, 1)])[\Phi_B] = Q_{72}$$

$$Q_{71} \xrightarrow{\tau} ((\underline{kw2}.b_1 wf.P_1 \parallel P_2) \parallel_B [PV(true, false, 1)])[\Phi_B] = Q_{73}$$

$$Q_{72} \xrightarrow{\tau} ((b_1 wf.P_1 \parallel b_2 wt.P_{21}) \parallel_B [PV(true, false, 2)])[\Phi_B] = D_{16}$$

$$Q_{72} \xrightarrow{\tau} ((\underline{kw2}.b_1 wf.P_1 \parallel P_{21}) \parallel_B [PV(true, true, 1)])[\Phi_B] = Q_{67}$$

$$Q_{73} \xrightarrow{\tau} ((b_1 wf.P_1 \parallel P_2) \parallel_B [PV(true, false, 2)])[\Phi_B] = D_{17}$$

$$Q_{73} \xrightarrow{\text{req}_2} ((\underline{kw2}.b_1 wf.P_1 \parallel b_2 wt.P_{21}) \parallel_B [PV(true, false, 1)])[\Phi_B] = Q_{72}$$

$$Q_{73} \xrightarrow{\tau} ((\underline{kw2}.b_1 wf.P_1 \parallel P_2) \parallel_B [PV(true, false, 1)])[\Phi_B] = Q_{73}$$

$$\mathbf{D}_{14} \xrightarrow{1} ((\underline{b_1 wt}.P_{11} \parallel \underline{kw1}.b_2 wf.P_2) \parallel_B [PV(false, true, 1)])[\Phi_B] = \mathbf{Q}_{74}$$

$$Q_{74} \xrightarrow{\tau} ((P_{11} \parallel \underline{kw1}.b_2 wf.P_2) \parallel_B [PV(true, true, 1)])[\Phi_B] = Q_{75}$$

$$Q_{74} \xrightarrow{\tau} ((\underline{b_1 wt}.P_{11} \parallel b_2 wf.P_2) \parallel_B [PV(false, true, 1)])[\Phi_B] = Q_{26}$$

$$Q_{75} \xrightarrow{\tau} ((P_{12} \parallel \underline{kw1}.b_2 wf.P_2) \parallel_B [PV(true, true, 1)])[\Phi_B] = Q_{76}$$

$$Q_{75} \xrightarrow{\tau} ((P_{11} \parallel b_2 wf.P_2) \parallel_B [PV(true, true, 1)])[\Phi_B] = S_{23}$$

$$Q_{76} \xrightarrow{\tau} ((P_{11} \parallel \underline{kw1}.b_2 wf.P_2) \parallel_B [PV(true, true, 1)])[\Phi_B] = Q_{75}$$

$$Q_{76} \xrightarrow{\tau} ((P_{12} \parallel b_2 wf.P_2) \parallel_B [PV(true, true, 1)])[\Phi_B] = S_{33}$$

$$\mathbf{D}_{15} \xrightarrow{1} ((\underline{P}_1 \parallel \underline{kw1}.b_2 wf.P_2) \parallel_B [PV(false, true, 1)])[\Phi_B] = \mathbf{Q}_{77}$$

$$Q_{77} \xrightarrow{\text{req}_1} ((b_1 wt.P_{11} \parallel \underline{kw1}.b_2 wf.P_2) \parallel_B [PV(false, true, 1)])[\Phi_B] = Q_{78}$$

$$Q_{77} \xrightarrow{\tau} ((P_1 \parallel \underline{kw1}.b_2 wf.P_2) \parallel_B [PV(false, true, 1)])[\Phi_B] = Q_{79}$$

$$Q_{77} \xrightarrow{\tau} ((\underline{P}_1 \parallel b_2 wf.P_2) \parallel_B [PV(false, true, 1)])[\Phi_B] = Q_{12}$$

$$Q_{78} \xrightarrow{\tau} ((P_{11} \parallel \underline{kw1}.b_2wf.P_2) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{75}$$

$$Q_{78} \xrightarrow{\tau} ((b_1wt.P_{11} \parallel b_2wf.P_2) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B] = S_{16}$$

$$Q_{79} \xrightarrow{\text{req}_1} ((b_1wt.P_{11} \parallel \underline{kw1}.b_2wf.P_2) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B] = Q_{78}$$

$$Q_{79} \xrightarrow{\tau} ((P_1 \parallel \underline{kw1}.b_2wf.P_2) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B] = Q_{79}$$

$$Q_{79} \xrightarrow{\tau} ((P_1 \parallel b_2wf.P_2) \parallel_B [\text{PV}(false, true, 1)])[\Phi_B] = S_{17}$$

$$\mathbf{D}_{16} \xrightarrow{1} ((\underline{b_1wf}.P_1 \parallel \underline{b_2wt}.P_{21}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = \mathbf{Q}_{80}$$

$$Q_{80} \xrightarrow{\tau} ((P_1 \parallel \underline{b_2wt}.P_{21}) \parallel_B [\text{PV}(false, false, 2)])[\Phi_B] = Q_{27}$$

$$Q_{80} \xrightarrow{\tau} ((\underline{b_1wf}.P_1 \parallel P_{21}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = Q_{81}$$

$$Q_{81} \xrightarrow{\tau} ((P_1 \parallel P_{21}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = S_5$$

$$Q_{81} \xrightarrow{\tau} ((\underline{b_1wf}.P_1 \parallel P_{22}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = Q_{82}$$

$$Q_{82} \xrightarrow{\tau} ((P_1 \parallel P_{22}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = D_{36}$$

$$Q_{82} \xrightarrow{\tau} ((\underline{b_1wf}.P_1 \parallel P_{21}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = Q_{81}$$

$$\mathbf{D}_{17} \xrightarrow{1} ((\underline{b_1wf}.P_1 \parallel \underline{P}_2) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = \mathbf{Q}_{83}$$

$$Q_{83} \xrightarrow{\tau} ((P_1 \parallel \underline{P}_2) \parallel_B [\text{PV}(false, false, 2)])[\Phi_B] = Q_{13}$$

$$Q_{83} \xrightarrow{\text{req}_2} ((\underline{b_1wf}.P_1 \parallel b_2wt.P_{21}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = Q_{84}$$

$$Q_{83} \xrightarrow{\tau} ((\underline{b_1wf}.P_1 \parallel P_2) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = Q_{85}$$

$$Q_{84} \xrightarrow{\tau} ((P_1 \parallel b_2wt.P_{21}) \parallel_B [\text{PV}(false, false, 2)])[\Phi_B] = S_2$$

$$Q_{84} \xrightarrow{\tau} ((\underline{b_1wf}.P_1 \parallel P_{21}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = Q_{81}$$

$$Q_{85} \xrightarrow{\tau} ((P_1 \parallel P_2) \parallel_B [\text{PV}(false, false, 2)])[\Phi_B] = \mathcal{S}(\text{Dekker})$$

$$Q_{85} \xrightarrow{\text{req}_2} ((\underline{b_1wf}.P_1 \parallel b_2wt.P_{21}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = Q_{84}$$

$$Q_{85} \xrightarrow{\tau} ((\underline{b_1wf}.P_1 \parallel P_2) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = Q_{85}$$

$$\mathbf{D}_{18} \xrightarrow{1} ((\underline{P}_{11} \parallel \underline{P}_{21}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = \mathbf{Q}_{86}$$

$$Q_{86} \xrightarrow{\tau} ((P_{12} \parallel \underline{P}_{21}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{48}$$

$$Q_{86} \xrightarrow{\tau} ((\underline{P}_{11} \parallel P_{22}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{40}$$

$$\mathbf{D}_{19} \xrightarrow{1} ((\underline{P}_{14} \parallel \underline{P}_{21}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = \mathbf{Q}_{87}$$

$$Q_{87} \xrightarrow{\text{cs}_1} ((kw2.b_1wf.P_1 \parallel \underline{P}_{21}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{88}$$

$$Q_{87} \xrightarrow{\tau} ((\underline{P}_{14} \parallel P_{22}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{54}$$

$$Q_{88} \xrightarrow{\tau} ((b_1wf.P_1 \parallel \underline{P}_{21}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = Q_{89}$$

$$Q_{88} \xrightarrow{\tau} ((kw2.b_1wf.P_1 \parallel P_{22}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = D_{29}$$

$$Q_{89} \xrightarrow{\tau} ((P_1 \parallel \underline{P}_{21}) \parallel_B [\text{PV}(\text{false}, \text{true}, 2)])[\Phi_B] = Q_{90}$$

$$Q_{89} \xrightarrow{\tau} ((b_1 \text{ wf}.P_1 \parallel P_{22}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = D_{33}$$

$$Q_{90} \xrightarrow{\text{req}_1} ((b_1 \text{ wt}.P_{11} \parallel \underline{P}_{21}) \parallel_B [\text{PV}(\text{false}, \text{true}, 2)])[\Phi_B] = Q_{91}$$

