Fair Testing

Arend Rensink
Department of Computer Science, University of Twente
Postbus 217, NL–7500 AE Enschede
rensink@cs.utwente.nl

Walter Vogler
Institut für Informatik, University of Augsburg
D–86135 Augsburg
vogler@uni-augsburg.de

December 23, 2005

Abstract

In this paper we present a solution to the long-standing problem of characterising the coarsest liveness-preserving pre-congruence with respect to a full (TCSP-inspired) process algebra. In fact, we present two distinct characterisations, which give rise to the same relation: an operational one based on a De Nicola-Hennessy-like testing modality which we call *should-testing*, and a denotational one based on a refined notion of *failures*.

One of the distinguishing characteristics of the should-testing pre-congruence is that it abstracts from divergences in the same way as Milner's observation congruence, and as a consequence is strictly coarser than observation congruence. In other words, should-testing has a built-in fairness assumption. This is in itself a property long sought-after; it is in notable contrast to the well-known *must-testing* of De Nicola and Hennessy (denotationally characterised by a combination of failures and divergences), which treats divergence as catrastrophic and hence is incompatible with observation congruence.

Due to these characteristics, should-testing supports modular reasoning and allows to use the proof techniques of observation congruence, but also supports additional laws and techniques. Moreover, we show decidability of should-testing (on the basis of the denotational characterisation). Finally, we demonstrate its advantages by the application to a number of examples, including a scheduling problem, a version of the Alternating Bit-protocol, and fair lossy communication channels.

Contents

1 Introduction

Over the years, the specification and analysis of distributed systems by means of process-algebraic languages and theories has become an established field of theoretical and applied research. Basic process algebraic theories like CCS (Milner [28]), CSP (Hoare [21]) and ACP (Baeten and Weijland [1]) as well as standardised specification formalisms like LOTOS (Bolognesi and Brinksma [5], ISO [22]) are being routinely applied.

One of the most interesting and fruitful areas of research in process algebra is that of behavioural equalities and pre-orders. Such a relation defines formally when one process is a correct implementation of another; ideally, the relation should regard processes as comparable if and only if one can replace the other as far as 'observable behaviour' is concerned. However, one of the insights that has been gained in the past two decades is that there does not really exist one canonical notion of observable behaviour; rather, depending on the formalisation of observability, many different notions of behavioural equivalence or inclusion arise (the reader may consult Van Glabbeek [39, 38] for an overview). Of course, other criteria apply as well, such as for example the availability of a mathematically tractable and well-understood theory, so that in practice a compromise between the various requirements must be found.

In their seminal paper [17], De Nicola and Hennessy present a framework for defining pre-orders that is widely acknowledged as a realistic scenario for *system testing*. They use their framework to define the *must-testing* and *may-testing* pre-orders, respectively, both based on a different notion of what one can observe when a system is submitted to an arbitrary test. Both of these pre-orders are backed up by a relatively simple and appealing denotational semantics, combining notions of *failure* and *divergence* in the case of must-testing (see Brookes, Hoare and Roscoe [13, 14]), and based on *traces* in the case of may-testing.

Another very successful family of pre-orders (most of them actually equivalences) arises out of the principle of (mutual) *simulation* of systems. The prime representatives of this family are *bisimilarity* (Milner [26], Park [33]) and especially also *observation congruence* (Milner [28]).

In favour of testing

We recall that, at least in principle, bisimulations provide the finer equivalences that keep track of the branching structure of behaviours in detail; moreover, they have a rather elegant proof theory based on the construction of bisimulation relations. Testing equivalences and pre-orders developed following the recipe of De Nicola and Hennessy are in principle coarser.

