Progress, Fairness and Justness in Process Algebra

Rob van Glabbeek[1,2] and Peter Höfner[1,2]

[1] NICTA*, Sydney, Australia
[2] Computer Science and Engineering, UNSW, Sydney, Australia

Abstract. To prove liveness properties of concurrent systems, it is often necessary to postulate progress, fairness and justness properties. This paper investigates how the necessary progress, fairness and justness assumptions can be added to or incorporated in a standard process-algebraic specification formalism. We propose a formalisation that can be applied to a wide range of process algebras. The presented formalism is used to reason about route discovery and packet delivery in the setting of wireless networks.

1 Introduction

In a process-algebraic setting, safety properties of concurrent systems are usually shown by the use of invariants on a labelled transition system (LTS). This does not require any assumptions about the behaviour of concurrent systems beyond their modelling as states in an LTS. In order to prove liveness properties on the other hand it is usually necessary to postulate certain progress, fairness and justness properties as part of the specification of the systems under investigation. This paper investigates how the necessary progress, fairness and justness properties can be added to or incorporated in a standard process-algebraic specification formalism. Liveness properties are formalised in terms of a temporal logic interpreted on complete paths in the LTS of the process algebra. Progress, fairness and justness properties are captured by fine-tuning the definition of what constitutes a complete path.

Section 2 introduces an Algebra of Broadcast Communication (ABC)—a variant of the process algebra CBS [20]—that is essentially CCS [16] augmented with a formalism for broadcast communication. ABC is given a structural operational semantics [18] that interprets expressions as states in an LTS. We develop our approach for formalising liveness properties as well as progress, fairness and justness assumptions in terms of this process algebra. However, the presented approach can be applied to a wide range of process algebras. ABC is largely designed to be a convenient starting point for transferring the presented theory to such algebras; it contains all the features for which we are aware that the

* NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

application of our theory poses non-trivial problems, and, at the same time, is kept as simple as possible. In [8] we apply the same approach to a more involved process algebra called AWN (Algebra for Wireless Networks).

Section 3 recalls Linear-time Temporal Logic (LTL) [19] and describes a way to interpret it on a labelled transition system that arises as the semantic interpretation of a process algebra like ABC. This yields a way to represent desirable properties of concurrent systems specified in such a process algebra by means of LTL properties. We illustrate this by formulating *packet delivery*, a liveness property studied in [8] in the context of wireless mesh network protocols. The presented development applies just as well to desirable properties of concurrent systems specified in branching time temporal logics such as CTL and CTL*.

In Section 4.1 we formulate an elementary *progress assumption* on the behaviour of processes, without which no useful liveness property of a system will hold. In the standard interpretation of temporal logic [19,7] a stronger progress assumption is built in, but we argue that this stronger version is not a valid assumption in the context of reactive systems. In order to derive a progress assumption that is both necessary and justifiable in the reactive context we envision, we introduce the concept of an *output action*, which cannot be blocked by the context in which a process is running. Although output actions are commonplace in many specification formalisms, their use in process algebra is limited at best, and we have not seen them used to define progress properties. The main reason for working with a language that is richer than CCS, is that restricted to CCS the set of output actions would be empty.

In Section 4.2 we discuss *weak* and *strong fairness assumptions* and propose a formalisation in the context of process algebras like ABC by augmenting a process-algebraic specification P with a *fairness specification*, which is given as a collection of temporal logic formulas. This follows the traditional approach of TLA [13] and other formalisms [9], "in which first the legal computations are specified, and then a fairness notion is used to exclude some computations which otherwise would be legal" [1]. However, in order to do justice to the reactive nature of the systems under consideration, we need a more involved consistency requirement between the process-algebraic specification of a system and its fairness specification.

In Section 4.3 we propose a *justness assumption* for parallel-composed transition systems, essentially assuming progress of all the component processes. In the literature, such justness properties are typically seen as special cases of weak fairness properties, and the term *justice* is often used as a synonym for *weak fairness*. Here we consider justness to be a notion distinct from fairness, and propose a completely different formalisation. Fairness is a property of schedulers that repeatedly choose between tasks, whereas justness is a property of parallel-composed transition systems. Nevertheless, we show that our notion of justness coincides with the original notion of justice of [14]—a weak fairness property. This requires an interpretation of the work of [14] applied to LTSs involving a more precise definition—and decision—of what it means for a transition to be continuously enabled.

Table 1. Structural operational semantics of ABC

$$\alpha.P \xrightarrow{\alpha} P \ (\textsc{Act}) \qquad \frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'} \ (\textsc{Sum-l}) \qquad \frac{Q \xrightarrow{\alpha} Q'}{P+Q \xrightarrow{\alpha} Q'} \ (\textsc{Sum-r})$$

$$\frac{P \xrightarrow{\eta} P'}{P|Q \xrightarrow{\eta} P'|Q} \ (\textsc{Par-l}) \qquad \frac{P \xrightarrow{c} P',\ Q \xrightarrow{\bar{c}} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \ (\textsc{Comm}) \qquad \frac{Q \xrightarrow{\eta} Q'}{P|Q \xrightarrow{\eta} P|Q'} \ (\textsc{Par-r})$$

$$\frac{P \xrightarrow{b\sharp_1} P',\ Q \xrightarrow{b?}\!\!\!\!\!\!\rightarrow}{P|Q \xrightarrow{b\sharp_1} P'|Q} \ (\textsc{Bro-l}) \qquad \frac{P \xrightarrow{b\sharp_1} P',\ Q \xrightarrow{b\sharp_2} Q'}{P|Q \xrightarrow{b\sharp} P'|Q'} \ (\textsc{Bro-c}) \qquad \frac{P \xrightarrow{b?}\!\!\!\!\!\!\rightarrow,\ Q \xrightarrow{b\sharp_2} Q'}{P|Q \xrightarrow{b\sharp_2} P|Q'} \ (\textsc{Bro-r})$$

$\sharp_1 \circ \sharp_2 = \sharp \neq \_$ with

$\circ$	!	?
!	_	!
?	!	?

$$\frac{P \xrightarrow{\ell} P'}{P[f] \xrightarrow{f(\ell)} P'[f]} \ (\textsc{Rel}) \qquad \frac{P \xrightarrow{\ell} P'}{P\backslash c \xrightarrow{\ell} P'\backslash c} \ (c \neq \ell \neq \bar{c}) \ (\textsc{Res}) \qquad \frac{P \xrightarrow{\ell} P'}{A \xrightarrow{\ell} P'} \ (A \stackrel{def}{=} P) \ (\textsc{Rec})$$

Finally, Section 5 addresses the question whether system specifications consisting of a process-algebraic and a fairness specification allow implementations that can be described entirely process-algebraic, i.e., without fairness component. Here an 'implementation' allows the replacement of nondeterministic choices by (more) deterministic choices following a particular scheduling policy. In the context of CCS we conjecture a negative answer by showing an extremely simple fair scheduling specification—given as a CCS expression augmented with a fairness specification—that could not be implemented by any CCS expression alone. This specification does allow an implementation in ABC without fairness component, which takes advantage of justness properties for output actions.

2 ABC—An Algebra of Broadcast Communication

The Algebra of Broadcast Communication (ABC) is parametrised with sets $\mathscr{A}$ of *agent identifiers*, $\mathscr{B}$ of *broadcast names* and $\mathscr{C}$ of *handshake communication names*; each $A \in \mathscr{A}$ comes with a defining equation $A \stackrel{def}{=} P$ with P being a guarded ABC expression as defined below.

The collections $\mathscr{B}!$ and $\mathscr{B}?$ of *broadcast* and *receive* actions are given by $\mathscr{B}\sharp := \{b\sharp \mid b \in \mathscr{B}\}$ for $\sharp \in \{!, ?\}$. The set $\bar{\mathscr{C}}$ of *handshake communication co-names* is $\bar{\mathscr{C}} := \{\bar{c} \mid c \in \mathscr{C}\}$, and the set $\mathscr{H}$ of *handshake actions* is $\mathscr{H} := \mathscr{C} \cup \bar{\mathscr{C}}$, the disjoint union of the names and co-names. The function $\bar{\cdot}$ is extended to $\mathscr{H}$ by declaring $\bar{\bar{c}} = c$.

Finally, $Act := \mathscr{B}! \cup \mathscr{B}? \cup \mathscr{H} \cup \{\tau\}$ is the set of *actions*. Below, A, B, C range over $\mathscr{A}$, b over $\mathscr{B}$, c over $\mathscr{H}$, η over $\mathscr{H} \cup \{\tau\}$ and α, ℓ over Act. A *relabelling* is a function $f : (\mathscr{B} \to \mathscr{B}) \cup (\mathscr{C} \to \mathscr{C})$. It extends to Act by $f(\bar{c}) = \overline{f(c)}$, $f(b\sharp) = f(b)\sharp$ and $f(\tau) := \tau$. The set $\mathrm{Ex}_{\mathrm{ABC}}$ of ABC expressions is the smallest set including:

0	*inaction*	$\alpha.P$	*prefixing*	$P + Q$	*choice*
$P\|Q$	*parallel composition*	$P\backslash c$	*restriction*	$P[f]$	*relabelling*
A	*agent identifier*				

for $P, Q \in \mathrm{Ex}_{\mathrm{ABC}}$ and relabellings f. An expression is guarded if each agent identifier occurs within the scope of a prefixing operator. The semantics of ABC is given by the labelled transition relation $\rightarrow_{\mathrm{ABC}} \subseteq \mathrm{Ex}_{\mathrm{ABC}} \times \mathit{Act} \times \mathrm{Ex}_{\mathrm{ABC}}$, where the transitions $P \xrightarrow{\ell} Q$ are derived from the rules of Table 1.

ABC is basically the Calculus of Communicating Processes (CCS) [16] augmented with a formalism for broadcast communication taken from the Calculus of Broadcasting Systems (CBS) [20]. The syntax without the broadcast and receive actions and all rules except (Bro-l), (Bro-c) and (Bro-r) are taken verbatim from CCS. However, the rules now cover the different name spaces; (Act) for example allows labels of broadcast and receive actions. The rule (Bro-c)—without rules like (Par-l) and (Par-r) with label $b!$—implements a form of broadcast communication where any broadcast $b!$ performed by a component in a parallel composition is guaranteed to be received by any other component that is ready to do so, i.e., in a state that admits a $b?$-transition. In order to ensure associativity of the parallel composition, one also needs this rule for components receiving at the same time ($\sharp_1{=}\sharp_2{=}?$). The rules (Bro-l) and (Bro-r) are added to make broadcast communication *non-blocking*: without them a component could be delayed in performing a broadcast simply because one of the other components is not ready to receive it.

Theorem 2.1. *Strong bisimilarity [16] is a congruence for all operators of ABC.*

Proof. This follows immediately from the observation that all rules of Table 1 are in the GSOS format of Bloom, Istrail & Meyer, using [3, Theorem 5.1.2]. □

To establish the associativity of parallel composition of ABC up to strong bisimilarity ($\leftrightarroweq$), we will employ a general result of Cranen, Mousavi & Reniers [5]. However, for this result to apply, we need a structural operational semantics of the language in the De Simone format [22]—so without negative premises.

To this end, let $\mathscr{B}{:} := \{b{:} \mid b \in \mathscr{B}\}$ be the set of *broadcast discards*, and $\mathscr{L} := \mathscr{B}{:} \uplus \mathit{Act}$ the set of *transition labels*. We enrich the transition relation of ABC with transitions labelled with discard communications, by adding the rules

$$0 \xrightarrow{b:} 0 \ \text{(Dis0)} \qquad \alpha.P \xrightarrow{b:} \alpha.P \ \ (\alpha{\neq}b?) \ \text{(Dis1)} \qquad \frac{P \xrightarrow{b:} P',\ Q \xrightarrow{b:} Q'}{P+Q \xrightarrow{b:} P'+Q'} \ \text{(Dis2)}$$

to Table 1, allowing $\sharp_1{=}\sharp_2{=}\sharp{=}:$ in (Bro-c),[1] and letting ℓ range over all of $\mathscr{L}$.

Lemma 2.2. [20] $P \xrightarrow{b:} Q$ *iff* $Q = P \wedge P \xrightarrow{b?}\!\!\!\!\!\!\rightarrow$, *for* $P, Q \in \mathrm{Ex}_{\mathrm{ABC}}$ *and* $b \in \mathscr{B}$.

Proof. A straightforward induction on derivability of transitions. □

Because of this, a negative premise $P \xrightarrow{b?}\!\!\!\!\!\!\rightarrow$ can be replaced by a positive premise $P \xrightarrow{b:} P'$ and all rules (Bro) in Table 1 can be unified into the single rule (Bro-c), where $\sharp_1, \sharp_2, \sharp$ range over $\{!, ?, :\}$ and $\circ$ is defined by the table on the right. The resulting rules are all in the De Simone format.

$\circ$	!	?	:
!	_	!	!
?	!	?	?
:	!	?	:

[1] The remaining cases are still undefined, i.e, $\sharp_1 \circ : = : \circ \sharp_2 = \_$ (for $\sharp_1, \sharp_2 \neq :$).

Corollary 2.3. *The original and modified structural operational semantics of ABC yield the same labelled transition relation $\rightarrow_{ABC}$ when transitions labelled b: are ignored.*

In fact, our 'modified' operational semantics stems directly from CBS [20].

Theorem 2.4. *In ABC, parallel composition is associative up to $\underline{\leftrightarrow}$.*

Proof. The associativity depends on the generated transition relation only, and is preserved when ignoring transitions with a particular label. So by Corollary 2.3 it suffices to investigate the modified semantics. The modified operational rules of ABC fit the ASSOC-De Simone rule format of [5], which guarantees associativity up to $\underline{\leftrightarrow}$. The detailed proof that our rules fit this format is similar to the proof of Theorem 4.4 in [8]. □

3 Formalising Temporal Properties

We will use Linear-time Temporal Logic (LTL) [19] to specify properties that one would like to establish for concurrent systems. For the purpose of this paper, any other temporal logic could have been used as well.

We briefly recapitulate the syntax and semantics of LTL; a thorough and formal introduction to this logic can be found e.g. in [11]. The logic is built from a set of *atomic propositions* that characterise facts that may hold in some state of a (concurrent) system. A classical example is that a 'transition ν is enabled', denoted by $en(\nu)$.

LTL formulas are interpreted on paths in a transition system,[2] where each state is labelled with the atomic propositions that hold in that state. A *path* π is a finite or infinite sequence of states such there is a transition from each state in π to the next, except the last one if π is finite. An atomic proposition p holds for a path π if p holds in the first state of π.

LTL [19] uses the temporal operators **X**, **G**, **F** and **U**.[3] The formulas $\mathbf{X}\phi$, $\mathbf{G}\phi$ and $\mathbf{F}\phi$ mean that ϕ holds in the second state on a given path, *globally* in all states, and *eventually* in some state, respectively; $\phi\mathbf{U}\psi$ means that ψ will hold eventually, and ϕ holds in all states *until* this happens.[4] Here a formula ϕ is deemed to *hold in a state on a path* π iff it holds for the remainder of π when starting from that state. LTL formulas can be combined by the logical connectives *conjunction* $\wedge$, *disjunction* $\vee$, *implication* $\Rightarrow$ and *negation* $\neg$.