$$Q_{90} \xrightarrow{\tau} ((P_1 \parallel \underline{P}_{21}) \parallel_B [\text{PV}(\text{false}, \text{true}, 2)])[\Phi_B] = Q_{90}$$

$$Q_{90} \xrightarrow{\tau} ((P_1 \parallel P_{24}) \parallel_B [\text{PV}(\text{false}, \text{true}, 2)])[\Phi_B] = S_9$$

$$Q_{91} \xrightarrow{\tau} ((P_{11} \parallel \underline{P}_{21}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = Q_{92}$$

$$Q_{91} \xrightarrow{\tau} ((b_1 \text{ wt}.P_{11} \parallel P_{24}) \parallel_B [\text{PV}(\text{false}, \text{true}, 2)])[\Phi_B] = S_8$$

$$Q_{92} \xrightarrow{\tau} ((P_{12} \parallel \underline{P}_{21}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = Q_{93}$$

$$Q_{92} \xrightarrow{\tau} ((P_{11} \parallel P_{22}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = S_{24}$$

$$Q_{93} \xrightarrow{\tau} ((b_1 \text{ wf}.P_{13} \parallel \underline{P}_{21}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = Q_{94}$$

$$Q_{93} \xrightarrow{\tau} ((P_{12} \parallel P_{22}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = S_{25}$$

$$Q_{94} \xrightarrow{\tau} ((P_{13} \parallel \underline{P}_{21}) \parallel_B [\text{PV}(\text{false}, \text{true}, 2)])[\Phi_B] = Q_{95}$$

$$Q_{94} \xrightarrow{\tau} ((b_1 \text{ wf}.P_{13} \parallel P_{22}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = S_{26}$$

$$Q_{95} \xrightarrow{\tau} ((P_{13} \parallel \underline{P}_{21}) \parallel_B [\text{PV}(\text{false}, \text{true}, 2)])[\Phi_B] = Q_{95}$$

$$Q_{95} \xrightarrow{\tau} ((P_{13} \parallel P_{24}) \parallel_B [\text{PV}(\text{false}, \text{true}, 2)])[\Phi_B] = S_{27}$$

$$\mathbf{D}_{20} \xrightarrow{\mathbf{1}} ((\underline{\mathbf{P}}_{11} \parallel \underline{\mathbf{P}}_{24}) \parallel_{\mathbf{B}} [\text{PV}(\text{true}, \text{true}, \mathbf{1})])[\Phi_{\mathbf{B}}] = \mathbf{Q}_{96}$$

$$Q_{96} \xrightarrow{\tau} ((P_{12} \parallel \underline{P}_{24}) \parallel_B [\text{PV}(\text{true}, \text{true}, 1)])[\Phi_B] = Q_{62}$$

$$Q_{96} \xrightarrow{\text{cs}_2} ((\underline{P}_{11} \parallel \underline{kw1}.b_2 \text{ wf}.P_2) \parallel_B [\text{PV}(\text{true}, \text{true}, 1)])[\Phi_B] = Q_{97}$$

$$Q_{97} \xrightarrow{\tau} ((P_{12} \parallel \underline{kw1}.b_2 \text{ wf}.P_2) \parallel_B [\text{PV}(\text{true}, \text{true}, 1)])[\Phi_B] = D_{32}$$

$$Q_{97} \xrightarrow{\tau} ((\underline{P}_{11} \parallel \underline{b_2 \text{ wf}.P_2}) \parallel_B [\text{PV}(\text{true}, \text{true}, 1)])[\Phi_B] = Q_{98}$$

$$Q_{98} \xrightarrow{\tau} ((P_{12} \parallel \underline{b_2 \text{ wf}.P_2}) \parallel_B [\text{PV}(\text{true}, \text{true}, 1)])[\Phi_B] = S_{33}$$

$$Q_{98} \xrightarrow{\tau} ((\underline{P}_{11} \parallel \underline{P_2}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = Q_{99}$$

$$Q_{99} \xrightarrow{\tau} ((P_{14} \parallel P_2) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = D_9$$

$$Q_{99} \xrightarrow{\text{req}_2} ((\underline{P}_{11} \parallel \underline{b_2 \text{ wt}.P_{21}}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = Q_{44}$$

$$Q_{99} \xrightarrow{\tau} ((\underline{P}_{11} \parallel \underline{P_2}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = Q_{99}$$

$$\mathbf{D}_{21} \xrightarrow{\mathbf{1}} ((\underline{kw2}.b_1 \text{ wf}.\mathbf{P}_1 \parallel \underline{\mathbf{P}}_{21}) \parallel_{\mathbf{B}} [\text{PV}(\text{true}, \text{true}, \mathbf{1})])[\Phi_{\mathbf{B}}] = \mathbf{Q}_{100}$$

$$Q_{100} \xrightarrow{\tau} ((b_1 \text{ wf}.P_1 \parallel \underline{P}_{21}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = Q_{89}$$

$$Q_{100} \xrightarrow{\tau} ((\underline{kw2}.b_1 \text{ wf}.P_1 \parallel P_{22}) \parallel_B [\text{PV}(\text{true}, \text{true}, 1)])[\Phi_B] = Q_{68}$$

$$\mathbf{D}_{22} \xrightarrow{\mathbf{1}} ((\underline{\mathbf{P}}_{11} \parallel \underline{kw1}.b_2 \text{ wf}.\mathbf{P}_2) \parallel_{\mathbf{B}} [\text{PV}(\text{true}, \text{true}, \mathbf{1})])[\Phi_{\mathbf{B}}] = \mathbf{Q}_{101}$$

$$Q_{101} \xrightarrow{\tau} ((P_{12} \parallel \underline{kw1}.b_2 \text{ wf}.P_2) \parallel_B [\text{PV}(\text{true}, \text{true}, 1)])[\Phi_B] = Q_{76}$$

$$Q_{101} \xrightarrow{\tau} ((\underline{P}_{11} \parallel \underline{b_2 \text{ wf}.P_2}) \parallel_B [\text{PV}(\text{true}, \text{true}, 1)])[\Phi_B] = Q_{98}$$

$$\mathbf{D}_{23} \xrightarrow{1} ((\underline{b_1 wf}.P_1 \parallel \underline{P}_{21}) \parallel_B [\mathbf{PV}(true, true, 2)])[\Phi_B] = \mathbf{Q}_{102}$$

$$Q_{102} \xrightarrow{\tau} ((P_1 \parallel \underline{P}_{21}) \parallel_B [\mathbf{PV}(false, true, 2)])[\Phi_B] = Q_{90}$$

$$Q_{102} \xrightarrow{\tau} ((\underline{b_1 wf}.P_1 \parallel P_{22}) \parallel_B [\mathbf{PV}(true, true, 2)])[\Phi_B] = Q_{82}$$

$$\mathbf{D}_{24} \xrightarrow{1} ((\underline{P}_{12} \parallel \underline{P}_{21}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = \mathbf{Q}_{103}$$

$$Q_{103} \xrightarrow{\tau} ((P_{11} \parallel \underline{P}_{21}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = Q_{47}$$

$$Q_{103} \xrightarrow{\tau} ((\underline{P}_{12} \parallel P_{22}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = Q_{104}$$

$$Q_{104} \xrightarrow{\tau} ((P_{11} \parallel P_{22}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = D_{28}$$

$$Q_{104} \xrightarrow{\tau} ((\underline{P}_{12} \parallel b_2 wf.P_{23}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = Q_{105}$$

$$Q_{105} \xrightarrow{\tau} ((P_{11} \parallel b_2 wf.P_{23}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = D_{37}$$

$$Q_{105} \xrightarrow{\tau} ((\underline{P}_{12} \parallel P_{23}) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = Q_{106}$$

$$Q_{106} \xrightarrow{\tau} ((P_{11} \parallel P_{23}) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = D_{38}$$

$$Q_{106} \xrightarrow{\tau} ((\underline{P}_{12} \parallel P_{23}) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = Q_{106}$$

$$\mathbf{D}_{25} \xrightarrow{1} ((\underline{P}_{12} \parallel \underline{P}_{22}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = \mathbf{Q}_{107}$$

$$Q_{107} \xrightarrow{\tau} ((P_{11} \parallel \underline{P}_{22}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = Q_{108}$$

$$Q_{107} \xrightarrow{\tau} ((\underline{P}_{12} \parallel b_2 wf.P_{23}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = Q_{105}$$

$$Q_{108} \xrightarrow{\tau} ((P_{12} \parallel \underline{P}_{22}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = Q_{109}$$

$$Q_{108} \xrightarrow{\tau} ((P_{11} \parallel b_2 wf.P_{23}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = D_{37}$$

$$Q_{109} \xrightarrow{\tau} ((P_{11} \parallel \underline{P}_{22}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = Q_{108}$$

$$Q_{108} \xrightarrow{\tau} ((P_{12} \parallel b_2 wf.P_{23}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = D_{26}$$

$$\mathbf{D}_{26} \xrightarrow{1} ((\underline{P}_{12} \parallel \underline{b_2 wf}.P_{23}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = \mathbf{Q}_{110}$$