An LTL formula holds for a process, modelled as a state in a transition system iff it holds for all *complete* paths in the system starting from that state.[5] A path is complete iff it leaves no transitions undone without a good reason; in the original work on LTL [15] the complete paths are exactly the infinite ones, but in Section 4 we propose a different concept of completeness: a path will be

[2] A *transition system* is given by a set S of *states* and a set $T \subseteq S \times S$ of *transitions*.
[3] **X** and **U** were not introduced in the original paper [19]; they were added later on.
[4] **G** and **F** can be expressed in terms of **U**: $\mathbf{F}\phi \equiv \mathtt{true}\mathbf{U}\phi$ and $\mathbf{G}\phi \equiv \neg\mathbf{F}\neg\phi$.
[5] A path staring from a state s is also called a path *of* s.

considered complete iff it is *progressing*, *fair* and *just*, as defined in Sections 4.1, 4.2 and 4.3, respectively.[6]

Below we will apply LTL to the labelled transition system $\mathcal{T}$ generated by the structural operational semantics of ABC.[7] Here, the most natural atomic propositions are the transition labels: they tell when an action takes place. These propositions hold for transitions rather than for states. Additionally, one can consider state-based propositions such as $en(\nu)$. In languages that maintain data variables,[8] propositions such as '$x \leq 7$' that report on the current value of such variables can also be associated to the states.

To incorporate the transition-based atomic propositions into the framework of temporal logic, we perform a translation of the transition-labelled transition system $\mathcal{T}$ into a state-labelled transition system $\mathcal{S}$, and apply LTL to the latter. A suitable translation, proposed in [6], introduces new states halfway the existing transitions—thereby splitting a transition ℓ into $\ell;\tau$—and attaches transition labels to the new 'midway' states. If we also have state-based atomic propositions stemming from $\mathcal{T}$, we furthermore declare any atomic proposition except $en(\nu)$ that holds in state Q to also hold for the new state midway a transition $P \xrightarrow{\ell} Q$. LTL formulas are interpreted on the paths in $\mathcal{S}$. Such a path is a sequence of states of $\mathcal{S}$, and thus an alternating sequence of states and transitions from $\mathcal{T}$. Here we will only consider paths that are infinite or end in a state of $\mathcal{T}$; paths ending 'midway a transition' will not be complete, progressing, fair or just.

Below we use LTL to formalise properties that say that whenever a precondition ϕ^{pre} holds in a reachable state, the system will eventually reach a state satisfying the postcondition ϕ^{post}. Such a property is called an *eventuality property* in [19]; it is formalised by the LTL formula $\mathbf{G}\big(\phi^{pre} \Rightarrow \mathbf{F}\phi^{post}\big)$.

Example 3.1. In a language like ABC we can model a network by means of a parallel composition of processes running on the nodes in the network. Each of those processes A_i, for $1 \leq i \leq n$, could be specified by a defining equation $A_i \stackrel{def}{=} P_i$, where P_i always ends with a recursive call to A_i. This way, the behaviour specified by P_i is repeated forever. The processes A_i send messages to each other along shared channels. Here a message m transmitted along a broadcast or handshake channel is modelled by a name $c_m \in \mathscr{B}$ or $c_m \in \mathscr{C}$.

Suppose the process A_0 can receive messages $m \in \{1, ..., k\}$ from the environment. This could be modelled by $A_0 \stackrel{def}{=} c_1.P_0^1 + \cdots + c_k.P_0^k$. The behaviour of the nodes in the network could be specified so as to guarantee that such a message will eventually reach the node running the process A_n, which will deliver it to the environment by performing the broadcast $d_m!$. We may assume that no other nodes can perform the actions c_m or $d_m!$.

[6] We declare a formula $\mathbf{X}\phi$ false on any path that lacks a second state.

[7] A *labelled transition system* (LTS) is given by a set S of states and a transition relation $T \subseteq S \times \mathscr{L} \times S$ for some set of labels $\mathscr{L}$. The LTS generated by ABC has $S := \mathrm{Ex}_{\mathrm{ABC}}$ and $T := \rightarrow_{\mathrm{ABC}}$.

[8] such as the algebra for wireless networks AWN [8]; see below

A useful property that this network should have is *packet delivery*: any message received from the environment by A_0 will eventually be delivered back to the environment by A_n. In LTL it can be formulated as $\mathbf{G}(c_m \Rightarrow \mathbf{F}d_m!)$.

In [8] we model a routing protocol in a process algebra for wireless networks (AWN) that captures dynamic topologies, where nodes drift in and out of transmission range, and communication between two nodes is successful only when they are within transmission range of each other. In this context a packet delivery property is formulated that can be obtained from a property like the one above by incorporating a number of side conditions.

4 Progress, Fairness and Justness

In [8, Sect. 9], as well as above, we formalise properties that say that under certain conditions some desired activity will eventually happen, or some desired state will eventually be reached. As a simple instance consider the transition systems in Figures 1(a)–(c), where the double-circled state satisfies a desired property ϕ. The formula $\mathbf{G}(a \Rightarrow \mathbf{F}\phi)$ says that once the action a occurs, eventually we will reach a state where ϕ holds. We investigate reasons why this formula might not hold, and formulate assumptions that guarantee it does.

4.1 Progress

The first thing that can go wrong is that the process of Figure 1(a)[9] performs a, thereby reaching the state s, and subsequently remains in the state s without ever performing the internal action τ that leads to the desired state t, satisfying ϕ. If there is the possibility of remaining in a state even when there are enabled internal actions, no useful liveness property about processes will ever be guaranteed. We therefore make an assumption that rules out this type of behaviour.

> *A process in a state that admits an internal action*[10] *will eventually perform an action.* (P_1)

(P_1) is called a *progress* property. It guarantees that the process depicted in Figure 1(a) satisfies the LTL formula $\mathbf{G}(a \Rightarrow \mathbf{F}\phi)$. We cannot assume progress when only external actions are possible. For instance, the process of Figure 1(a) will not necessarily perform the action a, and hence need not satisfy the formula $\mathbf{F}\phi$. The reason is that external actions could be synchronisations with the environment, and the environment may not be ready to synchronise. In ABC this can happen if a is a handshake action $c \in \mathscr{H}$ or a receive action $b? \in \mathscr{B}?$. Here it makes sense to distinguish two kinds of external actions: those whose execution requires cooperation from the environment in which the process runs, and those

[9] Following the approach of CCS [16] we identify processes and states, and do not use a notion of an initial state. When speaking of a process depicted graphically, by default we mean the state indicted by the short arrow in the figure.

[10] ABC offers only one internal action τ. Any other action is called *external*.

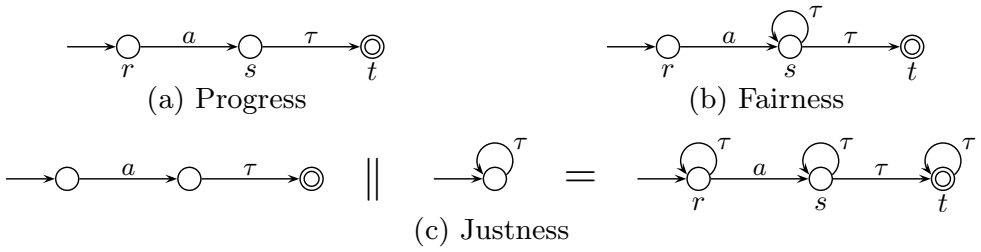

Fig. 1. Progress, Fairness and Justness

that do not. We call the latter kind *output actions*. As far as progress properties go, output actions can be treated just like internal actions:

> *A process in a state that admits an output action will eventually perform an action.* (P_2)

In case a is an output action, which can happen independent of the environment, the formula $\mathbf{F}\phi$ holds for the process of Figure 1(a). In the remainder we treat internal actions and output actions together; we call them *non-blocking* actions.

We formalise (P_1) and (P_2) through a suitable modification of the definition of a complete path. In early work on temporal logic, formulas were interpreted on Kripke structures: transition systems with unlabelled transitions, subject to the condition of *totality*, saying that each state admits at least one outgoing transition. In this context, the complete paths are defined to be all infinite paths of the transition system. When giving up totality, it is customary to deem complete also those paths that end in a state from which no further transitions are possible [6]. Here, we go a step further, and consider paths that are either infinite or end in a state from which no further non-blocking actions are possible. Those paths are called *progressing*. This definition exactly captures (P_1) and (P_2).

This proposal is a middle ground between two extremes. Dropping all progress properties amounts to defining *each* path to be complete. This yields a temporal logic that is not powerful enough to establish nontrivial eventuality properties. Defining a path to be complete only when it cannot be further extended, on the other hand, incorporates progress properties that do not hold for reactive systems. It would give rise to the unwarranted conclusion that the property $\mathbf{F}\phi$ holds for the process of Figure 1(a), regardless of the nature of a.

As we will show, progressing paths do not capture fairness and justness properties; hence they should not be called complete. In Sections 4.2 and 4.3 we will propose a notion of a complete path that is progressing and also captures such properties. This restriction concerns the infinite paths only; for finite paths complete will coincide with progressing.

It remains to decide which of the external actions generated by the structural operational semantics of a language should be classified as output actions. Some actions cannot be output, since they can be blocked by the environment. In ABC, any handshake action $c \in \mathscr{H}$ can be blocked by restriction. Since $(c.0)\backslash c$ cannot perform any action, c cannot be output. For the remaining external actions, the user can decide whether they are output or not. For ABC we decide that a broadcast ($b!$) is an output action, whereas a receive ($b?$) is not. Informal

intuition explains the reason: a process that broadcasts a message should be able to perform this action independent of whether other processes are receiving it. On the other hand, a process should only be able to receive a message that *is* sent by another process.

The above analysis assumes the original operational semantics of ABC, with negative premises. For the modified semantics with discard-transitions, the question arises whether b: should count as an output action. Here the right answer is that b: is a transition label that does not count as an action at all. The reason is that we want our progress property (P_2) to imply that the ABC process $b_1!.b_2!.0$ will eventually execute the broadcast action b_2. However, a potentially complete path that invalidates this property consists of $b_1!$ followed by infinitely many broadcast discards b_1: (or b_2:). In the original operational semantics such a path does not exist, and the property is satisfied. To obtain the same result in the modified operational semantics, we classify b: as a non-action. This way, (P_2) says that after b_1: the process will eventually perform a transition that is not a discard; this must be $b_2!$. Formally, this is achieved by excluding from the definition of progressing path all paths that end in infinitely many discard-transitions (all looping in the final state), where the final state in the path admits a non-blocking action. To avoid further encounters with this complication, we will henceforth assume the original operational semantics.

4.2 Fairness

With the progress requirements (P_1) and (P_2) embedded in our semantics of LTL, the process of Figure 1(a) satisfies the formula $\mathbf{G}(a \Rightarrow \mathbf{F}\phi)$. Yet, the process of Figure 1(b) does not satisfy this formula. The reason is that in state s a choice is made between two internal transitions. One leads to the desired state satisfying ϕ, whereas the other gives the process a chance to make the decision again. This can go wrong in exactly one way, namely if the τ-loop is chosen every time.

For some applications it is warranted to make a *global fairness assumption*, saying that in verifications we may simply assume our processes to eventually escape from a loop such as in Figure 1(b) and do the right thing. A process-algebraic verification approach based on such an assumption is described in [2]. Moreover, a global fairness assumption is incorporated in the weak bisimulation semantics employed in [16]. Different global fairness assumptions in process algebra appear in [4].

An alternative approach, which we follow here, is to explicitly declare certain choices to be fair, while leaving open the possibility that others are not. A *strong fairness* assumption requires that if a task is enabled infinitely often,[11] but allowing interruptions during which it is not enabled, it will eventually be scheduled. Such a property is expressed in LTL as $\mathbf{G}(\mathbf{GF}\psi \Rightarrow \mathbf{F}\phi)$,[12] or equivalently $\mathbf{GF}\psi \Rightarrow \mathbf{GF}\phi$; here ψ is the condition that states that the task is enabled,

[11] or in the final state of a run, although for many tasks this is a logical impossibility

[12] These properties were introduced in LTL in [10] under the name 'responsiveness to persistence'.

whereas ϕ states that it is being executed. A *weak fairness* assumption requires that if a task, from some point onwards, is perpetually enabled, it will eventually be scheduled. In LTL this is expressed as $\mathbf{G}(\mathbf{G}\psi \Rightarrow \mathbf{F}\phi)$,[13] or equivalently $\mathbf{FG}\psi \Rightarrow \mathbf{GF}\phi$.

If a formula ψ holds in state s of Figure 1(b), and ϕ in t, then the strong fairness assumption $\mathbf{G}(\mathbf{GF}\psi \Rightarrow \mathbf{F}\phi)$ ensures the choice at s to be fair. If ψ even holds in (or during) the transition that constitutes the τ-loop, the weak fairness assumption $\mathbf{G}(\mathbf{G}\psi \Rightarrow \mathbf{F}\phi)$ suffices. If this property is part of the specification, the process of Figure 1(b) will satisfy the desired eventually property $\mathbf{G}(a \Rightarrow \mathbf{F}\phi)$.

In general, we propose a specification framework where a process is specified by a pair of a process expression P (for instance in the language ABC) and a *fairness specification* $\mathscr{F}$, consisting of a collection of LTL formulas (where for instance the actions of ABC are allowed as atomic propositions). Typically, $\mathscr{F}$ contains strong or weak fairness properties.

The semantics of such a specification is again a pair. The first component is the state P in the LTS generated by ABC, and the second component, the set of *fair paths*, is a subset of the progressing paths starting from P, namely those that satisfy the formulas in $\mathscr{F}$.

We require the state P in the LTS and the fairness specification to be consistent with each other. By this we mean that from P one cannot reach a state from where, given a sufficiently uncooperative environment, it is impossible to satisfy the fairness specification—in other words [12], 'the automaton can never "paint itself into a corner" '. In [12,13] this requirement is called *machine closure*, and demands that any finite path in the LTS, starting from P, can be extended to a path satisfying $\mathscr{F}$. Since we deal with a reactive system here, we need a more involved consistency requirement, taking into account all possibilities of the environment to allow or block transitions that are not fully controlled by the specified system itself. This requirement can best be explained in terms of a two player game between a *scheduler* and the *environment*.

The game begins with any finite path π starting from P, ending in a state $Q \in \mathrm{Ex}_{\mathrm{ABC}}$, chosen by the environment. In each turn, first the environment selects a set $\mathit{next}(Q)$ of transitions originating from Q; this set has to include all transitions labelled with non-blocking actions originating from Q, but can also include further transitions starting from Q. If $\mathit{next}(Q)$ is empty, the game ends; otherwise the scheduler selects a transition from this set, which is, together with its target state, appended to π, and a new turn starts with the prolonged finite path. The *result* of the game is the finite path in which the game ends, or—if it does not—the infinite path that arises as the limit of all finite paths encountered during the game. The game is *won* by the scheduler iff the result is a progressing[14] path that satisfies $\mathscr{F}$. Now P is *consistent* with $\mathscr{F}$ iff there exists a winning strategy for the scheduler.

[13] These properties were introduced in LTL in [10] under the name 'responsiveness to insistence', and deemed 'the minimal fairness requirement' for any scheduler.

[14] When adopting our proposal of Section 4.3, the resulting path should even be just.

4.3 Justness

Now suppose we have two concurrent systems that work independently in parallel, such as two completely disconnected nodes in a network. One of them is modelled by the transition system of Figure 1(a), and the other is doing internal transitions in perpetuity. The parallel composition is depicted on the left-hand side of Figure 1(c). According to our structural operational semantics, the overall transition system resulting from this parallel composition is the one depicted on the right. In this transition system, the LTL formula $\mathbf{G}(a \Rightarrow \mathbf{F}\phi)$ is not valid, because, after performing the action a, the process may do an infinite sequence of internal transitions that stem from the 'right' component in the parallel composition, instead of the transition to the desired success state. Yet the formula $\mathbf{G}(a \Rightarrow \mathbf{F}\phi)$ does hold intuitively, because no amount of internal activity in the right node should prevent the left component from making progress. That this formula does not hold can be seen as a pitfall stemming from the use of interleaving semantics. The intended behaviour of the process is captured by the following *justness* property:

> *If a combination of components in a parallel composition is in a state that admits a non-blocking action, then one (or more) of them will eventually partake in an action.* (*J*)

Thus justness guarantees progress of all components in a parallel composition, and of all combinations of such components. In the ABC expression $((P|Q)\backslash a)|R$ for instance, we might reach a state where P admits an action $c \in \mathscr{H}$ with $c \neq a$ and R admits $\bar{c}$. Thereby, the combination of these components admits an action τ. Our justness assumption now requires that the combination of P and R will eventually perform an action. This could be the τ-action obtained from synchronising c and $\bar{c}$, but it also could be any other action from either P or R.