$$Q_{110} \xrightarrow{\tau} ((P_{11} \parallel \underline{b_2 wf}.P_{23}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = Q_{111}$$

$$Q_{110} \xrightarrow{\tau} ((\underline{P}_{12} \parallel P_{23}) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = Q_{106}$$

$$Q_{111} \xrightarrow{\tau} ((P_{12} \parallel \underline{b_2 wf}.P_{23}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = Q_{112}$$

$$Q_{111} \xrightarrow{\tau} ((P_{11} \parallel P_{23}) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = D_{38}$$

$$Q_{112} \xrightarrow{\tau} ((P_{11} \parallel \underline{b_2 wf}.P_{23}) \parallel_B [\mathbf{PV}(true, true, 1)])[\Phi_B] = Q_{111}$$

$$Q_{112} \xrightarrow{\tau} ((P_{12} \parallel P_{23}) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = D_{39}$$

$$\mathbf{D}_{27} \xrightarrow{1} ((\underline{P}_{14} \parallel \underline{P}_{23}) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = \mathbf{Q}_{113}$$

$$Q_{113} \xrightarrow{cs_1} ((kw2.b_1 wf.P_1 \parallel \underline{P}_{23}) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = Q_{114}$$

$$Q_{113} \xrightarrow{\tau} ((\underline{P}_{14} \parallel P_{23}) \parallel_B [\mathbf{PV}(true, false, 1)])[\Phi_B] = Q_{56}$$

$$Q_{114} \xrightarrow{\tau} ((b_1 wf.P_1 \parallel \underline{P}_{23}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = Q_{115}$$

$$Q_{114} \xrightarrow{\tau} ((kw2.b_1 wf.P_1 \parallel P_{23}) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = D_{31}$$

$$Q_{115} \xrightarrow{\tau} ((P_1 \parallel \underline{P}_{23}) \parallel_B [\text{PV}(false, false, 2)])[\Phi_B] = Q_{116}$$

$$Q_{115} \xrightarrow{\tau} ((b_1 wf.P_1 \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = D_{16}$$

$$Q_{116} \xrightarrow{\text{req}_1} ((b_1 wt.P_{11} \parallel \underline{P}_{23}) \parallel_B [\text{PV}(false, false, 2)])[\Phi_B] = Q_{117}$$

$$Q_{116} \xrightarrow{\tau} ((P_1 \parallel \underline{P}_{23}) \parallel_B [\text{PV}(false, false, 2)])[\Phi_B] = Q_{116}$$

$$Q_{116} \xrightarrow{\tau} ((P_1 \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(false, false, 2)])[\Phi_B] = S_2$$

$$Q_{117} \xrightarrow{\tau} ((P_{11} \parallel \underline{P}_{23}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = Q_{118}$$

$$Q_{117} \xrightarrow{\tau} ((b_1 wt.P_{11} \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(false, false, 2)])[\Phi_B] = S_1$$

$$Q_{118} \xrightarrow{\tau} ((P_{14} \parallel \underline{P}_{23}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = Q_{119}$$

$$Q_{118} \xrightarrow{\tau} ((P_{11} \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = S_6$$

$$Q_{119} \xrightarrow{\text{cs}_1} ((kw2.b_1 wf.P_1 \parallel \underline{P}_{23}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = Q_{120}$$

$$Q_{119} \xrightarrow{\tau} ((P_{14} \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = S_{10}$$

$$Q_{120} \xrightarrow{\tau} ((b_1 wf.P_1 \parallel \underline{P}_{23}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = Q_{115}$$

$$Q_{120} \xrightarrow{\text{cs}_1} ((kw2.b_1 wf.P_1 \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = S_{14}$$

$$D_{28} \xrightarrow{1} ((\underline{P}_{11} \parallel \underline{P}_{22}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{121}$$

$$Q_{121} \xrightarrow{\tau} ((P_{12} \parallel \underline{P}_{22}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{109}$$

$$Q_{121} \xrightarrow{\tau} ((\underline{P}_{11} \parallel b_2 wf.P_{23}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{41}$$

$$D_{29} \xrightarrow{1} ((\underline{kw2}.b_1 wf.P_1 \parallel \underline{P}_{22}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{122}$$

$$Q_{122} \xrightarrow{\tau} ((b_1 wf.P_1 \parallel \underline{P}_{22}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = Q_{123}$$

$$Q_{122} \xrightarrow{\tau} ((\underline{kw2}.b_1 wf.P_1 \parallel b_2 wf.P_{23}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{69}$$

$$Q_{123} \xrightarrow{\tau} ((P_1 \parallel \underline{P}_{22}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = Q_{124}$$

$$Q_{123} \xrightarrow{\tau} ((b_1 wf.P_1 \parallel P_{21}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = D_{23}$$

$$Q_{124} \xrightarrow{\text{req}_1} ((b_1 wt.P_{11} \parallel \underline{P}_{22}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = Q_{125}$$

$$Q_{124} \xrightarrow{\tau} ((P_1 \parallel \underline{P}_{22}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = Q_{124}$$

$$Q_{124} \xrightarrow{\tau} ((P_1 \parallel P_{21}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = S_5$$

$$Q_{125} \xrightarrow{\tau} ((P_{11} \parallel \underline{P}_{22}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = Q_{126}$$

$$Q_{125} \xrightarrow{\tau} ((b_1 wt.P_{11} \parallel P_{21}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = S_4$$

$$Q_{126} \xrightarrow{\tau} ((P_{12} \parallel \underline{P}_{22}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = Q_{127}$$

$$Q_{126} \xrightarrow{\tau} ((P_{11} \parallel P_{21}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = S_{18}$$

$$Q_{127} \xrightarrow{\tau} ((b_1 wf.P_{13} \parallel \underline{P}_{22}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = Q_{128}$$

$$Q_{127} \xrightarrow{\tau} ((P_{12} \parallel P_{21}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = S_{28}$$

$$Q_{128} \xrightarrow{\tau} ((P_{13} \parallel \underline{P}_{22}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = Q_{129}$$

$$Q_{128} \xrightarrow{\tau} ((b_1 wf.P_{13} \parallel P_{21}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = S_{37}$$

$$Q_{129} \xrightarrow{\tau} ((P_{13} \parallel \underline{P}_{22}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = Q_{129}$$

$$Q_{129} \xrightarrow{\tau} ((P_{13} \parallel P_{21}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = S_{38}$$

$$\mathbf{D}_{30} \xrightarrow{\mathbf{1}} ((\underline{kw2}.b_1 wf.P_1 \parallel \underline{b_2 wf}.P_{23}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = \mathbf{Q}_{130}$$

$$Q_{130} \xrightarrow{\tau} ((b_1 wf.P_1 \parallel \underline{b_2 wf}.P_{23}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = Q_{131}$$

$$Q_{130} \xrightarrow{\tau} ((\underline{kw2}.b_1 wf.P_1 \parallel P_{23}) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = Q_{70}$$

$$Q_{131} \xrightarrow{\tau} ((P_1 \parallel \underline{b_2 wf}.P_{23}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = Q_{132}$$

$$Q_{131} \xrightarrow{\tau} ((b_1 wf.P_1 \parallel P_{23}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = D_{35}$$

$$Q_{132} \xrightarrow{\text{req}_1} ((b_1 wt.P_{11} \parallel \underline{b_2 wf}.P_{23}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = Q_{133}$$

$$Q_{132} \xrightarrow{\tau} ((P_1 \parallel \underline{b_2 wf}.P_{23}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = Q_{132}$$

$$Q_{132} \xrightarrow{\tau} ((P_1 \parallel P_{23}) \parallel_B [\text{PV}(false, false, 2)])[\Phi_B] = Q_{40}$$

$$Q_{133} \xrightarrow{\tau} ((P_{11} \parallel \underline{b_2 wf}.P_{23}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = Q_{134}$$

$$Q_{133} \xrightarrow{\tau} ((b_1 wt.P_{11} \parallel P_{23}) \parallel_B [\text{PV}(false, false, 2)])[\Phi_B] = D_{41}$$

$$Q_{134} \xrightarrow{\tau} ((P_{12} \parallel \underline{b_2 wf}.P_{23}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = Q_{135}$$

$$Q_{134} \xrightarrow{\tau} ((P_{11} \parallel P_{23}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = D_{42}$$

$$Q_{135} \xrightarrow{\tau} ((b_1 wf.P_{13} \parallel \underline{b_2 wf}.P_{23}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = Q_{136}$$

$$Q_{135} \xrightarrow{\tau} ((P_{12} \parallel P_{23}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = D_{43}$$

$$Q_{136} \xrightarrow{\tau} ((P_{13} \parallel \underline{b_2 wf}.P_{23}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = Q_{137}$$

$$Q_{136} \xrightarrow{\tau} ((b_1 wf.P_{13} \parallel P_{23}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = D_{44}$$

$$Q_{137} \xrightarrow{\tau} ((P_{13} \parallel \underline{b_2 wf}.P_{23}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = Q_{137}$$