Note that progress is a special case of justness, obtained by considering any process as the combination of all its parallel components.

We now formalise the justness requirement (*J*).
Any transition $P|Q \xrightarrow{\ell} R$ derives, through the rules of Table 1, from

- a transition $P \xrightarrow{\ell} P'$ and a state Q, where $R = P'|Q$,
- two transitions $P \xrightarrow{\ell_1} P'$ and $Q \xrightarrow{\ell_2} Q'$, where $R = P'|Q'$,
- or from a state P and a transition $Q \xrightarrow{\ell} Q'$, where $R = P|Q'$.

This transition/state, transition/transition or state/transition pair is called a *decomposition* of $P|Q \xrightarrow{\ell} R$; it need not be unique. Now a *decomposition* of a path π of $P|Q$ into paths π_1 and π_2 of P and Q, respectively, is obtained by decomposing each transition in the path, and concatenating all left-projections into a path of P and all right-projections into a path of Q—notation $\pi \in \pi_1|\pi_2$. Here it could be that π is infinite, yet either π_1 or π_2 (but not both) are finite. Again, decomposition of paths need not be unique.

Likewise, any transition $P[f] \xrightarrow{\ell} R$ stems from a transition $P \xrightarrow{\ell_1} P'$, where $R = P'[f]$. This transition is called a decomposition of $P[f] \xrightarrow{\ell} R$. A *decomposition* of a path π of $P[f]$ is obtained by decomposing each transition in the

path, and concatenating all transitions so obtained into a path of P. In the same way one defines a decomposition of a path of $P\backslash c$.

We now define a path of a process to be *just* if it models a run that can actually occur in some environment, even when postulating (J); we call it *Y-just*, for $Y \subseteq \mathscr{H}$, if it can occur in an environment that from some point onwards blocks all actions in $Y \cup \mathscr{B}?$.

Definition 4.1. *Y-justness*, for $Y \subseteq \mathscr{H}$, is the largest family of predicates on the paths in the transition system $\mathcal{S}$ associated to the LTS $\mathcal{T}$ of ABC such that

- a finite Y-just path ends in a state of $\mathcal{T}$ that admits actions from $Y \cup \mathscr{B}?$ only;
- a Y-just path of a process $P|Q$ can be decomposed into an X-just path of P and a Z-just path of Q such that $Y \supseteq X \cup Z$ and $X \cap \bar{Z} = \emptyset$—here $\bar{Z} := \{\bar{c} \mid c \in Z\}$;
- a Y-just path of $P\backslash c$ can be decomposed into a $Y \cup \{c, \bar{c}\}$-just path of P;
- a Y-just path of $P[f]$ can be decomposed into an $f^{-1}(Y)$-just path of P;
- and each suffix of a Y-just path is Y-just.

A path is *just* if it is Y-just for some $Y \subseteq \mathscr{H}$.

The last clause in the second requirement prevents an X-just path of P and a Z-just path of Q to compose into an $X \cup Z$-just path of $P|Q$ when X contains an action c and Z the complementary action $\bar{c}$. The reason is that no environment can block both actions for their respective components, as nothing can prevent them from synchronising with each other. The fifth requirement helps characterising processes of the form $b.0 + (A|b.0)$ and $a.(A|b.0)$, with $A \stackrel{def}{=} a.A$. Here, the first transition 'gets rid of' the choice and of the leading action a, respectively, and reduces the justness of paths of such processes to their suffixes.

If $Y \subseteq Z$ then any Y-just path is also Z-just. As a consequence, a path is just iff it is $\mathscr{H}$-just. In Appendix A we show that a finite path is just iff it does not end in a state from which a non-blocking action is possible, i.e., iff it is progressing as defined in Section 4.1.

A path is called *complete* if it is fair as well as just, and hence also progressing.

The above definition of a just path captures our (progress and) justness requirement, and ensures that the formula $\mathbf{G}(a \Rightarrow \mathbf{F}\phi)$ holds for the process of Figure 1(c). For example, the infinite path π starting from r that after the a-transition keeps looping through the τ-loop at s can only be derived as $\pi_1|\pi_2$, where π_1 is a finite path ending right after the a-transition. Since π_1 fails to be just (its end state admits a τ-transition), π fails to be just too, and hence does not count when searching for a complete path that fails to satisfy $\mathbf{G}(a \Rightarrow \mathbf{F}\phi)$.

4.4 Justness versus Justice

The concept of *justice* was introduced in [14]: 'A computation is said to be *just* if it is finite or if every transition which is continuously enabled beyond a certain point is taken infinitely many times.' In LTL this amounts to $\mathbf{FG}\, en(\nu) \Rightarrow \mathbf{GF}\, \nu$ for each transition ν, thus casting justice as a weak fairness property.

In [14] the identity of a transition, when appearing in a parallel composition, is not affected by the current state of the parallel component. For instance, the

two transitions $c.0|d.0 \xrightarrow{c} 0|d.0$ and $c.0|0 \xrightarrow{c} 0|0$ —they differ in their source and target states—are seen as the same transition of the process $c.0|d.0$, stemming from the left component and scheduled either before or after the d-transition of the right component. In Appendix D, to be read after B and C, we introduce the notion of an *abstract transition*—an equivalence class of *concrete transitions*—to formalise the transitions intended in [14].

In the context of reactive systems, an (abstract) transition ν typically is a synchronisation between the system and its environment. In case the environment does not synchronise, ν cannot happen, even when it is continuously enabled. For this reason, here justice is only reasonable for abstract transitions ν labelled with non-blocking actions.

In applying the concept of justice from [14] to LTSs, there is potential ambiguity in what counts as 'continuously'. Consider the ABC system specified by $B \stackrel{def}{=} c.B + b!.0$. By Definition 4.1, the computation consisting of cs only is just; it satisfies (J). However, it could be argued that $b!$ is continuously enabled. This would make the computation unjust in the sense of [14]. On the other hand, the choice between c and $b!$ may be non-deterministic, and could always be resolved in favour of c. Therefore we do not consider this computation unjust, and adopt the principle of 'noninstantaneous readiness' [1], stating that the enabledness of the $b!$ is interrupted when performing the c-transition. In our model, this is implemented by the midway states corresponding with transitions. As a result, we judge the specified execution just, and hence do not claim that $b!$ will happen eventually.

On the other hand, in our vision the enabledness of a transition cannot be interrupted by performing a concurrent transition. For instance, the execution c^ω of the process $C|b!.0$, where $C \stackrel{def}{=} c.C$, is unjust, because the $b!$-transition $\nu_{b!}$ is continuously enabled and never taken. In Appendices C and D we formalise this by a novel definition of the predicates $en(\nu)$, such that $en(\nu_{b!})$ holds during the transition $C|b!.0 \xrightarrow{c} C|b!.0$.

In doing so, we have to overcome a problem illustrated by the process $C|B$ with B and C as above. Whether the path $C|B \xrightarrow{c} C|B \xrightarrow{c} \ldots$ counts as being just by the mantra of [14] depends on whether $en(\nu_{b!})$ holds during each transition $C|B \xrightarrow{c} C|B$ in that path. This, in turn, depends on whether these transitions originate from C, so that they are concurrent with $\nu_{b!}$, or from B. We formalise this by using a richer transition system $\mathcal{U}$ in which the two transitions $C|B \xrightarrow{c} C|B$ are distinguished. The states of $\mathcal{U}$ are the states of $\mathcal{T}$ together with the *derivations* of transitions of $\mathcal{T}$ from the rules of Table 1—the latter are the *concrete transitions* alluded to above. The transitions of $\mathcal{U}$ are $P \to \chi$ and $\chi \to Q$, for any derivation χ of a transition $P \xrightarrow{\alpha} Q$. The predicates $en(\nu)$ are defined on the states of $\mathcal{U}$. The transition system $\mathcal{S}$ associated to the LTS $\mathcal{T}$ of ABC can be obtained from $\mathcal{U}$ by consistently identifying multiple derivations of the same transition. Now, any path π in $\mathcal{U}$ projects onto a path $\widehat{\pi}$ in $\mathcal{S}$, and any path in $\mathcal{S}$ is of the form $\widehat{\pi}$. Details can be found in Appendix B.

In the literature [7,15], the concept of *weak fairness* often occurs as a synonym for "justice". At the same time, the potential ambiguity in what counts as being continuously enabled is resolved differently from the approach we take here:

a transition that from some point onwards is enabled *in every state* cannot be ignored forever. Under this notion of weak fairness, the system B discussed above will surely perform the $b!$-action. It would be useful to have different names for a concept of justice or weak fairness that adopts the principle of noninstantaneous readiness and one that does not.

The following theorem states that the former concept of justice is in perfect agreement with our notion of justness of Section 4.3. Its proof can be found in Appendix E.

Theorem 4.2. *A path of an ABC process is just in the sense of Definition 4.1 iff it is of the form $\widehat{\pi}$ for a path π in $\mathcal{U}$ that satisfies the LTL formulas $\mathbf{FG}\,en(\nu) \Rightarrow \mathbf{GF}\,\nu$ for each abstract transition ν with $\ell(\nu) \in Act$ a non-blocking action.*

5 Implementing Fairness Specifications

For certain properties of the form $(\bigvee_i \mathbf{GF}a_i) \Rightarrow (\bigvee_j \mathbf{GF}b_j)$ where the a_i and b_i are action occurrences—hence for specific strong fairness properties—one can define a *fairness operator* that transforms a given LTS into a LTS that satisfies the property [21]. This is done by eliminating all the paths that do not satisfy the property via a carefully designed parallel composition. In the same vein, we ask whether any process specification involving a fairness specification can be implemented by means of a process-algebraic expression without fairness component. Here we give an example that we believe cannot be implemented in standard process algebras like CCS. To make this more precise, let CCS$^!$ be the fragment of ABC without receive actions; equivalently, this is the fragment of CCS in which certain names b induce no co-names $\bar{b}$ and no restriction operators $\backslash b$. These actions are deemed output actions, meaning that we do not consider environments that can prevent them from occurring.

Consider the CCS$^!$ process $(I_1 \,|\, G \,|\, I_2)\backslash c_1 \backslash c_2$, where

$$I_i \stackrel{def}{=} r_i.\bar{c_i}.I_i \quad (i \in \{1,2\}) \text{ and } \qquad G \stackrel{def}{=} c_1.t_1.e.G + c_2.t_2.e.G$$

augmented with the fairness specification $\bigwedge_{i=1,2} \mathbf{G}(r_i \Rightarrow \mathbf{F}(t_i!))$.

Here t_1, t_2, e are output actions. This process could be called a *fair scheduler*. The actions r_1 and r_2 can be seen as requests received from the environment to perform tasks t_1 and t_2, respectively. Each r_i triggers a task t_i. Moreover, between each two occurrences of t_i and t_j an action e needs to be scheduled.

Conjecture 1. *There does not exist a CCS$^!$ expression G such that the process $(I_1 \,|\, G \,|\, I_2)\backslash c_1 \backslash c_2$, with I_1 and I_2 as above, has the following properties:*

1. *On each complete (= just) path, each r_i is followed by a t_i.*
2. *On each finite path no more t_is than r_is occur.*
3. *Between each pair of occurrences of t_i and t_j ($i, j \in \{1,2\}$) an action e occurs.*

We use CCS$^!$ rather than CCS to prevent the environment invalidating *1.* by disallowing t_i. We believe that there is no way to encode a fair scheduler with these properties in CCS$^!$ without the help of a fairness specification.

However, we can do it in ABC:

$$I_1 \stackrel{def}{=} r_1.c_1!.I_1 \qquad I_2 \stackrel{def}{=} r_2.c_2!.I_2$$
$$G \stackrel{def}{=} c_1?.G_1 + c_2?.G_2 \qquad G' \stackrel{def}{=} e.G + c_1?.G'_1 + c_2?.G'_2$$
$$G_i \stackrel{def}{=} c_j?.G_{ij} + t_i.G' \qquad G'_i \stackrel{def}{=} e.G_i + c_j?.G'_{ij}$$
$$G_{ij} \stackrel{def}{=} t_i.G'_j \qquad G'_{ij} \stackrel{def}{=} e.G_{ij}$$

with $i, j \in \{1, 2\}$ and $i \neq j$. This scheduler satisfies the fairness specification since the justness properties for output actions require that once r_i occurs, $c_i!$ must follow, and then t_i will eventually happen, at the latest when G_{ij} is reached.

Currently, it is an open question whether arbitrary fairness specifications can be implemented in ABC.

6 Conclusion and Outlook

In this paper we have investigated how progress, fairness and justness assumptions can be handled within a process-algebraic specification formalism. Our semantics of a process is a state P in an LTS together with a set of complete paths: paths of P that are *progressing*, *fair* and *just*. We specify the fair paths by means of temporal logic, using a *fairness specification* in addition to a process-algebraic expression P. The progressing and just paths, on the other hand, are completely determined by the syntax of P.

To demonstrate that the introduced approach is not only a theoretical result, we have applied the formalism to a more involved process algebra called AWN (Algebra for Wireless Networks) and analysed the IETF-standardised *Ad hoc On-demand Distance Vector (AODV) routing protocol* [17]. We investigated two fundamental properties of routing protocols: *route discovery* and *packet delivery*. Route discovery—a property that every routing protocol ought to satisfy—states that if a route discovery process is initiated in a state where the source is connected to the destination and no (relevant) link breaks, then the source will eventually discover a route to the destination. Surprisingly, using the presented mechanism we could show that this property does not hold. The second property, packet delivery, was already sketched in Section 3; it has been shown that this property does not hold either. As a consequence, AODV does not satisfy two of the most crucial properties of routing protocols. Details can be found in [8]. The formalisation of progress, fairness and justness presented here was crucial for these results; without making these assumptions, no routing protocol would satisfy the route discovery and packet delivery properties.

Future work will include the definition of suitable semantic equivalences on an LTS together with a set of complete paths, and their algebraic characterisations.

References

1. Apt, K.R., Francez, N., Katz, S.: *Appraising fairness in languages for distributed programming*. Distributed Computing 2(4), 226–241 (1988)
2. Baeten, J.C.M., Bergstra, J.A., Klop, J.W.: *On the consistency of Koomen's fair abstraction rule*. TCS 51(1/2), 129–176 (1987)
3. Bloom, B., Istrail, S., Meyer, A.: *Bisimulation can't be traced*. J. ACM 42(1), 232–268 (1995)
4. Costa, G., Stirling, C.: Weak and strong fairness in CCS. Information and Computation **73**(3), 207–244 (1987).
5. Cranen, S., Mousavi, M.R., Reniers, M.A.: *A rule format for associativity*. In: van Breugel, F., Chechik, M. (eds.) CONCUR '08. LNCS, vol. 5201, pp. 447–461. Springer (2008)
6. De Nicola, R., Vaandrager, F.W.: *Three logics for branching bisimulation*. J. ACM 42(2), 458–487 (1995)
7. Emerson, E.A.: *Temporal and modal logic*. In: Handbook of Theoretical Computer Science (vol. B): Formal Models and Semantics, pp. 995–1072. MIT Press (1990)
8. Fehnker, A., van Glabbeek, R.J., Höfner, P., McIver, A.K., Portmann, M., Tan, W.L.: *A process algebra for wireless mesh networks used for modelling, verifying and analysing AODV*. Tech. Rep. 5513, NICTA (2013), available at `http://arxiv.org/abs/1312.7645`
9. Francez, N.: *Fairness*. Springer (1986)
10. Gabbay, D.M., Pnueli, A., Shelah, S., Stavi, J.: *On the temporal analysis of fairness*. In: Abrahams, P.W., Lipton, R.J., Bourne, S.R. (eds.) POPL '80. pp. 163–173. ACM (1980)
11. Huth, M., Ryan, M.D.: *Logic in Computer Science - Modelling and Reasoning about Systems*. CUP, 2nd edn. (2004)
12. Lamport, L.: *Fairness and hyperfairness*. Distrib. Comput. 13(4), 239–245 (2000)
13. Lamport, L.: *The temporal logic of actions*. ACM Trans. Program. Lang. Syst. 16(3), 872–923 (1994)
14. Lehmann, D.J., Pnueli, A., Stavi, J.: *Impartiality, justice and fairness: The ethics of concurrent termination*. In: Even, S., Kariv, O. (eds.) ICALP'81. LNCS, vol. 115, pp. 264–277. Springer (1981)
15. Manna, Z., Pnueli, A.: *The Temporal Logic of Reactive and Concurrent Systems*. Springer. (1992)
16. Milner, R.: *Communication and Concurrency*. Prentice Hall (1989)
17. Perkins, C.E., Belding-Royer, E.M., Das, S.: *Ad hoc on-demand distance vector (AODV) routing*. RFC 3561 (Experimental), Network Working Group (2003), `http://www.ietf.org/rfc/rfc3561.txt`
18. Plotkin, G.D.: *A structural approach to operational semantics*. JLAP 60–61, 17–139 (2004), originally appeared in 1981
19. Pnueli, A.: *The temporal logic of programs*. In: FOCS '77. pp. 46–57. IEEE (1977)
20. Prasad, K.V.S.: *A calculus of broadcasting systems*. In: Abramsky, S., Maibaum, T.S.E. (eds.) TAPSOFT '91. LNCS, vol. 493, pp. 338–358. Springer (1991)
21. Puhakka, A., Valmari, A.: *Liveness and fairness in process-algebraic verification*. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR'01. LNCS, vol. 2154, pp. 202–217. Springer (2001)
22. de Simone, R.: *Higher-level synchronising devices in* Meije-SCCS. TCS 37, 245–267 (1985)

Appendices

A Finite Paths are Just iff they are Progressing

Proposition A.1. *A finite path in $\mathcal{S}$ is Y-just, for $Y \subseteq \mathscr{H}$, iff its last state is a state $Q \in \mathrm{Ex}_{\mathrm{ABC}}$ of $\mathcal{T}$ and all transitions enabled in Q are labelled with actions from $Y \cup \mathscr{B}?$.*

Proof. "$\Rightarrow$": This follows immediately from the first requirement of Definition 4.1.

"$\Leftarrow$": Define a path in $\mathcal{S}$ to be *Y-just$_{fin}$* if it is finite, its last state is a state $Q \in \mathrm{Ex}_{\mathrm{ABC}}$ of $\mathcal{T}$, and all transitions enabled in Q are labelled with actions from $Y \cup \mathscr{B}?$. Then the family of predicates Y-justness$_{\mathrm{fin}}$, for $Y \subseteq \mathscr{H}$, satisfies the five requirements of Definition 4.1. Since Y-justness is the largest family of predicates satisfying those requirements, Y-justness$_{\mathrm{fin}}$ implies Y-justness. □

It follows that a finite path is just iff it is progressing.

B A Concrete Kripke Structure for ABC

In Section 3 we extracted a transition system $\mathcal{S}$ with unlabelled transitions—a *Kripke structure* [11], but without the condition of totality—out of the LTS $\mathcal{T}$ generated by the structural operational semantics of ABC. The states of $\mathcal{S}$ are the states of $\mathcal{T}$, that is, $\mathrm{Ex}_{\mathrm{ABC}}$, together with the transitions $P \xrightarrow{\alpha} Q$ of $\mathcal{T}$. The transitions of $\mathcal{S}$ are $P \to (P \xrightarrow{\alpha} Q)$ and $(P \xrightarrow{\alpha} Q) \to Q$, for any transition $P \xrightarrow{\alpha} Q$ of $\mathcal{T}$. Next, we would like to define predicates $en(\nu)$ on the states of $\mathcal{S}$ indicating whether an (abstract) transition ν is enabled in a state s of $\mathcal{S}$. If s is actually a state of $\mathcal{T}$, this is the case if s is the source of ν. If s is a transition ζ of $\mathcal{T}$, this should be the case if ν is enabled in the source of ζ, and moreover ν and ζ are concurrent, in the sense that they stem from different parallel components. A problem with this plan has been illustrated in Section 4.4 by the process $C|B$, with $B \stackrel{def}{=} c.B + b!.0$ and $C \stackrel{def}{=} c.C$. The $b!$-labelled transition ν is enabled in (or during) the transition $C|B \xrightarrow{c} C|B$ if this transition stems from C, but not if it stems from B. However, our transition system $\mathcal{S}$ fails to distinguish transitions based on the components from which they stem.

For this reason, we here define a different Kripke structure $\mathcal{U}$ that makes the required distinctions. The states of $\mathcal{U}$ are the states $\mathrm{Ex}_{\mathrm{ABC}}$ of $\mathcal{T}$ together with the *derivations* of transitions of $\mathcal{T}$ from the rules of Table 1—the latter are the *concrete transitions* alluded to in Section 4.4. The transitions of $\mathcal{U}$ are of the form $P \to \chi$ and $\chi \to Q$, for $\mathcal{T}$-states P, Q and derivations χ corresponding to a transition $P \xrightarrow{\alpha} Q$. The Kripke structure $\mathcal{S}$ can be obtained from $\mathcal{U}$ by consistently identifying multiple derivations of the same transition.

We start by giving a name to every derivation of an ABC transition from the rules of Table 1. The unique derivation of the transition $\alpha.P \xrightarrow{\alpha} P$ using the rule (Act) is called $\xrightarrow{\alpha}P$. The derivation obtained by application of (Comm) or (Bro-c) on the derivations χ and ζ of the premises of that rule is called $\chi|\zeta$. The derivation obtained by application of (Par-l) or (Bro-l) on the derivation

χ of the (positive) premise of that rule, and using process Q at the right of $|$, is $\chi|Q$. In the same way, (Par-r) and (Bro-r) yield $P|\zeta$, whereas (Sum-l), (Sum-r), (Rel), (Res) and (Rec) yield $\chi{+}Q$, $P{+}\chi$, $\chi[f]$, $\chi\backslash c$ and $A{:}\chi$. For a derivation χ of a transition $P \xrightarrow{\alpha} Q$ write $src(\chi) := P$, $target(\chi) := Q$ and $\ell(\chi) := \alpha$.

It remains to define atomic propositions on $\mathcal{U}$. Following Section 3 we have an atomic proposition α for each $\alpha \in Act$, and a state u of $\mathcal{U}$ is labelled with α iff u is a derivation of a transition $P \xrightarrow{\alpha} Q$. Additionally, Section 4.4 announced atomic propositions ν and $en(\nu)$ for each *abstract transition* ν; this is the subject of Appendix D.

Let $\widehat{\cdot}$ be the mapping from the states of $\mathcal{U}$ to the states of $\mathcal{S}$ given by $\widehat{P} = P$ for any process $P \in \mathrm{Ex}_{\mathrm{ABC}}$, and $\widehat{\chi} = (P \xrightarrow{\alpha} Q)$ for any derivation χ of a transition $P \xrightarrow{\alpha} Q$ in $\mathcal{T}$. Then each state of $\mathcal{S}$ is of the form $\widehat{u}$, and there is a transition $\widehat{u} \to s$ in $\mathcal{S}$ iff there is a transition $u \to v$ in $\mathcal{U}$ with $\widehat{v} = s$. A *path* π in $\mathcal{U}$ is a finite or infinite sequence $u_0u_1u_2\ldots$ of states of $\mathcal{U}$ such that $u_i \to u_{i+1}$ for all i. This amounts to an alternating sequence of processes $P \in \mathrm{Ex}_{\mathrm{ABC}}$ and derivations χ of transitions from $\mathcal{T}$. For any such path π let $\widehat{\pi} := \widehat{u}_0\widehat{u}_1\widehat{u}_2\ldots$. Then $\widehat{\pi}$ is a path in $\mathcal{S}$, and furthermore any path in $\mathcal{S}$ is of this form. Both in $\mathcal{U}$ and in $\mathcal{S}$ we only consider paths that are infinite or end in a state of $\mathcal{T}$.

C An Asymmetric Concurrency Relation between Transitions

We define a *concurrency* relation $\frown\!\!\bullet$ between the derivations of the outgoing transitions of a process $P \in \mathrm{Ex}_{\mathrm{ABC}}$. With $\chi \frown\!\!\bullet \zeta$ we mean that the possible occurrence of χ is unaffected by the possible occurrence of ζ. More precisely, χ and ζ need to be enabled in the state P, and $\chi \not\frown\!\!\bullet \zeta$ indicates that the occurrence of ζ ends or interrupts the enabledness of χ, whereas $\chi \frown\!\!\bullet \zeta$ indicates that χ remains enabled during the execution of ζ.

Example C.1. Let P be the process A with $A \stackrel{def}{=} a.A + c.A$, and let χ and ζ be the derivations of the a- and c-transitions of P. Then $\chi \not\frown\!\!\bullet \zeta$, because the occurrence of ζ interrupts the enabledness of χ, even though right after ζ has occurred we again reach a state where χ is enabled.

Example C.2. Let P be the process $a.0|c.0$, and let χ and ζ be the derivations of the a- and c-transitions. Then $\chi \frown\!\!\bullet \zeta$, because the occurrence of ζ does not affect the (parallel) occurrence of χ in any way.

Example C.3. Let P be the process $b!.0|(b?.0 + c.0)$, and let χ and ζ be the derivations of the $b!$- and c-transitions of P. The broadcast $b!$ is in our view completely under the control of the left component; it will occur regardless of whether the right component listens to it or not. It so happens that if $b!$ occurs in state P, the right component will listen to it, thereby disabling the possible occurrence of c. For this reason we have $\chi \frown\!\!\bullet \zeta$ but $\zeta \not\frown\!\!\bullet \chi$.

Definition C.4. *Concurrency* is the smallest relation $\smile\!\!\bullet$ on derivations such that

- $\chi|Q \smile\!\!\bullet P|\zeta$ and $P|\zeta \smile\!\!\bullet \chi|Q$ if $src(\chi) = P$ and $src(\zeta) = Q$,
- $\chi|\varsigma \smile\!\!\bullet P|\zeta$ and $\varsigma|\chi \smile\!\!\bullet \zeta|P$ if $src(\chi) = P$, $src(\varsigma) = src(\zeta)$ and $\ell(\varsigma) \in \mathscr{B}?$,
- $\chi \smile\!\!\bullet \zeta$ implies $\chi{+}P \smile\!\!\bullet \zeta{+}P$, $P{+}\chi \smile\!\!\bullet P{+}\zeta$, $\chi|P \smile\!\!\bullet \zeta|P$ and $P|\chi \smile\!\!\bullet P|\zeta$,
- $\chi \smile\!\!\bullet \zeta$ implies $\chi|P \smile\!\!\bullet \zeta|\xi$, $\chi|\xi \smile\!\!\bullet \zeta|P$, $P|\chi \smile\!\!\bullet \xi|\zeta$ and $\xi|\chi \smile\!\!\bullet P|\zeta$ if $P = src(\xi)$,
- $\chi \smile\!\!\bullet \zeta$ implies $\chi|\varsigma \smile\!\!\bullet \zeta|\xi$ and $\varsigma|\chi \smile\!\!\bullet \xi|\zeta$ if $src(\varsigma) = src(\xi)$ and $\ell(\varsigma) \in \mathscr{B}?$,
- $\chi \smile\!\!\bullet \zeta \wedge \varsigma \smile\!\!\bullet \xi$ implies $\chi|\varsigma \smile\!\!\bullet \zeta|\xi$, and
- $\chi \smile\!\!\bullet \zeta$ implies $\chi\backslash c \smile\!\!\bullet \zeta\backslash c$, $\chi[f] \smile\!\!\bullet \zeta[f]$, $A{:}\chi \smile\!\!\bullet A{:}\zeta$ for any $c \in \mathscr{H}$, relabelling f and $A \in \mathscr{A}$,

for arbitrary derivations χ, ζ, ς, ξ, and expressions $P, Q \in \mathrm{Ex}_{\mathrm{ABC}}$, provided that the composed derivations exist. We say that χ and ζ are *concurrent* (in signs $\chi \smile \zeta$) if $\chi \smile\!\!\bullet \zeta$ and $\zeta \smile\!\!\bullet \chi$.

Observation C.5. The relation $\smile\!\!\bullet$ is irreflexive. Moreover, if $\chi \smile\!\!\bullet \zeta$ then $src(\chi) = src(\zeta)$.

This follows by a straightforward induction on the definition of $\smile\!\!\bullet$.

Example C.6. Let $A \stackrel{def}{=} c.A$ and $B \stackrel{def}{=} \bar{c}.B + (\tau.B + b!.0)$. Transition $A|B \xrightarrow{\tau} A|B$ has 2 derivations: $(A{:}\stackrel{c}{\rightarrow}A)|B{:}\big((\stackrel{\bar{c}}{\rightarrow}B){+}(\tau.B{+}b!.0)\big)$ and $A|B{:}\big(\bar{c}.B{+}((\stackrel{\tau}{\rightarrow}B){+}b!.0)\big)$. Only the latter is concurrent with $(A{:}\stackrel{c}{\rightarrow}A)|B$ (using the first clause above).

Example C.7. One has $(((\stackrel{a}{\rightarrow}0)|c.0) + d.0)[f] \smile ((a.0|(\stackrel{c}{\rightarrow}0)) + d.0)[f]$, using the first, third and seventh clauses above. Both are derivations of transitions with source $((a.0|c.0) + d.0)[f]$.

Example C.8. One has $((\stackrel{a}{\rightarrow}0)|c.0)|((\stackrel{\bar{a}}{\rightarrow}0)|\bar{c}.0) \smile (a.0|(\stackrel{c}{\rightarrow}0))|(\bar{a}.0|(\stackrel{\bar{c}}{\rightarrow}0)))$, using the first and sixth clauses above. Both are derivations of transitions with source $(a.0|c.0)|(\bar{a}.0|\bar{c}.0)$.

Example C.9. One has $((\stackrel{a}{\rightarrow}0)|c.0)|(\stackrel{\bar{a}}{\rightarrow}0) \smile (a.0|(\stackrel{c}{\rightarrow}0))|\bar{a}.0$, using the first and fourth clauses above. Both are derivations of transitions with source $(a.0|c.0)|\bar{a}.0$.

Example C.10. One has $\stackrel{b!}{\rightarrow}0\,|((\stackrel{b?}{\rightarrow}0){+}c.0) \smile\!\!\bullet\ b!.0\,|(b?.0{+}(\stackrel{c}{\rightarrow}0))$, using the second clause above. Both are derivations of transitions with source $b!.0|(b?.0 + c.0)$. However, $b!.0\,|(b?.0{+}(\stackrel{c}{\rightarrow}0)) \not\smile\!\!\bullet\ \stackrel{b!}{\rightarrow}0\,|((\stackrel{b?}{\rightarrow}0){+}c.0)$. See Example C.3 for motivation.