$$Q_{137} \xrightarrow{\tau} ((P_{13} \parallel P_{23}) \parallel_B [\text{PV}(false, false, 2)])[\Phi_B] = D_{45}$$

$$\mathbf{D}_{31} \xrightarrow{\mathbf{1}} ((\underline{kw2}.b_1 wf.P_1 \parallel \underline{P}_{23}) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = \mathbf{Q}_{138}$$

$$Q_{138} \xrightarrow{\tau} ((b_1 wf.P_1 \parallel \underline{P}_{23}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = Q_{115}$$

$$Q_{138} \xrightarrow{\tau} ((\underline{kw2}.b_1 wf.P_1 \parallel P_{23}) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = Q_{70}$$

$$\mathbf{D}_{32} \xrightarrow{\mathbf{1}} ((\underline{P}_{12} \parallel \underline{kw1}.b_2 wf.P_2) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = \mathbf{Q}_{139}$$

$$Q_{139} \xrightarrow{\tau} ((P_{11} \parallel \underline{kw1}.b_2 wf.P_2) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{75}$$

$$Q_{139} \xrightarrow{\tau} ((\underline{P}_{12} \parallel b_2 wf.P_2) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{140}$$

$$Q_{140} \xrightarrow{\tau} ((P_{11} \parallel b_2 wf.P_2) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = S_{23}$$

$$Q_{140} \xrightarrow{\tau} ((\underline{P}_{12} \parallel P_2) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = Q_{141}$$

$$\begin{aligned}
Q_{141} &\xrightarrow{\tau} ((P_{11} \parallel P_2) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = D_5 \\
Q_{141} &\xrightarrow{\text{req}_2} ((\underline{P}_{12} \parallel b_2wt.P_{21}) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = Q_{142} \\
Q_{141} &\xrightarrow{\tau} ((\underline{P}_{12} \parallel P_2) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = Q_{141}
\end{aligned}$$

$$\begin{aligned}
Q_{142} &\xrightarrow{\tau} ((P_{11} \parallel b_2wt.P_{21}) \parallel_B [\text{PV}(true, false, 1)])[\Phi_B] = D_4 \\
Q_{142} &\xrightarrow{\tau} ((\underline{P}_{12} \parallel P_{21}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{143}
\end{aligned}$$

$$\begin{aligned}
Q_{143} &\xrightarrow{\tau} ((P_{11} \parallel P_{21}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = D_{18} \\
Q_{143} &\xrightarrow{\tau} ((\underline{P}_{12} \parallel P_{22}) \parallel_B [\text{PV}(true, true, 1)])[\Phi_B] = Q_{104}
\end{aligned}$$

$$\mathbf{D}_{33} \xrightarrow{1} ((\underline{b_1wf.P_1} \parallel \underline{P_{22}}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = \mathbf{Q}_{144}$$

$$\begin{aligned}
Q_{144} &\xrightarrow{\tau} ((P_1 \parallel \underline{P_{22}}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = Q_{124} \\
Q_{144} &\xrightarrow{\tau} ((\underline{b_1wf.P_1} \parallel P_{21}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = Q_{81}
\end{aligned}$$

$$\mathbf{D}_{34} \xrightarrow{1} ((\underline{b_1wf.P_1} \parallel \underline{b_2wf.P_{23}}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = \mathbf{Q}_{145}$$

$$\begin{aligned}
Q_{145} &\xrightarrow{\tau} ((P_1 \parallel \underline{b_2wf.P_{23}}) \parallel_B [\text{PV}(true, true, 2)])[\Phi_B] = Q_{132} \\
Q_{145} &\xrightarrow{\tau} ((\underline{b_1wf.P_1} \parallel P_{23}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = Q_{146}
\end{aligned}$$

$$\begin{aligned}
Q_{146} &\xrightarrow{\tau} ((P_1 \parallel P_{23}) \parallel_B [\text{PV}(false, false, 2)])[\Phi_B] = D_{40} \\
Q_{146} &\xrightarrow{\tau} ((\underline{b_1wf.P_1} \parallel b_2wt.P_{21}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = Q_{84}
\end{aligned}$$

$$\mathbf{D}_{35} \xrightarrow{1} ((\underline{b_1wf.P_1} \parallel \underline{P_{23}}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = \mathbf{Q}_{147}$$

$$\begin{aligned}
Q_{147} &\xrightarrow{\tau} ((P_1 \parallel \underline{P_{23}}) \parallel_B [\text{PV}(false, false, 2)])[\Phi_B] = Q_{116} \\
Q_{147} &\xrightarrow{\tau} ((\underline{b_1wf.P_1} \parallel b_2wt.P_{21}) \parallel_B [\text{PV}(true, false, 2)])[\Phi_B] = Q_{84}
\end{aligned}$$

$$\mathbf{D}_{36} \xrightarrow{1} ((\underline{P_1} \parallel \underline{P_{22}}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = \mathbf{Q}_{148}$$

$$\begin{aligned}
Q_{148} &\xrightarrow{\text{req}_1} ((b_1wt.P_{11} \parallel \underline{P_{22}}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = Q_{125} \\
Q_{148} &\xrightarrow{\tau} ((P_1 \parallel \underline{P_{22}}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = Q_{124} \\
Q_{148} &\xrightarrow{\tau} ((\underline{P_1} \parallel P_{21}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = Q_{149}
\end{aligned}$$

$$\begin{aligned}
Q_{149} &\xrightarrow{\text{req}_1} ((b_1wt.P_{11} \parallel P_{21}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = S_4 \\
Q_{149} &\xrightarrow{\tau} ((P_1 \parallel P_{21}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = S_5 \\
Q_{149} &\xrightarrow{\tau} ((\underline{P_1} \parallel P_{24}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = Q_{150}
\end{aligned}$$

$$\begin{aligned}
Q_{150} &\xrightarrow{\text{req}_1} ((b_1wt.P_{11} \parallel P_{24}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = S_8 \\
Q_{150} &\xrightarrow{\tau} ((P_1 \parallel P_{24}) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = S_9 \\
Q_{150} &\xrightarrow{\text{cs}_2} ((\underline{P_1} \parallel kw1.b_2wf.P_2) \parallel_B [\text{PV}(false, true, 2)])[\Phi_B] = Q_{151}
\end{aligned}$$

$$\begin{aligned}
Q_{151} &\xrightarrow{\text{req}_1} ((b_1 \text{wt}.P_{11} \parallel kw1.b_2 \text{wf}.P_2) \parallel_B [\text{PV}(\text{false}, \text{true}, 2)])[\Phi_B] = S_{12} \\
Q_{151} &\xrightarrow{\tau} ((P_1 \parallel kw1.b_2 \text{wf}.P_2) \parallel_B [\text{PV}(\text{false}, \text{true}, 2)])[\Phi_B] = S_{13} \\
Q_{151} &\xrightarrow{\text{cs}_2} ((\underline{P}_1 \parallel b_2 \text{wf}.P_2) \parallel_B [\text{PV}(\text{false}, \text{true}, 1)])[\Phi_B] = Q_{12}
\end{aligned}$$

$$D_{37} \xrightarrow{1} ((\underline{P}_{11} \parallel \underline{b_2 \text{wf}.P_{23}}) \parallel_B [\text{PV}(\text{true}, \text{true}, 1)])[\Phi_B] = Q_{152}$$

$$\begin{aligned}
Q_{152} &\xrightarrow{\tau} ((P_{12} \parallel \underline{b_2 \text{wf}.P_{23}}) \parallel_B [\text{PV}(\text{true}, \text{true}, 1)])[\Phi_B] = Q_{112} \\
Q_{152} &\xrightarrow{\tau} ((\underline{P}_{11} \parallel P_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = Q_{42}
\end{aligned}$$

$$D_{38} \xrightarrow{1} ((\underline{P}_{11} \parallel \underline{P}_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = Q_{153}$$

$$\begin{aligned}
Q_{153} &\xrightarrow{\tau} ((P_{14} \parallel \underline{P}_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = Q_{154} \\
Q_{153} &\xrightarrow{\tau} ((\underline{P}_{11} \parallel P_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = Q_{42}
\end{aligned}$$

$$\begin{aligned}
Q_{154} &\xrightarrow{\text{cs}_1} ((kw2.b_1 \text{wf}.P_1 \parallel \underline{P}_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = Q_{114} \\
Q_{154} &\xrightarrow{\tau} ((P_{14} \parallel P_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = D_{27}
\end{aligned}$$

$$D_{39} \xrightarrow{1} ((\underline{P}_{12} \parallel \underline{P}_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = Q_{155}$$

$$\begin{aligned}
Q_{155} &\xrightarrow{\tau} ((P_{11} \parallel \underline{P}_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = Q_{156} \\
Q_{155} &\xrightarrow{\tau} ((\underline{P}_{12} \parallel P_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = Q_{106}
\end{aligned}$$

$$\begin{aligned}
Q_{156} &\xrightarrow{\tau} ((P_{14} \parallel \underline{P}_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = Q_{154} \\
Q_{156} &\xrightarrow{\tau} ((P_{11} \parallel P_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 1)])[\Phi_B] = D_{38}
\end{aligned}$$