Example C.11. One has $((\stackrel{b!}{\rightarrow}0)|c.0)|((\stackrel{b?}{\rightarrow}0) + \bar{c}.0) \smile\!\!\bullet\ (b!.0|(\stackrel{c}{\rightarrow}0))|(b?0 + (\stackrel{\bar{c}}{\rightarrow}0))$, using the first and fifth clauses above. Both are derivations of transitions with source $(b!.0|c.0)|(b!.0 + \bar{c}.0)$.

D Enabling Abstract Transitions

Below we define the concept of an *abstract transition* as an equivalence class of *concrete* transitions, the latter being the derivations of ABC transitions from the rules of Table 1. The main idea is that a transition ν stemming from one side of a parallel composition yields only one abstract transition of the parallel composition itself, regardless of the state of the other component, and, if ν performs a broadcast action, regardless of whether the other component performs a receive action synchronising with ν.

Henceforth, ν, ν_1 and ν_2, ranges over abstract transitions, χ, ζ, ς and ξ over derivations, P, Q over ABC expressions, and u, v over states of $\mathcal{U}$, which are either derivations or ABC expressions.

Definition D.1. Let $\equiv$ be the smallest equivalence relation on derivations χ with $\ell(\chi) \notin \mathscr{B}?$, satisfying

- $\chi|P \equiv \chi|Q$ and $P|\chi \equiv Q|\chi$,
- $\chi|\varsigma \equiv \chi|P$ and $\varsigma|\chi \equiv P|\chi$ if $\ell(\chi) \in \mathscr{B}!$ (and thus $\ell(\varsigma) \in \mathscr{B}?$),
- $\chi{+}P \equiv \chi \equiv P{+}\chi$, $A{:}\chi \equiv \chi$ for $A \in \mathscr{A}$,
- $\chi \equiv \zeta$ implies $\chi\backslash c \equiv \zeta\backslash c$, $\chi[f] \equiv \zeta[f]$, $\chi|P \equiv \zeta|P$ and $P|\chi \equiv P|\zeta$, and moreover
- $\chi \equiv \zeta \wedge \varsigma \equiv \xi$ implies $\chi|\varsigma \equiv \zeta|\xi$,

for arbitrary derivations χ, ζ, ς, ξ, and expressions $P, Q \in \mathrm{Ex}_{\mathrm{ABC}}$, provided that the composed derivations exist. An equivalence class $[\chi]_\equiv$ is called an *abstract transition*; it can uniquely be denoted by leaving out $A{:}$, $P{+}$ and $+P$ and writing $\zeta|\_$ for $\zeta|P$ or for $\zeta|\varsigma$ with $\ell(\zeta) \in \mathscr{B}!$—and likewise $\_|\zeta$ for $P|\zeta$ or for $\varsigma|\zeta$ with $\ell(\zeta) \in \mathscr{B}!$—in all subexpressions of χ. If $\nu = [\chi]_\equiv$ then the derivation χ is called a *representative* of the abstract transition ν.

By definition $\chi \equiv \zeta$ implies $\ell(\chi) = \ell(\zeta)$. Setting $\ell([\chi]_\equiv) := \ell(\chi)$, we note that receive-actions are excluded, i.e., for any abstract transition ν we have $\ell(\nu) \notin \mathscr{B}?$.

Observation D.2. *If $\chi|u \equiv v_1|v_2$ with $\ell(\chi) \notin \mathscr{B}?$ then $\chi \equiv v_1$. As a consequence, since no derivation is related to a process, $\chi|u \not\equiv Q|\zeta$, for any derivation ζ and $Q \in \mathrm{Ex}_{\mathrm{ABC}}$.*

Observation D.3. *If $\chi[f] \equiv \zeta[f]$ or $\chi\backslash c \equiv \zeta\backslash c$ then $\chi \equiv \zeta$.*

The abstract transitions, with their labels, can be seen as the smallest set such that

- $\xrightarrow{\alpha}P$ is an abstract tr. for $\alpha \in \mathscr{B}! \cup \mathscr{H} \cup \{\tau\}$ and $P \in \mathrm{Ex}_{\mathrm{ABC}}$, and $\ell(\xrightarrow{\alpha}P) = \alpha$,
- if ν is an abstract tr. then so are $\nu|\_$ and $\_|\nu$, with $\ell(\nu|\_) = \ell(\_|\nu) = \ell(\nu)$,
- if ν_1 and ν_2 are abstract trs. with $\ell(\nu_1) = \overline{\ell(\nu_2)}$ then so is $\nu_1|\nu_2$, with $\ell(\nu_1|\nu_2) = \tau$,
- if ν is an abstract tr. with $c \neq \ell(\nu) \neq \bar{c}$ then so is $\nu\backslash c$, with $\ell(\nu\backslash c) = \ell(\nu)$, and
- if ν is an abstract tr. and f a relabelling then so is $\nu[f]$, with $\ell(\nu[f]) = f(\ell(\nu))$.

Abstract transitions only reflect the syntactical structure of derivations; they do not take semantics into account. Hence, $\nu|\_ \neq \_|\nu$ and $(\_|\nu)|\_ \neq \_|(\nu|\_)$.

For each abstract transition ν we introduce atomic propositions ν and $en(\nu)$. The former says that ν occurs. It holds for a state u of $\mathcal{U}$ iff u is a derivation ζ such that $\nu = [\zeta]_{\equiv}$. The latter is defined by a case distinction on the type of state u. An abstract transition ν is *enabled* (denoted by $en(\nu)$) in $P \in \mathrm{Ex}_{\mathrm{ABC}}$ iff $P = src(\chi)$ for a representative χ of ν. It is enabled in (or during) a derivation ζ iff ν has a representative χ with $\chi \frown\!\!\bullet\, \zeta$. As we shall see, in that case ν is also enabled in $src(\zeta)$ as well as $target(\zeta)$. We write $u \models p$ if the atomic proposition p holds in the state u of $\mathcal{U}$.

Example D.4. The abstract transition $\nu = \_|(\xrightarrow{c}0)$, with $c \in \mathscr{H}$ and representatives $\chi_1 = 0|(\xrightarrow{c}0)$ and $\chi_2 = a.0 + \big(e.0|(\xrightarrow{c}0)\big)$, is enabled during the derivation $\zeta = a.0 + \big((\xrightarrow{e}0)|c.0\big)$ of the transition $a.0 + (e.0|c.0) \xrightarrow{e} 0|c.0$. This is the case because $\chi_2 \frown \zeta$. Accordingly, ν is also enabled in the source $src(\zeta) = a.0 + (e.0|c.0)$ as well as in the target $target(\zeta) = 0|c.0$ of that transition. This example would break down without the identification $\chi \equiv P{+}\chi$.

Example D.5. Let $C \stackrel{def}{=} d.0|e.0$. Then the abstract transition $\nu = \_|(\xrightarrow{d}0|\_)$ is enabled during the derivation $\zeta = c.0|C{:}(d.0|\xrightarrow{e}0)$ of the transition $c.0|C \xrightarrow{e} c.0|(d.0|0)$. Accordingly, it is enabled in the source $src(\zeta) = c.0|C$ as well as in the target $target(\zeta) = c.0|(d.0|0)$ of that transition. This example would break down without the identification $C{:}(d.0|\_) \equiv (d.0|\_)$.

Example D.6. Let $D \stackrel{def}{=} c.(b?.0 + e.D)$. In our view, the infinite path labelled $(ce)^\omega$ of the process $b!.0|D$ is unjust, because the output $b!$ is continuously enabled, yet never taken. The idea is that the component $b!.0$ will perform this output regardless of whether the other component is listening. For this reason, we need to formalise this output as a single abstract transition ν such that the path labelled $(ce)^\omega$ satisfies $\mathbf{FG}\ en(\nu)$. However, in the state $b!.0|D$—and during the execution of $b!.0|D{:}\xrightarrow{c}(b?.0+e.D)$—the derivation $(\xrightarrow{b!}0)|D$ is enabled, yet in the state $b!.0|(b?.0 + e.D)$ the derivation $\xrightarrow{b!}0|((\xrightarrow{b?}0) + c.D)$ is enabled. In order to regard these two derivations as representatives of the same abstract transition $(\xrightarrow{b!}0)|\_$ we employ the equivalence $\chi|Q \equiv \chi|\varsigma$ when $\ell(\chi) \in \mathscr{B}!$.

Furthermore, during the execution of $b!.0|(b?.0 + \xrightarrow{e}D)$, a representative χ of $(\xrightarrow{b!}0)|\_$ with $src(\chi) = b!.0|(b?.0 + e.D)$ needs to be enabled as well. The only candidate is $\chi = \xrightarrow{b!}0|((\xrightarrow{b?}0) + c.D)$, so $\xrightarrow{b!}0|((\xrightarrow{b?}0) + c.D) \frown\!\!\bullet\, b!.0|(b?.0 + \xrightarrow{e}D)$. This is further motivation for the second clause in the Definition C.4 above.

Lemma D.7. *If $u \models en(\nu)$ for a state u of $\mathcal{U}$ and an abstract transition ν then $u|v \models en(\nu|\_)$ for any state $u|v$ of $\mathcal{U}$.*

Proof. We make case distinctions based on whether u and v are processes P, Q or derivations.

– Suppose $u = P \models en(\nu)$. Then $P = src(\chi)$ for a representative χ of ν.

  - Let $v = Q \in \mathrm{Ex}_{\mathrm{ABC}}$. In case $\ell(\chi) = b!$ and $Q \xrightarrow{b?}$, let ς be a derivation of a transition $Q \xrightarrow{b?} Q'$. Then there exists a derivation $\chi|\varsigma$, with $src(\chi|\varsigma) = P|Q$, which is a representative of the abstract transition $\nu|\_$. In all other cases there exists a derivation $\chi|Q$, with $src(\chi|Q)=P|Q$, which is a representative of the abstract transition $\nu|\_$. Either way $P|Q \models en(\nu|\_)$.
  - Now let $v = \xi$ be a derivation with $Q := src(\xi)$. In case $\ell(\chi) = b!$ and $Q \xrightarrow{b?}$, let ς be a derivation of a transition $Q \xrightarrow{b?} Q'$. Then there exists a derivation $\chi|\varsigma$, with $\chi|\varsigma \smile\!\!\bullet\, P|\xi$, which is a representative of the abstract transition $\nu|\_$. In all other cases there exists a derivation $\chi|Q$, with $\chi|Q \smile P|\xi$, which is a representative of the abstract transition $\nu|\_$. Either way $P|\xi \models en(\nu|\_)$.

– Suppose $u = \zeta \models en(\nu)$. Then $\chi \smile\!\!\bullet\, \zeta$ for a representative χ of ν.
  - Let $v = Q \in \mathrm{Ex}_{\mathrm{ABC}}$. In case $\ell(\chi) = b!$ and $Q \xrightarrow{b?}$, let ς be a derivation of a transition $Q \xrightarrow{b?} Q'$. Then there exists a derivation $\chi|\varsigma$, with $\chi|\varsigma \smile\!\!\bullet\, \zeta|Q$, which is a representative of the abstract transition $\nu|\_$. In all other cases there exists a derivation $\chi|Q$, with $\chi|Q \smile\!\!\bullet\, \zeta|Q$, which is a representative of the abstract transition $\nu|\_$. Either way $\zeta|Q \models en(\nu|\_)$.
  - Now let $v = \xi$ be a derivation with $Q := src(\xi)$. In case $\ell(\chi) = b!$ and $Q \xrightarrow{b?}$, let ς be a derivation of a transition $Q \xrightarrow{b?} Q'$. Then there exists a derivation $\chi|\varsigma$, with $\chi|\varsigma \smile\!\!\bullet\, \zeta|\xi$, which is a representative of the abstract transition $\nu|\_$. In all other cases there exists a derivation $\chi|Q$, with $\chi|Q \smile\!\!\bullet\, \zeta|\xi$, which is a representative of the abstract transition $\nu|\_$. Either way $\zeta|\xi \models en(\nu|\_)$. □

Lemma D.8. *Let $u_i \models en(\nu_i)$ for i=1,2 with $\ell(\nu_1) = \overline{\ell(\nu_2)} \in \mathscr{H}$. Then $u_1|u_2 \models en(\nu_1|\nu_2)$, provided $u_1|u_2$ is a state of $\mathcal{U}$.*

Proof. We make case distinctions based on whether u_1 and u_2 are processes or derivations.

Suppose $u_i = P_i \models en(\nu_i)$ for i=1,2. Then $P_i = src(\chi_i)$ for representatives χ_i of ν_i. Now $src(\chi_1|\chi_2) = P_1|P_2$ and $\chi_1|\chi_2$ is a representative of $\nu_1|\nu_2$. So $P_1|P_2 \models en(\nu_1|\nu_2)$.

Suppose $u_i = \zeta_i \models en(\nu_i)$ for i=1,2. Then $\chi_i \smile\!\!\bullet\, \zeta_i$ for representatives χ_i of ν_i. So $\chi_1|\chi_2 \smile\!\!\bullet\, \zeta_1|\zeta_2$. Moreover, $\chi_1|\chi_2$ is a representative of $\nu_1|\nu_2$, and thus $\zeta_1|\zeta_2 \models en(\nu_1|\nu_2)$.

Suppose $P_1 \models en(\nu_1)$ and $\zeta_2 \models en(\nu_2)$. Then $P_1 = src(\chi_1)$ and $\chi_2 \smile\!\!\bullet\, \zeta_2$ for representatives χ_i of ν_i. Now $\chi_1|\chi_2 \smile\!\!\bullet\, P_1|\zeta_2$ and $\chi_1|\chi_2$ is a representative of $\nu_1|\nu_2$. So $P_1|\zeta_2 \models en(\nu_1|\nu_2)$.

The remaining case follows by symmetry. □

Lemma D.9. *If $u \models en(\nu)$, $c \in \mathscr{H}$ and $c \neq \ell(\nu) \neq \bar{c}$ then $u\backslash c \models en(\nu\backslash c)$.*

Proof. We make a case distinction based on whether u is processes P or a derivations ζ.

Suppose $P \models en(\nu)$. Then $P = src(\chi)$ for a representative χ of ν. Now $src(\chi\backslash c) = P\backslash c$ and $\chi\backslash c$ is a representative of $\nu\backslash c$. So $P\backslash c \models en(\nu\backslash c)$.

Suppose $\zeta \models en(\nu)$. Then $\chi \smile\!\!\bullet\, \zeta$ for a representative χ of ν. So $\chi\backslash c \smile\!\!\bullet\, \zeta\backslash c$. Moreover, $\chi\backslash c$ is a representative of $\nu\backslash c$, and thus $\zeta\backslash c \models en(\nu\backslash c)$. □

Lemma D.10. *If $u \models en(\nu)$ then $u[f] \models en(\nu[f])$.*

Proof. Similar to the proof of Lemma D.9. □

Proposition D.11. *If an abstract transition ν is enabled during a derivation ζ then ν is also enabled in $src(\zeta)$ as well as $target(\zeta)$.*

Proof. If ν is enabled during ζ then there is a representative χ of ν such that $\chi \frown\!\bullet\, \zeta$. By Observation C.5 $src(\chi) = src(\zeta)$, so ν is also enabled in $src(\zeta)$.

For the other statement, we apply structural induction on χ.

Let $\chi = \stackrel{\alpha}{\rightarrow}\chi'$. By Definition C.4 there is no derivation ζ with $\chi \frown\!\bullet\, \zeta$, which is a contradiction to our assumptions.

Let $\chi = A{:}\chi'$. Since $\chi \frown\!\bullet\, \zeta$, ζ has the form $A{:}\zeta'$ with $\chi' \frown\!\bullet\, \zeta'$. As $\nu = [\chi]_{\equiv} = [\chi']_{\equiv}$, ν is also enabled during ζ', and by induction ν is enabled in $target(\zeta') = target(\zeta)$.

Let $\chi = \chi'{+}P$. Since $\chi \frown\!\bullet\, \zeta$, ζ has the form $\zeta'{+}P$ with $\chi' \frown\!\bullet\, \zeta'$. As $\nu = [\chi]_{\equiv} = [\chi']_{\equiv}$, ν is also enabled during ζ', and by induction ν is enabled in $target(\zeta') = target(\zeta)$.

The case $\chi = P{+}\chi'$ follows by symmetry.

Let $\chi = \chi'\backslash c$. Since $\chi \frown\!\bullet\, \zeta$, ζ has the form $\zeta'\backslash c$ with $\chi' \frown\!\bullet\, \zeta'$. So $\nu' := [\chi']_{\equiv}$ is enabled during ζ', and by induction ν' is enabled in $target(\zeta')$. Moreover, $\ell(\nu') = \ell(\chi') \neq c, \bar{c}$. Consequently, using Lemma D.9, $\nu = \nu'\backslash c$ is enabled in $target(\zeta')\backslash c = target(\zeta)$.

Let $\chi = \chi'[f]$. Since $\chi \frown\!\bullet\, \zeta$, ζ has the form $\zeta'[f]$ with $\chi' \frown\!\bullet\, \zeta'$. So $\nu' := [\chi']_{\equiv}$ is enabled during ζ', and by induction ν' is enabled in $target(\zeta')$. Consequently, using Lemma D.10, $\nu = \nu'[f]$ is enabled in $target(\zeta')[f] = target(\zeta)$.

Let $\chi = \chi_1|P$. Since $\chi \frown\!\bullet\, \zeta$, Definition C.4 offers three possibilities for ζ:

- Suppose that ζ has the form $Q|\zeta_2$ with $src(\chi_1) = Q$ and $src(\zeta_2) = P$. Then $\nu_1 := [\chi_1]_{\equiv}$ is enabled in Q. Hence, by Lemma D.7, $\nu = ([\chi_1]_{\equiv})|\_$ is enabled in $P|target(\zeta_2) = target(\zeta)$.
- Suppose that ζ has the form $\zeta_1|P$ with $\chi_1 \frown\!\bullet\, \zeta_1$. Then $\nu_1 := [\chi_1]_{\equiv}$ is enabled during ζ_1, and by induction ν_1 is enabled in $target(\zeta_1)$. Consequently, using Lemma D.7, $\nu = \nu_1|\_$ is enabled in $target(\zeta_1)|P = target(\zeta)$.
- Suppose that ζ has the form $\zeta_1|\zeta_2$ with $\chi_1 \frown\!\bullet\, \zeta_1$ and $P = src(\zeta_2)$. Then $\nu_1 := [\chi_1]_{\equiv}$ is enabled during ζ_1, and by induction ν_1 is enabled in $target(\zeta_1)$. Consequently, using Lemma D.7, $\nu = \nu_1|\_$ is enabled in $target(\zeta_1)|target(\zeta_1) = target(\zeta)$.

The case $\chi = P|\chi_2$ follows by symmetry.

Let $\chi = \chi_1|\chi_2$ with $\ell(\chi) = \tau$. Then $\ell(\chi_1) = \overline{\ell(\chi_2)} \in \mathscr{H}$. Since $\chi \frown\!\bullet\, \zeta$, Definition C.4 offers three possibilities for ζ:

- Suppose ζ has the form $\zeta_1|P$ with $\chi_1 \frown\!\bullet\, \zeta_1$ and $P = src(\chi_2)$. Then $\nu_1 := [\chi_1]_{\equiv}$ is enabled during ζ_1, and by induction ν_1 is enabled in $target(\zeta_1)$. Consequently, using Lemma D.7, $\nu = \nu_1|\_$ is enabled in $target(\zeta_1)|P = target(\zeta)$.
- The case that ζ has the form $P|\zeta_2$ with $\chi_2 \frown\!\bullet\, \zeta_2$ and $P = src(\chi_1)$ follows by symmetry.

– Suppose ζ has the form $\zeta_1|\zeta_2$ with $\chi_i \frown\!\bullet\, \zeta_i$ for $i = 1, 2$. Then $\nu_i := [\chi_i]_\equiv$ is enabled during ζ_i, and by induction ν_i is enabled in $\mathit{target}(\zeta_i)$. Consequently, using Lemma D.8, $\nu = \nu_1|\nu_2$ is enabled in $\mathit{target}(\zeta_1)|\mathit{target}(\zeta_2) = \mathit{target}(\zeta)$.

Let $\chi = \chi_1|\chi_2$ with $\ell(\chi) \neq \tau$. Then $\ell(\chi) \in \mathscr{B}!$, since the case $\ell(\chi) = \ell(\nu) \in \mathscr{B}?$ cannot occur. So $\ell(\chi_1) = b!$ and $\ell(\chi_2) = b?$ for some $b \in \mathscr{B}$, or vice versa. W.l.o.g. we assume the first of these cases. Since $\chi \frown\!\bullet\, \zeta$, Definition C.4 offers five possibilities for ζ:

– Suppose ζ has the form $P|\zeta_2$ with $P = \mathit{src}(\chi_1)$ and $\mathit{src}(\chi_2) = \mathit{src}(\zeta_2)$. Then $\nu_1 := [\chi_1]_\equiv$ is enabled in P. Hence, by Lemma D.7, $\nu = ([\chi_1]_\equiv)|\_$ is enabled in $P|\mathit{target}(\zeta_2) = \mathit{target}(\zeta)$.
– The possibility that $\zeta = P|\zeta_2$ with $\chi_2 \frown\!\bullet\, \zeta_2$ and $P = \mathit{src}(\chi_1)$ is a special case of the last one.
– Suppose ζ has the form $\zeta_1|P$ with $\chi_1 \frown\!\bullet\, \zeta_1$ and $P = \mathit{src}(\chi_2)$. Then $\nu_1 := [\chi_1]_\equiv$ is enabled during ζ_1, and by induction ν_1 is enabled in $\mathit{target}(\zeta_1)$. Consequently, using Lemma D.7, $\nu = \nu_1|\_$ is enabled in $\mathit{target}(\zeta_1)|P = \mathit{target}(\zeta)$.
– Suppose ζ has the form $\zeta_1|\zeta_2$ with $\chi_1 \frown\!\bullet\, \zeta_1$ and $\mathit{src}(\chi_2) = \mathit{src}(\zeta_2)$. Then $\nu_1 := [\chi_1]_\equiv$ is enabled during ζ_1, and by induction ν_1 is enabled in $\mathit{target}(\zeta_1)$. Consequently, using Lemma D.7, $\nu = \nu_1|\_$ is enabled in $\mathit{target}(\zeta_1)|\mathit{target}(\zeta_2) = \mathit{target}(\zeta)$.
– The possibility $\zeta = \zeta_1|\zeta_2$ with $\chi_i \frown\!\bullet\, \zeta_i$ for $i = 1, 2$ is a special case of the previous one. □

Lemma D.12. *For derivations χ and ζ, $\chi \frown\!\bullet\, \zeta$ implies $\chi \not\equiv \zeta$.*

Proof. In case $\ell(\chi) \in \mathscr{B}?$ the statement is trivial by Definition D.1; so assume $\ell(\chi) \notin \mathscr{B}?$. We apply structural induction on χ.

Let $\chi = \stackrel{\alpha}{\rightarrow}\chi'$. By Definition C.4 there is no derivation ζ with $\chi \frown\!\bullet\, \zeta$, which is a contradiction to the antecedent.

Let $\chi = A{:}\chi'$. Since $\chi \frown\!\bullet\, \zeta$, ζ has the form $A{:}\zeta'$ with $\chi' \frown\!\bullet\, \zeta'$. Assume $A{:}\chi' \equiv A{:}\zeta'$. Then, by Definition D.1, $\chi' \equiv A{:}\chi' \equiv A{:}\zeta' \equiv \zeta'$, a contradiction to the induction hypothesis.

Let $\chi = \chi'{+}P$. Since $\chi \frown\!\bullet\, \zeta$, ζ has the form $\zeta'{+}P$ with $\chi' \frown\!\bullet\, \zeta'$. Assume $\chi'{+}P \equiv \zeta'{+}P$. Then, by Definition D.1, $\chi' \equiv \chi'{+}P \equiv \zeta'{+}P \equiv \zeta'$, a contradiction to the induction hypothesis.

The case $\chi = P{+}\chi'$ follows by symmetry.

Let $\chi = \chi'\backslash c$. Since $\chi \frown\!\bullet\, \zeta$, ζ has the form $\zeta'\backslash c$ with $\chi' \frown\!\bullet\, \zeta'$. Assume $\chi'\backslash c \equiv \zeta'\backslash c$. Then, by Observation D.3, $\chi' \equiv \zeta'$, a contradiction to the induction hypothesis.

Let $\chi = \chi'[f]$. Since $\chi \frown\!\bullet\, \zeta$, ζ has the form $\zeta'[f]$ with $\chi' \frown\!\bullet\, \zeta'$. Assume $\chi'[f] \equiv \zeta'[f]$. Then, by Observation D.3, $\chi' \equiv \zeta'$, a contradiction to the induction hypothesis.

Let $\chi = \chi_1|Q$. Note that $\ell(\chi_1) = \ell(\chi) \notin \mathscr{B}?$. Since $\chi \frown\!\bullet\, \zeta$, Definition C.4 offers three possibilities for ζ:

– Suppose that ζ has the form $P|\zeta_2$. By Observation D.2, $\chi_1|Q \not\equiv P|\zeta_2$.

– Suppose that ζ has the form $\zeta_1|u$ with $\chi_1 \frown\!\bullet\ \zeta_1$ (combining the cases $\zeta_1|Q$ and $\zeta_1|\zeta_2$). Assume $\chi_1|Q \equiv \zeta_1|u$. Then, by Observation D.2, $\chi_1 \equiv \zeta_1$, a contradiction to the induction hypothesis.

The case $\chi = P|\chi_2$ follows by symmetry.

Let $\chi = \chi_1|\chi_2$. As $\ell(\chi) \notin \mathscr{B}?$ either $\ell(\chi_1) \notin \mathscr{B}?$ or $\ell(\chi_2) \notin \mathscr{B}?$. W.l.o.g. we assume the first. Since $\chi \frown\!\bullet\ \zeta$, Definition C.4 offers seven possibilities for ζ, which can be summarised by the following 4 cases.

– Suppose ζ has the form $P|\zeta_2$. Then by Observation D.2 $\chi_1|\chi_2 \not\equiv P|\zeta_2$.
– The case where ζ has the form $\zeta_1|P$ with $src(\chi_2) = P$, $src(\chi_1) = src(\zeta_1)$ and $\ell(\chi_1) \in \mathscr{B}?$ cannot occur since $\ell(\chi_1) \notin \mathscr{B}?$.
– The case where ζ has the form $\zeta_1|\zeta_2$ with $\chi_2 \frown\!\bullet\ \zeta_2$, $src(\chi_1) = src(\zeta_1)$ and $\ell(\chi_1) \in \mathscr{B}?$ cannot occur either since $\ell(\chi_1) \notin \mathscr{B}?$.
– Finally, suppose ζ has the form $\zeta_1|u$ with $\chi_1 \frown\!\bullet\ \zeta_1$. Assume $\chi_1|\chi_2 \equiv \zeta_1|u$. Then, by Observation D.2, $\chi_1 \equiv \zeta_1$, a contradiction to the induction hypothesis. □

E Proof of Theorem 4.2

Theorem 4.2 makes a connection between the Y-just paths in $\mathcal{T}$ (or equivalently $\mathcal{S}$) and a weak fairness property for paths in $\mathcal{U}$. To establish its "⇐"-direction, we introduce two intermediate concepts: the Y-just paths in $\mathcal{U}$, for $Y \subseteq \mathscr{H}$, and the ν-enabled paths in $\mathcal{U}$, for abstract transitions ν. For the "⇒"-direction we introduce one intermediate concept: the ν-enabled paths in $\mathcal{S}$.

E.1 The "⇐"-direction

Observation E.1. For a derivation χ with $src(\chi) = P_1|P_2$ we have either that
– χ has the form $\chi_1|P_2$ with $src(\chi_1) = P_1$ and $target(\chi) = target(\chi_1)|P_2$, or
– χ has the form $\chi_1|\chi_2$ with $src(\chi_i) = P_i$ for i=1, 2
and $target(\chi) = target(\chi_1)|target(\chi_2)$, or
– χ has the form $P_1|\chi_2$ with $src(\chi_2) = P_2$ and $target(\chi) = P_1|target(\chi_1)$.

Hence all processes and derivations on a path π starting from a state $u_1|u_2$ of $\mathcal{U}$ have the form $\_|\_$. Let π_1 be the sequence of left- and π_2 the sequence of right-components of these processes and derivations, after (finite or infinite) subsequences of repeated elements are contracted to single elements. Then π_i is a path of u_i (i=1, 2) and together they constitute the *decomposition* of π, denoted $\pi \Rrightarrow \pi_1|\pi_2$.

Observation E.2. If $src(\chi)$=$P\backslash c$ then χ=$\chi'\backslash c$ with $src(\chi')$=P and $target(\chi)$=$target(\chi')\backslash c$.

Hence all processes and derivations on a path π of a state $u\backslash c$ in $\mathcal{U}$ have the form $\_\backslash c$. Let π' be the sequence obtained from π by stripping off these outermost occurrences of $\backslash c$. Then π' is a path of u, called the *decomposition* of π, denoted $\pi \Rrightarrow \pi'\backslash c$. In the same way one defines the decomposition of a path π of a state $u[f]$; notation $\pi \Rrightarrow \pi'[f]$.

Observation E.3. Let π be a path in $\mathcal{U}$. Then $\pi \Rrightarrow \pi_1|\pi_2$ implies $\widehat{\pi} \in \widehat{\pi}_1|\widehat{\pi}_2$. Likewise, if $\pi \Rrightarrow \pi'\backslash c$ or $\pi \Rrightarrow \pi'[f]$ then $\widehat{\pi}'$ is a decomposition of $\widehat{\pi}$.

Although in $\mathcal{S}$ the decomposition of a path from $P|Q$ need not be unique, in $\mathcal{U}$ it is. Armed with these definitions of decompositions of paths in $\mathcal{U}$, we define Y-justness, for $Y \subseteq \mathscr{H}$, on the paths of $\mathcal{U}$ in the exact same way as on $\mathcal{S}$ (see Definition 4.1).

Proposition E.4. *If a path π in $\mathcal{U}$ is Y-just, for $Y \subseteq \mathscr{H}$, then so is the path $\widehat{\pi}$ in $\mathcal{S}$.*

Proof. Define a path in $\mathcal{S}$ to be Y-*just*$_{\mathcal{U}}$, for $Y \subseteq \mathscr{H}$, if it has the form $\widehat{\pi}$ for a Y-just path π in $\mathcal{U}$. We show that the family of predicates Y-justness$_{\mathcal{U}}$ satisfies the five requirements of Definition 4.1.