$$D_{40} \xrightarrow{1} ((\underline{P}_1 \parallel \underline{P}_{23}) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])[\Phi_B] = Q_{157}$$

$$\begin{aligned}
Q_{157} &\xrightarrow{\text{req}_1} ((b_1 \text{wt}.P_{11} \parallel \underline{P}_{23}) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])[\Phi_B] = Q_{117} \\
Q_{157} &\xrightarrow{\tau} ((P_1 \parallel \underline{P}_{23}) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])[\Phi_B] = Q_{116} \\
Q_{157} &\xrightarrow{\tau} ((\underline{P}_1 \parallel b_2 \text{wt}.P_{21}) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])[\Phi_B] = Q_{158}
\end{aligned}$$

$$\begin{aligned}
Q_{158} &\xrightarrow{\text{req}_1} ((b_1 \text{wt}.P_{11} \parallel b_2 \text{wt}.P_{21}) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])[\Phi_B] = S_1 \\
Q_{158} &\xrightarrow{\tau} ((P_1 \parallel b_2 \text{wt}.P_{21}) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])[\Phi_B] = S_2 \\
Q_{158} &\xrightarrow{\tau} ((\underline{P}_1 \parallel P_{21}) \parallel_B [\text{PV}(\text{false}, \text{true}, 2)])[\Phi_B] = Q_{149}
\end{aligned}$$

$$D_{41} \xrightarrow{1} ((\underline{b_1 \text{wt}.P_{11}} \parallel \underline{P}_{23}) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])[\Phi_B] = Q_{159}$$

$$\begin{aligned}
Q_{159} &\xrightarrow{\tau} ((P_{11} \parallel \underline{P}_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])[\Phi_B] = Q_{118} \\
Q_{159} &\xrightarrow{\tau} ((\underline{b_1 \text{wt}.P_{11}} \parallel b_2 \text{wt}.P_{21}) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])[\Phi_B] = Q_{160}
\end{aligned}$$

$$\begin{aligned}
Q_{160} &\xrightarrow{\tau} ((P_{11} \parallel b_2 \text{wt}.P_{21}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])[\Phi_B] = S_6 \\
Q_{160} &\xrightarrow{\tau} ((\underline{b_1 \text{wt}.P_{11}} \parallel P_{21}) \parallel_B [\text{PV}(\text{false}, \text{true}, 2)])[\Phi_B] = Q_{161}
\end{aligned}$$

$$\begin{aligned}
Q_{161} &\xrightarrow{\tau} ((P_{11} \parallel P_{21}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = S_{18} \\
Q_{161} &\xrightarrow{\tau} ((\underline{b_1 wt}.P_{11} \parallel P_{24}) \parallel_B [\text{PV}(\text{false}, \text{true}, 2)])[\Phi_B] = Q_{162} \\
Q_{162} &\xrightarrow{\tau} ((P_{11} \parallel P_{24}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = S_{19} \\
Q_{162} &\xrightarrow{\text{cs}_2} ((\underline{b_1 wt}.P_{11} \parallel \text{kw}1.b_2 wf.P_2) \parallel_B [\text{PV}(\text{false}, \text{true}, 2)])[\Phi_B] = Q_{163} \\
Q_{163} &\xrightarrow{\tau} ((P_{11} \parallel \text{kw}1.b_2 wf.P_2) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = S_{21} \\
Q_{163} &\xrightarrow{\tau} ((\underline{b_1 wt}.P_{11} \parallel b_2 wf.P_2) \parallel_B [\text{PV}(\text{false}, \text{true}, 1)])[\Phi_B] = Q_{26}
\end{aligned}$$

$$\mathbf{D}_{42} \xrightarrow{\mathbf{1}} ((\underline{\mathbf{P}}_{11} \parallel \underline{\mathbf{P}}_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])[\Phi_B] = \mathbf{Q}_{164}$$

$$\begin{aligned}
Q_{164} &\xrightarrow{\tau} ((P_{14} \parallel \underline{P}_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])[\Phi_B] = Q_{119} \\
Q_{164} &\xrightarrow{\tau} ((\underline{P}_{11} \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])[\Phi_B] = Q_{165}
\end{aligned}$$

$$\begin{aligned}
Q_{165} &\xrightarrow{\tau} ((P_{14} \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])[\Phi_B] = S_{10} \\
Q_{165} &\xrightarrow{\tau} ((\underline{P}_{11} \parallel P_{21}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = Q_{166}
\end{aligned}$$

$$\begin{aligned}
Q_{166} &\xrightarrow{\tau} ((P_{12} \parallel P_{21}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = S_{28} \\
Q_{166} &\xrightarrow{\tau} ((\underline{P}_{11} \parallel P_{22}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = Q_{167}
\end{aligned}$$

$$\begin{aligned}
Q_{167} &\xrightarrow{\tau} ((P_{12} \parallel P_{22}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = S_{25} \\
Q_{167} &\xrightarrow{\tau} ((\underline{P}_{11} \parallel P_{21}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = Q_{166}
\end{aligned}$$

$$\mathbf{D}_{43} \xrightarrow{\mathbf{1}} ((\underline{\mathbf{P}}_{12} \parallel \underline{\mathbf{P}}_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])[\Phi_B] = \mathbf{Q}_{168}$$

$$\begin{aligned}
Q_{168} &\xrightarrow{\tau} ((b_1 wf.P_{13} \parallel \underline{P}_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])[\Phi_B] = Q_{169} \\
Q_{168} &\xrightarrow{\tau} ((\underline{P}_{12} \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])[\Phi_B] = Q_{170}
\end{aligned}$$

$$\begin{aligned}
Q_{169} &\xrightarrow{\tau} ((P_{13} \parallel \underline{P}_{23}) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])[\Phi_B] = Q_{171} \\
Q_{169} &\xrightarrow{\tau} ((b_1 wf.P_{13} \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])[\Phi_B] = D_{46}
\end{aligned}$$

$$\begin{aligned}
Q_{170} &\xrightarrow{\tau} ((b_1 wf.P_{13} \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])[\Phi_B] = D_{46} \\
Q_{170} &\xrightarrow{\tau} ((\underline{P}_{12} \parallel P_{21}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = Q_{172}
\end{aligned}$$

$$\begin{aligned}
Q_{171} &\xrightarrow{\tau} ((P_{13} \parallel \underline{P}_{23}) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])[\Phi_B] = Q_{171} \\
Q_{171} &\xrightarrow{\tau} ((P_{13} \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])[\Phi_B] = D_{47}
\end{aligned}$$

$$\begin{aligned}
Q_{172} &\xrightarrow{\tau} ((b_1 wf.P_{13} \parallel P_{21}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = S_{37} \\
Q_{172} &\xrightarrow{\tau} ((\underline{P}_{12} \parallel P_{22}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = Q_{173}
\end{aligned}$$

$$\begin{aligned}
Q_{173} &\xrightarrow{\tau} ((b_1 wf.P_{13} \parallel P_{22}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = S_{26} \\
Q_{173} &\xrightarrow{\tau} ((\underline{P}_{12} \parallel P_{21}) \parallel_B [\text{PV}(\text{true}, \text{true}, 2)])[\Phi_B] = Q_{172}
\end{aligned}$$

$$\mathbf{D}_{44} \xrightarrow{\mathbf{1}} ((\underline{b_1 wf}.P_{13} \parallel \underline{\mathbf{P}}_{23}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])[\Phi_B] = \mathbf{Q}_{174}$$

$$\begin{aligned}
Q_{174} &\xrightarrow{\tau} ((P_{13} \parallel \underline{P}_{23}) \parallel_B [\text{PV}(\text{false}, \text{false}, 2)])[\Phi_B] = Q_{171} \\
Q_{174} &\xrightarrow{\tau} ((\underline{b_1 wf}.P_{13} \parallel b_2 wt.P_{21}) \parallel_B [\text{PV}(\text{true}, \text{false}, 2)])[\Phi_B] = Q_{175}
\end{aligned}$$