- A finite Y-just$_{\mathcal{U}}$ path $\widehat{\pi}$, with π a just path in $\mathcal{U}$, ends in some state $\widehat{P}$. Since $\widehat{P} = P$, π ends in the same state. Hence that state admits actions from $Y \cup \mathscr{B}?$ only.
- Let π be a Y-just path in $\mathcal{U}$, so that $\widehat{\pi}$, and hence π, starts from a process $P|Q$. Then $\pi \Rrightarrow \pi_1|\pi_2$ for an X-just path π_1 of P and a Z-just path π_2 of Q such that $Y \supseteq X \cup Z$ and $X \cap \bar{Z} = \emptyset$. By Observation E.3 $\widehat{\pi} \in \widehat{\pi}_1|\widehat{\pi}_2$, where $\widehat{\pi}_1$ is X-just$_{\mathcal{U}}$ and $\widehat{\pi}_2$ is Z-just$_{\mathcal{U}}$.
- Let π be a Y-just path in $\mathcal{U}$ starting from a process $P\backslash c$. Then $\pi \Rrightarrow \pi'\backslash c$ for a $Y \cup \{c, \bar{c}\}$-just path π' of P. By Observation E.3 $\widehat{\pi}'$ is a decomposition of $\widehat{\pi}$, where $\widehat{\pi}'$ is $Y \cup \{c, \bar{c}\}$-just$_{\mathcal{U}}$.
- The case that $\widehat{\pi}$ is a path of $P[f]$ proceeds in exactly the same way.
- Each suffix of $\widehat{\pi}$ has the form $\widehat{\pi}'$ for π' a suffix of π. So if $\widehat{\pi}$ is Y-just$_{\mathcal{U}}$ because π is Y-just then π' must be Y-just, and hence $\widehat{\pi}$ is Y-just$_{\mathcal{U}}$.

Since Y-justness is the largest family of predicates on paths in $\mathcal{S}$ that satisfies those requirements, Y-justness$_{\mathcal{U}}$ of paths in $\mathcal{S}$ implies Y-justness of paths in $\mathcal{S}$. □

Definition E.5. *ν-enabledness*, for ν an abstract transition, is the smallest family of predicates on the paths in $\mathcal{U}$ such that

- a finite path is ν-enabled if its last state $Q \in \mathrm{Ex}_{\mathrm{ABC}}$ enables ν, i.e. $Q \models \mathit{en}(\nu)$;
- a path $\pi \Rrightarrow \pi_1|\pi_2$ is ν-enabled if either ν has the form $\nu_1|\_$ and π_1 is ν_1-enabled, or $\nu = \_|\nu_2$ and π_2 is ν_2-enabled, or $\nu = \nu_1|\nu_2$ and π_i is ν_i-enabled for $i = 1, 2$;
- a path $\pi \Rrightarrow \pi'\backslash c$ is ν-enabled if ν has the form $\nu'\backslash c$ and π' is ν'-enabled;
- a path $\pi \Rrightarrow \pi'[f]$ is ν-enabled if ν has the form $\nu'[f]$ and π' is ν'-enabled;
- and a path is ν-enabled if it has a suffix that is ν-enabled.

Proposition E.6. *Let π be a path in $\mathcal{U}$ and $Y \subseteq \mathscr{H}$. If, for all abstract transitions ν with $\ell(\nu) \notin Y$, π is not ν-enabled, then π is Y-just.*

Proof. Define a path π in $\mathcal{U}$ to be Y-*just*$_{\mathit{en}}$, for $Y \subseteq \mathscr{H}$, if it is ν-enabled for no abstract transition ν with $\ell(\nu) \notin Y$. Note that if π is Y-just$_{\mathit{en}}$, it is also Y'-just$_{\mathit{en}}$ for any $Y \subseteq Y' \subseteq \mathscr{H}$. We show that the family of predicates Y-justness$_{\mathit{en}}$, for $Y \subseteq \mathscr{H}$, satisfies the five requirements of Definition 4.1.

- Let π be a finite Y-just$_{en}$ path. Suppose the last state Q of π admits an action $\alpha \notin Y \cup \mathscr{B}?$. Then $Q \models en(\nu)$ for an abstract transition ν with $\ell(\nu) = \alpha \notin Y \subseteq Y \cup \mathscr{B}?$. So π is ν-enabled, contradicting the Y-justness$_{en}$ of π.
- Suppose π is a Y-just$_{en}$ path of a process $P|Q$. Then Y includes all labels of abstract transitions ν for which π is ν-enabled. By Observation E.1 there are paths π_i for $i{=}1,2$ with $\pi \Rrightarrow \pi_1|\pi_2$. Let X be the set of labels of abstract transitions ν for which π_1 is ν-enabled, and let Z be the set of labels of abstract transitions ν for which π_2 is ν-enabled. If π_1 is ν-enabled then π is $\nu|\_$-enabled by Definition E.5. Since $\ell(\nu|\_) = \ell(\nu)$ this implies that $X \subseteq Y$. In the same way it follows that $Z \subseteq Y$.
  Now suppose that $X \cap \bar{Z} \neq \emptyset$. Then π_i is ν_i-enabled, for $i{=}1,2$, for abstract transitions ν_i with $\ell(\nu_1) = c \in \mathscr{H}$ and $\ell(\nu_2) = \bar{c}$. So by Definition E.5 π is $\nu_1|\nu_2$-enabled, in contradiction with $\ell(\nu_1|\nu_2) = \tau \notin Y \subseteq \mathscr{H}$. We therefore conclude that $X \cap \bar{Z} = \emptyset$.
  By definition, π_1 is X-just$_{en}$ and π_2 is Z-just$_{en}$.
- Suppose π is a Y-just$_{en}$ path of a process $P\backslash c$. Then Y includes all labels of abstract transitions ν for which π is ν-enabled. By Observation E.2 there is a path π' with $\pi \Rrightarrow \pi'\backslash c$. Let X be the set of labels of abstract transitions ν for which π' is ν-enabled. If π' is ν-enabled and $c \neq \ell(\nu) \neq \bar{c}$ then π is $\nu\backslash c$-enabled by Definition E.5. Since $\ell(\nu\backslash c) = \ell(\nu)$ this implies that $X \setminus \{c, \bar{c}\} \subseteq Y$. It follows that π' is X-just$_{en}$, and hence $Y \cup \{c, \bar{c}\}$-just$_{en}$.
- Suppose π is a Y-just$_{en}$ path of a process $P[f]$. Then Y includes all labels of abstract transitions ν for which π is ν-enabled. By the remark after Observation E.2, there is a path π' with $\pi \Rrightarrow \pi'[f]$. Let X be the set of labels of abstract transitions ν for which π' is ν-enabled. If π' is ν-enabled then π is $\nu[f]$-enabled by Definition E.5. Since $\ell(\nu[f]) = f(\ell(\nu))$ this implies that $f(X) \subseteq Y$. It follows that π' is X-just$_{en}$, and hence $f^{-1}(Y)$-just$_{en}$.
- Suppose π' is a suffix of an Y-just$_{en}$ path π. Then Y includes all labels of abstract transitions ν for which π is ν-enabled. By the last clause of Definition E.5, Y thereby includes all labels of abstract transitions ν for which π' is ν-enabled. Hence π' is Y-just$_{en}$.

Since Y-justness is the largest family of predicates that satisfies those requirements, Y-justness$_{en}$ implies Y-justness. □

Henceforth, we write $\pi \models \phi$ if the LTL formula ϕ holds for the path π in $\mathcal{U}$, that is, if π satisfies ϕ. Note that a finite path satisfies $\mathbf{FG}\,\phi$ if ϕ holds in its last state.

Proposition E.7. *If π is ν-enabled then $\pi \models \mathbf{FG}\, en(\nu)$.*

Proof. We apply induction on ν-enabledness of a path π in $\mathcal{U}$, using the five clauses of Definition E.5.

Suppose π is ν-enabled because it is finite and its last state $Q \in \mathrm{Ex}_{\mathrm{ABC}}$ enables ν. Then $\pi \models \mathbf{FG}\, en(\nu)$.

Suppose $\pi \Rrightarrow \pi_1|\pi_2$ is ν-enabled because $\nu = \nu_1|\_$ and π_1 is ν_1-enabled. By induction $\pi_1 \models \mathbf{FG}\, en(\nu_1)$. Let $\pi_1' = u_0 u_1 u_2 \ldots$ be a suffix of π_1 with $\pi_1' \models \mathbf{G}\, en(\nu_1)$. Then, for all $i \geq 0$, $u_i \models en(\nu_1)$ and thus $u_i|v \models en(\nu_1|\_)$ for any state

$u_i|v$ of $\mathcal{U}$ by Lemma D.7. Therefore $\pi \models \mathbf{FG}\, en(\nu)$. The case that $\nu = \_|\nu_2$ and π_2 is ν_2-enabled goes likewise.

Suppose $\pi \Rrightarrow \pi_1|\pi_2$ is ν-enabled because $\nu = \nu_1|\nu_2$ and π_i is ν_i-enabled for $i = 1, 2$. Then $\ell(\nu_1) = \overline{\ell(\nu_2)} \in \mathscr{H}$. By induction $\pi_i \models \mathbf{FG}\, en(\nu_i)$. Let $\pi_1' = u_0u_1u_2\ldots$ and $\pi_2' = v_0v_1v_2\ldots$ be (finite or infinite) suffixes of π_1 and π_2 with $\pi_i' \models \mathbf{G}\, en(\nu_i)$ for $i = 1, 2$. Then, for all $j, k \geq 0$, $u_j \models en(\nu_1)$ and $v_k \models en(\nu_2)$ and thus $u_j|v_k \models en(\nu_1|\nu_2)$ by Lemma D.8, whenever $u_j|v_k$ is a state of $\mathcal{U}$. Therefore $\pi \models \mathbf{FG}\, en(\nu)$.

Suppose $\pi \Rrightarrow \pi'\backslash c$ (with $c \in \mathscr{H}$) is ν-enabled because $\nu = \nu'\backslash c$ and π' is ν'-enabled. Then $c \neq \ell(\nu') \neq \bar{c}$. By induction, $\pi' \models \mathbf{FG}\, en(\nu')$. Using Lemma D.9, one finds $\pi \models \mathbf{FG}\, en(\nu)$.

Suppose $\pi \Rrightarrow \pi'[f]$ is ν-enabled because $\nu = \nu[f]$ and π' is ν'-enabled. By induction $\pi' \models \mathbf{FG}\, en(\nu')$. Using Lemma D.10, one finds $\pi \models \mathbf{FG}\, en(\nu)$.

Suppose π is ν-enabled because it has a suffix π' that is ν-enabled. Then, by induction, $\pi' \models \mathbf{FG}\, en(\nu)$. Hence $\pi \models \mathbf{FG}\, en(\nu)$. □

The following result is the "⇐"-direction of Theorem 4.2.

Proposition E.8. *If π is a path in $\mathcal{U}$ with $\pi \models \mathbf{FG}\, en(\nu) \Rightarrow \mathbf{GF}\, \nu$ for each abstract transition ν with $\ell(\nu) \in \mathscr{B}! \cup \{\tau\}$, then $\widehat{\pi}$ is just in the sense of Definition 4.1.*

Proof. By definition no state $P \in \mathrm{Ex}_{\mathrm{ABC}}$ satisfies ν. So, any state of $\mathcal{U}$ that could satisfy ν as well as $en(\nu)$ needs to be a derivation ζ. Assume $\zeta \models en(\nu)$. Then there is a representative χ of ν, i.e., $\nu = [\chi]_{\equiv}$, with $\chi \smile\!\!\bullet\, \zeta$. Using Lemma D.12, we get $\nu = [\chi]_{\equiv} \neq [\zeta]_{\equiv}$. In case ζ would also satisfy ν, we have, by definition, $\nu = [\zeta]_{\equiv}$, which is a contradiction. Hence there is no abstract transition ν and state u in $\mathcal{U}$ for which the propositions $en(\nu)$ and ν both hold. Consequently, the formula $\mathbf{FG}\, en(\nu) \Rightarrow \mathbf{GF}\, \nu$ is equivalent to $\neg\mathbf{FG}\, en(\nu)$.

Let π be a path in $\mathcal{U}$ with $\pi \models \neg\mathbf{FG}\, en(\nu)$ for each ν with $\ell(\nu) \in \mathscr{B}! \cup \{\tau\}$. Then, by Proposition E.7, π is ν-enabled for no ν with $\ell(\nu) \in \mathscr{B}! \cup \{\tau\}$. Hence, by Proposition E.6, π is $\mathscr{H}$-just. Therefore, by Proposition E.4, $\widehat{\pi}$ is $\mathscr{H}$-just, and hence just. □

E.2 The "⇒"-direction

E.2.1 On the targets of derivations enabling abstract transitions

Lemma E.9. *If $\zeta \models en(\nu|\_)$, $\zeta \models en(\_|\nu)$ or $\zeta \models en(\nu_1|\nu_2)$ for a derivation ζ and abstract transitions ν, ν_1, ν_2, then $target(\zeta)$ has the form $P_1|P_2$.*

Proof. Since $\zeta \models en(\nu)$ means that $\chi \smile\!\!\bullet\, \zeta$ for a representative χ of ν, the lemma can be rephrased as: "If $\chi \smile\!\!\bullet\, \zeta$ for a representative χ of an abstract transition $\nu|\_$, $\_|\nu$ or $\nu_1|\nu_2$, $target(\zeta)$ has the form $P_1|P_2$." We prove this statement by structural induction on χ.

- Let χ be $\chi'|P$, $P|\chi'$ or $\chi_1|\chi_2$. By the definition of $\smile\!\!\bullet$, ζ must then have the form $Q|\zeta'$, $\zeta'|Q$ or $\zeta_1|\zeta_2$. In each of these cases $target(\zeta)$ has the form $P_1|P_2$.

- Let $\chi = \chi'+P$. Then ζ must have the form $\zeta'+P$ with $\chi' \rightharpoonup \zeta'$ by the definition of $\rightharpoonup$. Since χ is a representative of an abstract transition $\nu|\_$ or $\nu_1|\nu_2$, so is χ'. By induction, $\mathit{target}(\zeta')$ has the form $P_1|P_2$. By rule (SUM-L) $\mathit{target}(\zeta) = \mathit{target}(\zeta')$, and thus of the form $P_1|P_2$.
- The cases $\chi = P+\chi'$ and $\chi = A{:}\chi'$ proceed likewise.

As χ represents an abstract transition $\nu|\_$ or $\nu_1|\nu_2$, it cannot have any other form. □

Lemma E.10. *If $\zeta \models \mathit{en}(\nu\backslash c)$ or $\zeta \models \mathit{en}(\nu[f])$, then $\mathit{target}(\zeta)$ has the form $P\backslash c$, and $P[f]$, resp.*

Proof. The first statement can be rephrased as: "If $\chi \rightharpoonup \zeta$ for a representative χ of an abstract transition $\nu\backslash c$, then $\mathit{target}(\zeta)$ has the form $P\backslash c$." We prove this statement by structural induction on χ.

- Let $\chi = \chi'\backslash c$. By the definition of $\rightharpoonup$ we have $\zeta = \zeta'\backslash c$ with $\chi' \rightharpoonup \zeta'$. Hence $\mathit{target}(\zeta)$ has the form $P\backslash c$.
- Let $\chi = A{:}\chi'$. Then ζ must have the form $A{:}\zeta'$ with $\chi' \rightharpoonup \zeta'$ by the definition of $\rightharpoonup$. Since χ is a representative of an abstract transition $\nu\backslash c$, so is χ'. By induction, $\mathit{target}(\zeta')$ has the form $P\backslash c$. By rule (REC) $\mathit{target}(\zeta) = \mathit{target}(\zeta')$, and thus of the form $P\backslash c$.
- The cases $\chi = \chi'+P$ and $\chi = P+\chi'$ proceed likewise, using (SUM-L) and (SUM-R).
- As χ represents an abstract transition $\nu\backslash c$, it cannot have any other form.