$$\begin{aligned}
Q_{175} &\xrightarrow{\tau} ((P_{13} \parallel b_2wt.P_{21}) \parallel_B [PV(true, false, 2)])[\Phi_B] = D_{47} \\
Q_{175} &\xrightarrow{\tau} ((\underline{b_1wf}.P_{13} \parallel P_{21}) \parallel_B [PV(true, true, 2)])[\Phi_B] = Q_{176} \\
\\
Q_{176} &\xrightarrow{\tau} ((P_{13} \parallel P_{21}) \parallel_B [PV(false, true, 2)])[\Phi_B] = S_{38} \\
Q_{176} &\xrightarrow{\tau} ((\underline{b_1wf}.P_{13} \parallel P_{22}) \parallel_B [PV(true, true, 2)])[\Phi_B] = Q_{177} \\
\\
Q_{177} &\xrightarrow{\tau} ((P_{13} \parallel P_{22}) \parallel_B [PV(false, true, 2)])[\Phi_B] = S_{39} \\
Q_{177} &\xrightarrow{\tau} ((\underline{b_1wf}.P_{13} \parallel P_{21}) \parallel_B [PV(true, true, 2)])[\Phi_B] = Q_{176} \\
\\
\mathbf{D_{45}} &\xrightarrow{\mathbf{1}} ((\underline{\mathbf{P}}_{13} \parallel \underline{\mathbf{P}}_{23}) \parallel_{\mathbf{B}} [PV(false, false, \mathbf{2})])[\Phi_{\mathbf{B}}] = \mathbf{Q_{178}} \\
\\
Q_{178} &\xrightarrow{\tau} ((P_{13} \parallel \underline{P}_{23}) \parallel_B [PV(false, false, 2)])[\Phi_B] = Q_{171} \\
\\
Q_{178} &\xrightarrow{\tau} ((\underline{P}_{13} \parallel b_2wt.P_{21}) \parallel_B [PV(false, false, 2)])[\Phi_B] = Q_{179} \\
\\
Q_{179} &\xrightarrow{\tau} ((P_{13} \parallel b_2wt.P_{21}) \parallel_B [PV(false, false, 2)])[\Phi_B] = D_{47} \\
Q_{179} &\xrightarrow{\tau} ((\underline{P}_{13} \parallel P_{21}) \parallel_B [PV(false, true, 2)])[\Phi_B] = Q_{180} \\
\\
Q_{180} &\xrightarrow{\tau} ((P_{13} \parallel P_{21}) \parallel_B [PV(false, true, 2)])[\Phi_B] = S_{38} \\
Q_{180} &\xrightarrow{\tau} ((\underline{P}_{13} \parallel P_{24}) \parallel_B [PV(false, true, 2)])[\Phi_B] = Q_{181} \\
\\
Q_{181} &\xrightarrow{\tau} ((P_{13} \parallel P_{24}) \parallel_B [PV(false, true, 2)])[\Phi_B] = S_{27} \\
Q_{181} &\xrightarrow{\text{cs}_2} ((\underline{P}_{13} \parallel kw1.b_2wf.P_2) \parallel_B [PV(false, true, 2)])[\Phi_B] = Q_{182} \\
\\
Q_{182} &\xrightarrow{\tau} ((P_{13} \parallel kw1.b_2wf.P_2) \parallel_B [PV(false, true, 2)])[\Phi_B] = S_{31} \\
Q_{182} &\xrightarrow{\tau} ((\underline{P}_{13} \parallel b_2wf.P_2) \parallel_B [PV(false, true, 1)])[\Phi_B] = Q_{183} \\
\\
Q_{183} &\xrightarrow{\tau} ((b_1wt.P_{11} \parallel b_2wf.P_2) \parallel_B [PV(false, true, 1)])[\Phi_B] = S_{16} \\
Q_{183} &\xrightarrow{\tau} ((\underline{P}_{13} \parallel P_2) \parallel_B [PV(false, false, 1)])[\Phi_B] = Q_{184} \\
\\
Q_{184} &\xrightarrow{\tau} ((b_1wt.P_{11} \parallel P_2) \parallel_B [PV(false, false, 1)])[\Phi_B] = D_2 \\
Q_{184} &\xrightarrow{\text{req}_2} ((\underline{P}_{13} \parallel b_2wt.P_{21}) \parallel_B [PV(false, false, 1)])[\Phi_B] = Q_{185} \\
Q_{184} &\xrightarrow{\tau} ((\underline{P}_{13} \parallel P_2) \parallel_B [PV(false, false, 1)])[\Phi_B] = Q_{184} \\
\\
Q_{185} &\xrightarrow{\tau} ((b_1wt.P_{11} \parallel b_2wt.P_{21}) \parallel_B [PV(false, false, 1)])[\Phi_B] = D_1 \\
Q_{185} &\xrightarrow{\tau} ((\underline{P}_{13} \parallel P_{21}) \parallel_B [PV(false, true, 1)])[\Phi_B] = Q_{186} \\
\\
Q_{186} &\xrightarrow{\tau} ((b_1wt.P_{11} \parallel P_{21}) \parallel_B [PV(false, true, 1)])[\Phi_B] = D_6 \\
Q_{186} &\xrightarrow{\tau} ((\underline{P}_{13} \parallel P_{24}) \parallel_B [PV(false, true, 1)])[\Phi_B] = Q_{187} \\
\\
Q_{187} &\xrightarrow{\tau} ((b_1wt.P_{11} \parallel P_{24}) \parallel_B [PV(false, true, 1)])[\Phi_B] = D_{10} \\
Q_{187} &\xrightarrow{\text{cs}_2} ((\underline{P}_{13} \parallel kw1.b_2wf.P_2) \parallel_B [PV(false, true, 1)])[\Phi_B] = Q_{188} \\
\\
Q_{188} &\xrightarrow{\tau} ((b_1wt.P_{11} \parallel kw1.b_2wf.P_2) \parallel_B [PV(false, true, 1)])[\Phi_B] = D_{14} \\
Q_{188} &\xrightarrow{\tau} ((\underline{P}_{13} \parallel b_2wf.P_2) \parallel_B [PV(false, true, 1)])[\Phi_B] = Q_{183}
\end{aligned}$$

$$\mathbf{D}_{46} \xrightarrow{1} ((\underline{b_1 wf}.P_{13} \parallel \underline{b_2 wt}.P_{21}) \parallel_B [\mathbf{PV}(true, false, \mathbf{2})])[\Phi_B] = \mathbf{Q}_{189}$$

$$Q_{189} \xrightarrow{\tau} ((P_{13} \parallel \underline{b_2 wt}.P_{21}) \parallel_B [\mathbf{PV}(false, false, 2)])[\Phi_B] = Q_{190}$$

$$Q_{189} \xrightarrow{cs_2} ((\underline{b_1 wf}.P_{13} \parallel P_{21}) \parallel_B [\mathbf{PV}(true, true, 2)])[\Phi_B] = Q_{176}$$

$$Q_{190} \xrightarrow{\tau} ((P_{13} \parallel \underline{b_2 wt}.P_{21}) \parallel_B [\mathbf{PV}(false, false, 2)])[\Phi_B] = Q_{190}$$

$$Q_{190} \xrightarrow{\tau} ((P_{13} \parallel P_{21}) \parallel_B [\mathbf{PV}(false, true, 2)])[\Phi_B] = S_{38}$$

$$\mathbf{D}_{47} \xrightarrow{1} ((\underline{P}_{13} \parallel \underline{b_2 wt}.P_{21}) \parallel_B [\mathbf{PV}(false, false, \mathbf{2})])[\Phi_B] = \mathbf{Q}_{191}$$

$$Q_{191} \xrightarrow{\tau} ((P_{13} \parallel \underline{b_2 wt}.P_{21}) \parallel_B [\mathbf{PV}(false, false, 2)])[\Phi_B] = Q_{190}$$

$$Q_{191} \xrightarrow{\tau} ((\underline{P}_{13} \parallel P_{21}) \parallel_B [\mathbf{PV}(false, true, 2)])[\Phi_B] = Q_{180}$$

B All Fair-Reachable Processes and Their Symmetric Processes are Making Progress

This appendix is devoted to proving Proposition 5.21. It states that all processes in \mathbf{D} and hence, by Proposition 5.20-2, that all processes in $\mathbf{D} \cup \mathbf{S}$ are progressing. We need the following observation:

Proposition B.1 $D_y \triangleright \mathbf{cs}_i$ if for any f-step from D_y of the form $D_y \xrightarrow{1} R \xrightarrow{v} S \in \tilde{\mathbb{P}}_{\ell_1}$ we have either $\mathbf{cs}_i \in v$ or $S \triangleright \mathbf{cs}_i$.

The transition system describing all f-steps from a given D_y can be built according to the process definitions and transitions between processes given in Appendix A. To increase readability, we have omitted τ -actions in the transition labels.

Iterative application of Proposition B.1 allows us to state which processes can perform specific actions. Since most processes can be dealt with in a similar way, we just report the proofs of a few cases. In order to trace which process has to guarantee the execution of what action, we use $D_y \langle i \rangle$ ($D_y \langle 1,2 \rangle$) as a shorthand of $D_y \in \mathbf{D}_i$ ($D_y \in \mathbf{D}_{1,2}$, respectively). Furthermore, again to increase readability, having already proven that for a given $D_y \in \mathbf{D}_{1,2}$ we have $D_y \triangleright \mathbf{cs}_1$ ($D_y \triangleright \mathbf{cs}_2$), we write $D_y \langle \mathcal{Y}, 2 \rangle$ ($D_y \langle 1, 2 \rangle$, respectively).

Proposition B.2

1. $\{D_8 \langle 1,2 \rangle, D_9 \langle 1 \rangle, D_{19} \langle 1,2 \rangle, D_{27} \langle 1,2 \rangle\} \triangleright \mathbf{cs}_1$;
2. $\{D_{10} \langle 1,2 \rangle, D_{11} \langle 2 \rangle, D_{20} \langle 1,2 \rangle\} \triangleright \mathbf{cs}_2$.

Proof:

Figure 1 shows that each f-step from D_8 contains the action \mathbf{cs}_1 (i.e. for each step of the form $D_8 \xrightarrow{1} Q_{52} \xrightarrow{v} D_y$ it is $\mathbf{cs}_1 \in v$); observe that the τ -loop at Q_{56} is not harmful: in an f-step (i.e. due to fairness) this loop will be exited. Thus, by Proposition B.1 we have that $D_8 \triangleright \mathbf{cs}_1$; similarly we have that $D_{10} \triangleright \mathbf{cs}_2$ (see Figure 3). \square

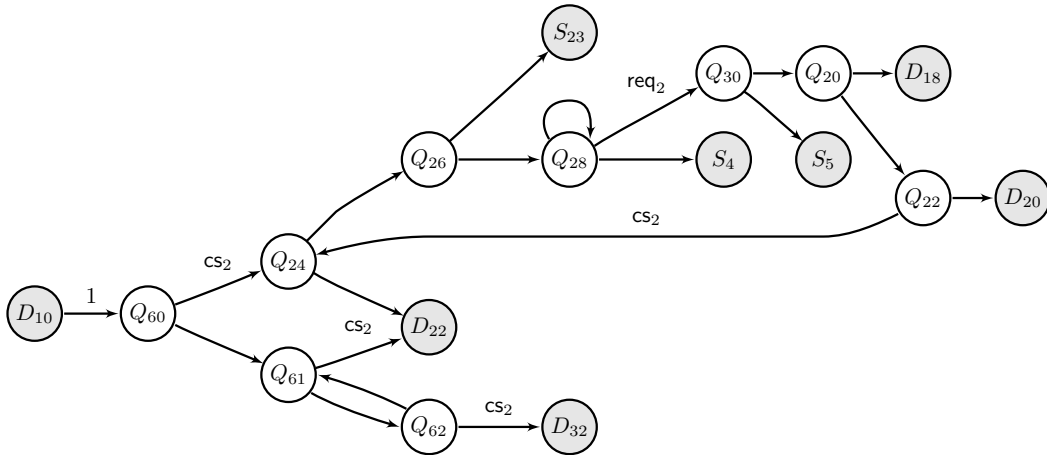


Figure 3: Fair-Steps of D_{10}

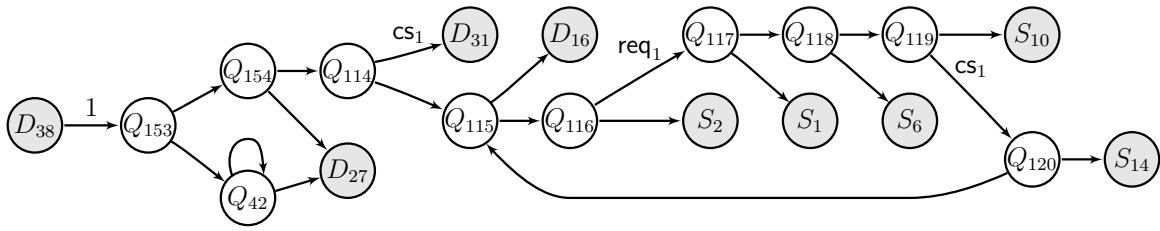


Figure 4: Fair-Steps of D_{38}

Proposition B.3

1. $D_{38} \langle 1,2 \rangle \triangleright \text{cs}_1$;
2. $D_{39} \langle 1,2 \rangle \triangleright \text{cs}_1$;
3. $\{D_{26} \langle 1,2 \rangle, D_{37} \langle 1,2 \rangle\} \triangleright \text{cs}_1$;
4. $\{D_{25} \langle 1,2 \rangle, D_{28} \langle 1,2 \rangle\} \triangleright \text{cs}_1$;
5. $\{D_{18} \langle 1,2 \rangle, D_{24} \langle 1,2 \rangle\} \triangleright \text{cs}_1$;
6. $\{D_4 \langle 1,2 \rangle, D_5 \langle 1 \rangle\} \triangleright \text{cs}_1$.

Proof: Let us prove Items 1, 3 and 6.

1. By Figure 4, $D_{38} \xrightarrow{1} Q \xrightarrow{v} S \in \tilde{\mathbb{P}}_{\ell 1}$ implies either $v \in \tau^*$ and $S = D_{27} \triangleright \text{cs}_1$ (by Proposition B.2-1) or $\text{cs}_1 \in v = \tau \text{cs}_1 v'$. By Proposition B.1 we have $D_{38} \triangleright \text{cs}_2$.
3. We prove the statement only for D_{26} . (The proof for D_{37} is similar). $D_{26} \xrightarrow{1} Q \xrightarrow{v} S \in \tilde{\mathbb{P}}_{\ell 1}$ and Figure 5 implies $v \in \tau^*$ and $S \in \{D_{38}, D_{39}\} \triangleright \text{cs}_1$ (by Items 1 and 2). Thus we have that $D_{26} \triangleright \text{cs}_1$.
6. As in the above cases (see Figures 6 and 2).

□

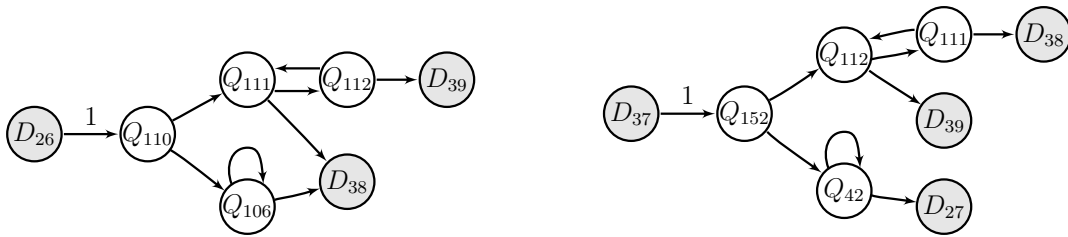


Figure 5: Fair-Steps of D_{26} and D_{37}

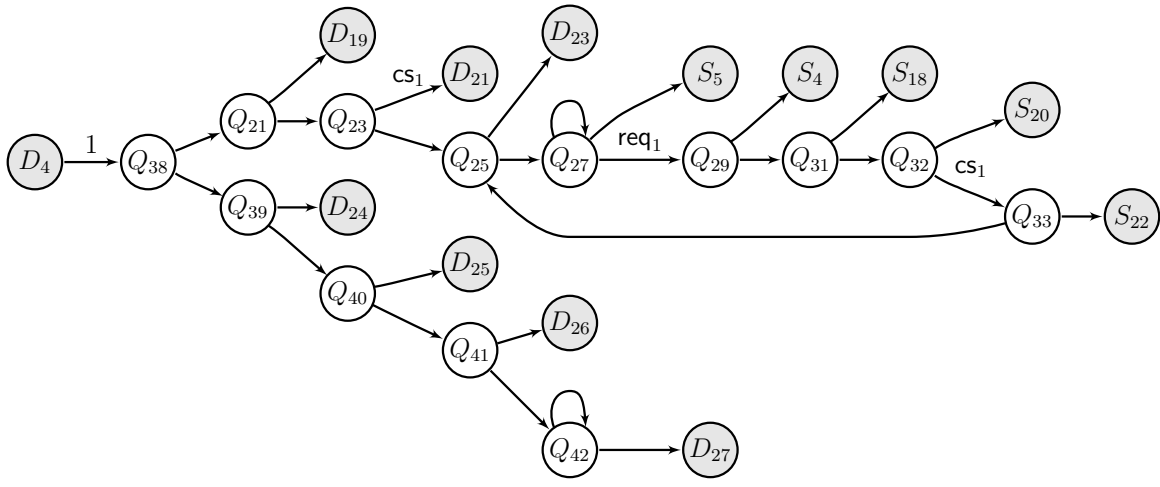


Figure 6: Fair-Steps of D_4

Proposition B.4

1. $\{D_{36} \langle 2 \rangle, D_{46} \langle 1,2 \rangle, D_{47} \langle 1,2 \rangle\} \triangleright \text{cs}_2$;
2. $\{D_{23} \langle 2 \rangle, D_{33} \langle 2 \rangle, D_{43} \langle 1,2 \rangle, D_{44} \langle 1,2 \rangle, D_{45} \langle 1,2 \rangle\} \triangleright \text{cs}_2$;
3. $\{D_{22} \langle 1 \rangle, D_{32} \langle 1 \rangle\} \triangleright \text{cs}_1$;
4. $D_{20} \langle 1,2 \rangle \triangleright \text{cs}_1$;
5. $\{D_1 \langle 1,2 \rangle, D_2 \langle 1 \rangle, D_6 \langle 1,2 \rangle, D_{10} \langle 1,2 \rangle, D_{14} \langle 1 \rangle\} \triangleright \text{cs}_1$;
6. $D_{16} \langle 2 \rangle \triangleright \text{cs}_2$;
7. $\{D_{35} \langle 2 \rangle, D_{40} \langle 2 \rangle, D_{41} \langle 1,2 \rangle, D_{42} \langle 1,2 \rangle\} \triangleright \text{cs}_2$;
8. $\{D_{30} \langle 2 \rangle, D_{31} \langle 2 \rangle, D_{34} \langle 2 \rangle\} \triangleright \text{cs}_2$;
9. $\{D_{12} \langle 2 \rangle, D_{21} \langle 2 \rangle, D_{29} \langle 2 \rangle\} \triangleright \text{cs}_2$.