The proof of the second statement proceeds likewise. □

E.2.2 Decomposing enabled abstract transitions

Observation E.11. *Using Observation E.1, any representative χ of $\nu|\_$ such that $\mathit{src}(\chi) = P_1|P_2$ is of the form $\chi'|P_2$ with χ' a representative of ν, or $\chi'|\varsigma$ with χ' a representative of ν and $\ell(\chi') = \ell(\chi) \in \mathscr{B}!$. Moreover $\mathit{src}(\chi') = P_1$.*

Lemma E.12. *If $u_1|u_2 \models \mathit{en}(\nu|\_)$ for a state $u_1|u_2$ of $\mathcal{U}$, then $u_1 \models \mathit{en}(\nu)$.*

Proof. We make a case distinction based on whether u_i is a process or a derivation.

Suppose $P_1|P_2 \models \mathit{en}(\nu|\_)$. Then $P_1|P_2 = \mathit{src}(\chi)$ for a representative χ of $\nu|\_$. By Observation E.11, it has the form $\chi'|v$ with χ' a representative of ν and $\mathit{src}(\chi') = P_1$. Therefore $P_1 \models \mathit{en}(\nu)$.

Suppose $P_1|\zeta_2 \models \mathit{en}(\nu|\_)$ with $\mathit{src}(P_1|\zeta_2) = P_1|P_2$. Then $\chi \rightharpoonup P_1|\zeta_2$ for a representative χ of $\nu|\_$. By Observation C.5 $\mathit{src}(\chi) = \mathit{src}(P_1|\zeta_2) = P_1|P_2$. So, by Observation E.11 it is has the form $\chi'|v$, with χ' a representative of ν and $\mathit{src}(\chi') = P_1$. Therefore $P_1 \models \mathit{en}(\nu)$.

Suppose $\zeta_1|u_2 \models \mathit{en}(\nu|\_)$ with $\mathit{src}(\zeta_1|u_2) = P_1|P_2$. Then $\chi \rightharpoonup \zeta_1|u_2$ for a representative χ of $\nu|\_$. By Observation C.5 $\mathit{src}(\chi) = \mathit{src}(\zeta_1|u_2) = P_1|P_2$. So, by Observation E.11 it is has either the form $\chi'|P_2$ or $\chi'|\varsigma$ ($\ell(\chi') \in \mathscr{B}!$), with χ' a representative of ν. So, $\chi'|P_2 \rightharpoonup \zeta_1|u_2$ or $\chi'|\varsigma \rightharpoonup \zeta_1|u_2$ and hence, by Definition C.4, $\chi' \rightharpoonup \zeta_1'$. Since χ' a representative of ν, $\zeta_1 \models \mathit{en}(\nu)$. □

Lemma E.13. *If $u_1|u_2 \models en(\nu_1|\nu_2)$ for a state $u_1|u_2$ of $\mathcal{U}$, then $u_i \models en(\nu_i)$ (i=1, 2).*

Proof. We make a case distinction based on whether u_i is a process or a derivation.

Suppose $P_1|P_2 \models en(\nu_1|\nu_2)$. Then $P_1|P_2 = src(\chi)$ for a representative χ of $\nu_1|\nu_2$. By Observation E.1, it has the form $v_1|v_2$. Since it is a representative of $\nu_1|\nu_2$, it has the form $\chi_1|\chi_2$, with χ_i a representative of ν_i (i=1, 2). Furthermore, $src(\chi_i) = P_i$. It follows that $P_i \models en(\nu_i)$ (i=1, 2).

Suppose $P_1|\zeta_2 \models en(\nu_1|\nu_2)$ with $src(P_1|\zeta_2) = P_1|P_2$. Then $\chi \leadsto P_1|\zeta_2$ for a representative χ of $\nu_1|\nu_2$. By Observation C.5 $src(\chi) = src(P_1|\zeta_2) = P_1|P_2$. So, by Observation E.1, χ has the form $v_1|v_2$. Since it is a representative of $\nu_1|\nu_2$, it has the form $\chi_1|\chi_2$, with χ_i a representative of ν_i (i=1, 2). Moreover, $\ell(\chi_1) = \ell(\nu_1) = \overline{\ell(\nu_2)} = \overline{\ell(\chi_2)} \in \mathscr{H}$. So $\chi_1|\chi_2 \leadsto P_1|\zeta_2$ and by Definition C.4 $src(\chi_1) = P_1$ and $\chi_2 \leadsto \zeta_2$. Since χ_i (i=1, 2) is a representative of ν_i, $P_1 \models en(\nu_1)$ and $\zeta_2 \models en(\nu_2)$.

The case $\zeta_1|P_2$ follows by symmetry.

Suppose $\zeta_1|\zeta_2 \models en(\nu_1|\nu_2)$ with $src(\zeta_1|\zeta_2) = P_1|P_2$. Then $\chi \leadsto \zeta_1|\zeta_2$ for a representative χ of $\nu_1|\nu_2$. By Observation C.5 $src(\chi) = src(P_1|\zeta_2) = P_1|P_2$. So, by Observation E.1, χ has the form $v_1|v_2$. Since it is a representative of $\nu_1|\nu_2$, it has the form $\chi_1|\chi_2$, with χ_i a representative of ν_i (i=1, 2). Moreover, $\ell(\chi_1)=\ell(\nu_1)=\overline{\ell(\nu_2)}=\overline{\ell(\chi_2)}\in\mathscr{H}$. So $\chi_1|\chi_2 \leadsto \zeta_1|\zeta_2$ and by Definition C.4 $\chi_i \leadsto \zeta_i$ (i=1, 2). Since χ_i is a representative of ν_i, $\zeta_i \models en(\nu_i)$. □

Lemma E.14. *If $u \models en(\nu\backslash c)$ for a state u of $\mathcal{U}$ of the form $u'\backslash c$, then $u' \models en(\nu)$.*

Proof. We make a case distinction based on whether u is a processes P or a derivation ζ.

Suppose $P \models en(\nu\backslash c)$. Then $P = src(\chi)$ for a representative χ of $\nu\backslash c$. Since, by assumption, P is of the form $P'\backslash c$, Observation E.2 says that χ is of the form $\chi'\backslash c$ with $src(\chi') = P'$. Since $\chi = \chi'\backslash c$ is a representative of $\nu\backslash c$, χ' must be a representative of ν. Hence $P' \models en(\nu)$.

Suppose $\zeta \models en(\nu\backslash c)$. Then $\chi \leadsto \zeta$ for a representative χ of $\nu\backslash c$. Since, by assumption, ζ is of the form $\zeta'\backslash c$, $src(\zeta)$ must be of the form $P'\backslash c$. By Observation C.5 $src(\chi) = src(\zeta)$, so by Observation E.2 χ is of the form $\chi'\backslash c$. Since $\chi = \chi'\backslash c$ is a representative of $\nu\backslash c$, χ' must be a representative of ν. As $\chi'\backslash c \leadsto \zeta'\backslash c$, we have $\chi' \leadsto \zeta'$. Thus $\zeta' \models en(\nu)$. □

Lemma E.15. *If $u[f] \models en(\nu[f])$ for a state u of $\mathcal{U}$, then $u \models en(\nu)$.*

Proof. Exactly as above, using an analogue of Observation E.2 for relabelling. □

E.2.3 ν-enabled paths in $\mathcal{S}$

Definition E.16. A path ρ in $\mathcal{S}$ is *ν-enabled* for an abstract transition ν, if either it is finite and its last state $Q \in \mathrm{Ex}_{\mathrm{ABC}}$ satisfies $Q \models en(\nu)$, or it is infinite and has a suffix ρ' such that $\zeta \models en(\nu)$ for all derivations ζ with $\widehat{\zeta}$ a transition in ρ'.

Lemma E.17. *If a path ρ in $\mathcal{S}$ is Y-just and ν-enabled then $\ell(\nu) \in Y$.*

Proof. If a finite path ρ in $\mathcal{S}$ is ν-enabled, its last state $Q \in \mathrm{Ex}_{\mathrm{ABC}}$ satisfies $en(\nu)$, and thus $Q \xrightarrow{\ell(\nu)}$. The first clause of Definition 4.1 (Y-justness) tells that $\ell(\nu) \in Y$.

For infinite paths ρ, we apply structural induction on ν. Let ρ be an infinite path that is Y-just and ν-enabled, and let ρ' be a suffix of ρ, such that $\zeta \models en(\nu)$ for each derivation ζ with $\widehat{\zeta}$ a transition in ρ'. Moreover, $P \models en(\nu)$ for each state of P on ρ', using Proposition D.11 and the definition of $\widehat{\ }$. Let ζ_0 be a derivation of the first transition in ρ', and let ρ'' be the suffix of ρ' starting from $Q := target(\zeta_0)$.

Let $\nu = \xrightarrow{\alpha} P$ for $\alpha \in Act$ and $P \in \mathrm{Ex}_{\mathrm{ABC}}$. Since no representative χ of ν is concurrent with any derivation ζ, it follows that ρ' contains no transitions, and hence consists of a single state only. This contradicts the presumed infinity of ρ.

Let $\nu = \nu_1|\_$. Since $\zeta_0 \models en(\nu_1|\_)$, by Lemma E.9 Q has the form $P_1|P_2$. By Definition 4.1 ρ'' can be decomposed into an X-just path ρ_1 of $\widehat{P}_1 = P_1$ and a Z-just path ρ_2 of $\widehat{P}_2 = P_2$ such that $Y \supseteq X \cup Z$ and $X \cap \bar{Z} = \emptyset$. By Observation E.1 and the definition of $\widehat{\ }$, all states u of ρ'' as well as all derivations u of transitions in ρ'' have the form $u_1|u_2$. Since each such $u_1|u_2$ satisfies $en(\nu_1|\_)$, by Lemma E.12 $u_1 \models en(\nu_1)$. It follows that $u_1 \models en(\nu_1)$ for each state u_1 in ρ_1 and for each derivation u_1 of a transition in ρ_1. Hence ρ_1 is ν_1-enabled. By induction $\ell(\nu_1) \in X$. So $\ell(\nu) = \ell(\nu_1) \in X \subseteq Y$.

The case $\nu = \_|\nu_2$ follows by symmetry.

Let $\nu = \nu_1|\nu_2$. Then $\ell(\nu_1) = \overline{\ell(\nu_2)} \in \mathscr{H}$. Since $\zeta_0 \models en(\nu_1|\nu_2)$, by Lemma E.9 Q has the form $P_1|P_2$. By Definition 4.1 ρ'' can be decomposed into an X-just path ρ_1 of $\widehat{P}_1 = P_1$ and a Z-just path ρ_2 of $\widehat{P}_2 = P_2$ such that $Y \supseteq X \cup Z$ and $X \cap \bar{Z} = \emptyset$. By Observation E.1 and the definition of $\widehat{\ }$, all states u of ρ'' as well as all derivations u of transitions in ρ'' have the form $u_1|u_2$. Since each such $u_1|u_2$ satisfies $en(\nu_1|\nu_2)$, by Lemma E.13 $u_i \models en(\nu_i)$ (i=1, 2). It follows that $u_i \models en(\nu_i)$ for each state u_i in ρ_i and for each derivation u_i of a transition in ρ_i. Hence ρ_i is ν_i-enabled. By induction $\ell(\nu_1) \in X$ and $\ell(\nu_1) = \overline{\ell(\nu_2)} \in \bar{Z}$, in contradiction with $X \cap \bar{Z} = \emptyset$. Therefore, this case cannot occur.

Let $\nu = \nu' \backslash c$ (with $c \in \mathscr{H}$). Then $c \neq \ell(\nu') \neq \bar{c}$. Since $\zeta_0 \models en(\nu' \backslash c)$, by Lemma E.10 Q has the form $P \backslash c$. By Definition 4.1 ρ'' can be decomposed into a $Y \cup \{c\}$-just path ρ''' of $\widehat{P} = P$. By Observation E.2 all derivations ζ of transitions in ρ'' have the form $\zeta' \backslash c$. Since each such $\zeta' \backslash c$ satisfies $en(\nu' \backslash c)$, by Lemma E.14 $\zeta' \models en(\nu')$. It follows that $\zeta' \models en(\nu')$ for each derivation ζ' of a transition in ρ'''. Hence ρ''' is ν'-enabled. By induction $\ell(\nu') \in Y \cup \{c\}$. Since $\ell(\nu') \neq c$ we obtain $\ell(\nu) = \ell(\nu') \in Y$.

Let $\nu = \nu'[f]$. Since $\zeta_0 \models en(\nu'[f])$, by Lemma E.10 Q has the form $P[f]$. By Definition 4.1 ρ'' can be decomposed into a $f^1(Y)$-just path ρ''' of $\widehat{P} = P$. By the relabelling variant of Observation E.2 all derivations ζ of transitions in ρ'' have the form $\zeta'[f]$. Since each such $\zeta'[f]$ satisfies $en(\nu'[f])$, by Lemma E.15 $\zeta' \models en(\nu')$. It follows that $\zeta' \models en(\nu')$ for each derivation ζ' of a transition in ρ'''. Hence ρ''' is ν'-enabled. By induction $\ell(\nu') \in f^1(Y)$. So $\ell(\nu) = f(\ell(\nu')) \in Y$. $\square$

The following result directly implies the "⇒"-direction of Theorem 4.2.

Proposition E.18. *Let ρ be a just path in $\mathcal{S}$. Then $\rho = \widehat{\pi}$ for a path π in $\mathcal{U}$ that satisfies $\pi \not\models \mathbf{FG}\, en(\nu)$ for each abstract transition ν with $\ell(\nu) \in \mathscr{B}! \cup \{\tau\}$.*

Proof. First, suppose that ρ is a finite just path in $\mathcal{S}$. By Definition 4.1 it ends in a state $Q \in \mathrm{Ex}_{\mathrm{ABC}}$ that admits actions from $\mathscr{H} \cup \mathscr{B}?$ only. Pick any path π in $\mathcal{U}$ with $\widehat{\pi} = \rho$. Then π ends in Q as well. Hence $\pi \models \mathbf{FG}\, en(\nu)$ for no abstract transition ν with $\ell(\nu) \in \mathscr{B}! \cup \{\tau\}$.

Next, consider the case that ρ is infinite. There are countably many abstract transitions. Let $(\nu_i)_{i=0}^{\infty}$ be an enumeration of the abstract transitions ν with $\ell(\nu) \in \mathscr{B}! \cup \{\tau\}$, such that each such ν occurs infinitely often in this sequence.

With induction on $i \in \mathbb{N}$, we construct finite paths π_i in $\mathcal{U}$ such that π_i will be a strict prefix of π_j when $i < j$, and $\widehat{\pi}_i$ is a prefix of ρ for each $i \in \mathbb{N}$.

Let π_0 be an arbitrary finite path in $\mathcal{U}$ with $\widehat{\pi}_0$ a prefix of ρ. Given π_i, let ζ_i be an arbitrary derivation such that $\zeta_i \not\models en(\nu_i)$ and $\widehat{\zeta_i}$ occurs in ρ past the prefix $\widehat{\pi}_i$. Such a ζ_i must exists, as otherwise ρ would be ν_i-enabled, which contradicts Lemma E.17. (Remember that by assumption ρ is just.)

We obtain π_{i+1} by extending π_i in a way such that $\widehat{\pi}_{i+1}$ is a prefix of ρ up to and including $\widehat{\zeta_i}$ and its target state; the last derivation of π_{i+1} is set to ζ_i. All derivations different from ζ_i that are not part of π_i can be chosen arbitrarily, under the restriction that $\widehat{\pi}_{i+1}$ is a prefix of ρ.

Now $\pi := \lim_{i\to\infty} \pi_i$ exists and satisfies $\rho = \widehat{\pi}$. By construction, $\pi \models \neg\mathbf{FG}\, en(\nu)$ for any abstract transition ν with $\ell(\nu) \in \mathscr{B}! \cup \{\tau\}$. □