Proof:

1. Figure 7 shows that for each f-step $D_{36} \xrightarrow{1} Q \xrightarrow{v} S \in \tilde{\mathbb{P}}_{\ell_1}$ it is either $\text{cs}_2 \in v$ or $S \in \{S_4, S_5, S_8, S_9, S_{18}, S_{28}, S_{37}, S_{38}\}$, see Figure 7. Moreover, by Propositions B.3-6, B.2-2 and B.3-(5, 4, 3, 1), we have that $\{D_4, D_5, D_8, D_9, D_{18}, D_{28}, D_{37}, D_{38}\} \triangleright \text{cs}_1$ and hence $S \triangleright \text{cs}_2$ (see Proposition 5.20-1). Again by Proposition B.1, we can conclude that $D_{36} \triangleright \text{cs}_2$.
2. $D_{23} \xrightarrow{1} Q \xrightarrow{v} S \in \tilde{\mathbb{P}}_{\ell_1}$ implies, by Figure 8, $S \in \{S_5, S_8, S_9, S_{24}, S_{25}, S_{26}, S_{27}, D_{36}\}$. As in the previous item, $\{D_5, D_8, D_9, D_{24}, D_{25}, D_{26}, D_{27}\} \triangleright \text{cs}_1$ and $D_{36} \triangleright \text{cs}_2$ imply $S \triangleright \text{cs}_2$. Also in this case we are done.

□

As a trivial consequence of the above results we can state the following lemma.

Lemma B.5 All processes in D_1 are making progress.

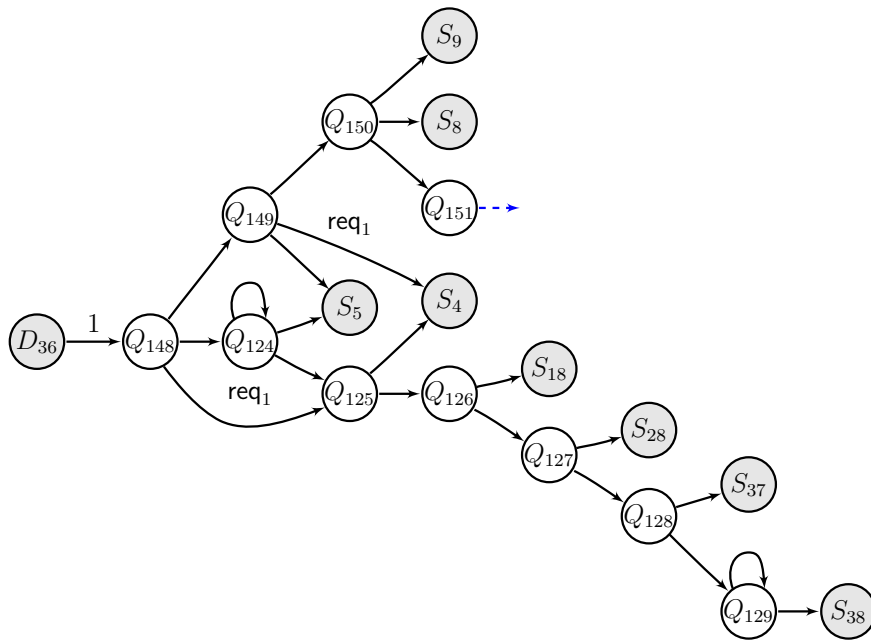


Figure 7: Fair-Steps of D_{36}

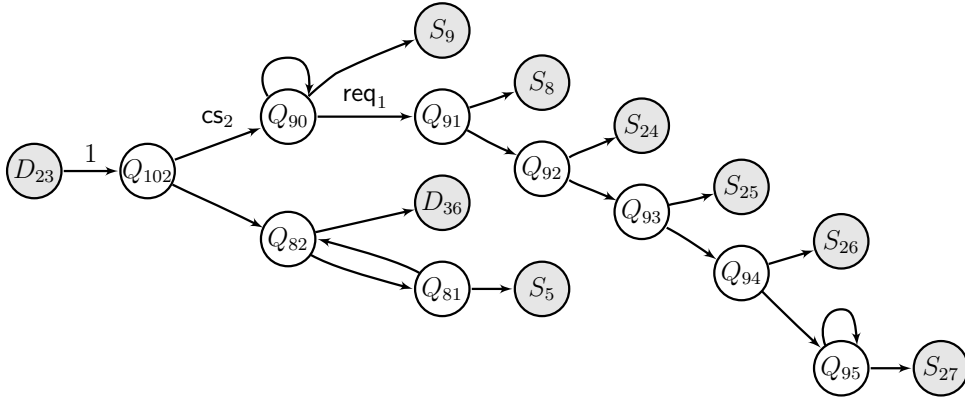


Figure 8: Fair-Steps of D_{23}

In order to state a similar result for processes in D_2 , it remains to check that $D_3 \triangleright cs_2$ and $D_7 \triangleright cs_2$. Moreover, at this stage, we have only proved that processes in $D_{1,2}$ (except for D_{10} and D_{20}) imply the execution of just one of the actions cs_1 and cs_2 . We also need to prove the following proposition.

Proposition B.6

1. $\{D_8 \langle \mathcal{Y}, 2 \rangle, D_{19} \langle \mathcal{Y}, 2 \rangle, D_{27} \langle \mathcal{Y}, 2 \rangle\} \triangleright cs_2$;
2. $D_{38} \langle \mathcal{Y}, 2 \rangle \triangleright cs_2$;
3. $D_{39} \langle \mathcal{Y}, 2 \rangle \triangleright cs_2$;
4. $D_{47} \langle 1, \mathcal{Z} \rangle \triangleright cs_1$;
5. $\{D_{26} \langle \mathcal{Y}, 2 \rangle, D_{37} \langle \mathcal{Y}, 2 \rangle\} \triangleright cs_2$;
6. $\{D_{44} \langle 1, \mathcal{Z} \rangle, D_{45} \langle 1, \mathcal{Z} \rangle, D_{46} \langle 1, \mathcal{Z} \rangle\} \triangleright cs_1$;

7. $\{D_{25} \langle \mathcal{Y}, 2 \rangle, D_{28} \langle \mathcal{Y}, 2 \rangle\} \triangleright \mathbf{cs}_2$;
8. $D_{43} \langle 1, \mathcal{Z} \rangle \triangleright \mathbf{cs}_1$;
9. $\{D_6 \langle \mathcal{Y}, 2 \rangle, D_7 \langle 2 \rangle, D_{18} \langle \mathcal{Y}, 2 \rangle, D_{24} \langle \mathcal{Y}, 2 \rangle\} \triangleright \mathbf{cs}_2$;
10. $D_{42} \langle 1, \mathcal{Z} \rangle \triangleright \mathbf{cs}_1$;
11. $\{D_1 \langle \mathcal{Y}, 2 \rangle, D_3 \langle 2 \rangle, D_4 \langle \mathcal{Y}, 2 \rangle\} \triangleright \mathbf{cs}_2$;
12. $D_{41} \langle 1, \mathcal{Z} \rangle \triangleright \mathbf{cs}_1$.

Proof:

1. $D_8 \xrightarrow{1} Q \xrightarrow{v} S \in \tilde{\mathbb{P}}_{\ell_1}$ implies $S \in \{D_{23}, D_{21}, D_{29}, D_{30}, D_{31}\} \cup \{S_4, S_5, S_{18}, S_{20}, S_{22}\}$ (see Figure 1). As above, $\{D_{23}, D_{21}, D_{29}, D_{30}, D_{31}\} \triangleright \mathbf{cs}_2$ and $\{D_4, D_5, D_{18}, D_{20}, D_{22}\} \triangleright \mathbf{cs}_1$ imply $S \triangleright \mathbf{cs}_2$ and, hence, $D_8 \triangleright \mathbf{cs}_2$.
5. We prove the statement only for D_{26} . The proof for D_{37} is similar. $D_{26} \xrightarrow{1} Q \xrightarrow{v} S \in \tilde{\mathbb{P}}_{\ell_1}$ and Figure 5 implies $v \in \tau^*$ and $S \in \{D_{38}, D_{39}\} \triangleright \mathbf{cs}_2$ (by Items 1 and 2). Thus we have that $D_{26} \triangleright \mathbf{cs}_2$.

□

Lemma B.7 All processes in \mathbb{D}_2 and in $\mathbb{D}_{1,2}$ are making progress.

Finally, Proposition 5.21 easily follows from Lemmas B.5 and B.7.