The higher resolution power of bisimulations can, in fact, be undesirable in practice. For instance, the transition systems B_1 and B_2 in Figure 1.1 are not observation congruent; but they are must-testing equivalent, and indeed one would sometimes like to implement behaviour B_1 by

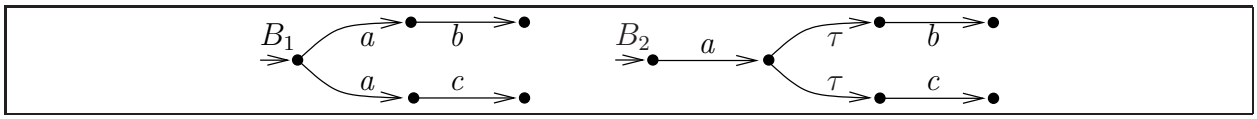


Figure 1.1: Must-testing equivalent behaviours that are not observation congruent.

B_2, resolving the choice between the two a-actions in B_1 internally through the internal τ-actions in B_2, and not through interaction with the environment. The idea is that, as the environment cannot influence the choice in either case, this should make no difference to the observable behaviour.

It should be mentioned that, at least to some degree, successful techniques for observation congruence can still be used if one is actually interested in a testing equivalence (or pre-order),

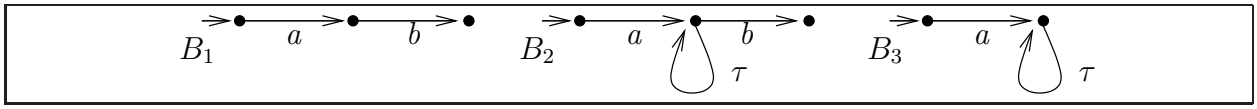


Figure 1.2: The difference between observation congruence and must-testing equivalence.

provided that the latter is really coarser than the former. Processes shown observation congruent by providing a bisimulation or by using suitable axioms are also seen to be testing equivalent. And, as e.g. Valmari argues in [34], the efficient reduction of a process to a minimal observation congruent one provides also a reduction w.r.t. testing equivalence, which then can be improved further; thus, one can only fare better when using the testing approach.

Finally, one may argue that (non-symmetric) pre-orders have an intrinsic advantage over equivalences: They better reflect the idea of an *implementation relation* in which moving to a smaller element expresses an implementation choice — usually involving some sort of deterministic reduction of the specified behaviour.

In favour of observation congruence

On the other hand, there is an important feature of observation congruence that classical testing does not provide: it incorporates a particular notion of *fairness*. For instance, the behaviours B_1 and B_2 shown in Figure 1.2 are observation congruent. Observation congruence works on the principle that the τ-loop of B_2 is executed an arbitrary but only finite number of times, in this case implying that eventually action b will be enabled. Such identification of behaviour can be very useful in practice: for example, when proving properties of systems with lossy communication media. In such cases τ-loops represent an unbounded but finite number of message losses. Interesting proofs of protocol correctness based on this principle are given by Larsen and Milner in [24] and by Brinksma in [9]. Most of the standard testing-preorders, on the other hand, are based on the interpretation of τ-loops as *divergences*, making them quasi-observable as a chaotic or under-specified process. In this interpretation, all behaviour after a divergence is ignored. As a consequence, the standard testing-preorders are only coarser than observation congruence for divergence-free processes, but not in general. For instance, B_1 and B_2 in Figure 1.2 are not must-testing equivalent; instead, B_2 is must-testing equivalent to B_3.

Joining strengths

It is of practical interest to have a testing pre-order that combines the ability to shift and reduce nondeterminism typical for testing with the capacity to model fairness. In particular, this should give a testing pre-order that is really compatible with observation congruence, giving the advantages discussed above.

At the same time, an important property for a suitable implementation relation is to support modular reasoning. Hence, the testing pre-order that we are looking for should be a *pre-congruence* with respect to all important combinators; i.e., it should be preserved when substituting related behaviours in a larger context. For instance, observation congruence and both the testing preorders mentioned above are (pre-)congruences.

The combination of testing, fairness and pre-congruence turns out to be surprisingly hard to achieve. A must-testing-like pre-order (which in this paper we call *acceptance testing*) that shares the fairness property of observation congruence is in fact not too difficult to find: it has been defined and studied under the name of *reduction* by Brinksma and Scollo in [12, 8] and the corresponding equivalence relation also by Vogler in [40] (see also Section 3.3 of [41]). Denotationally, acceptance

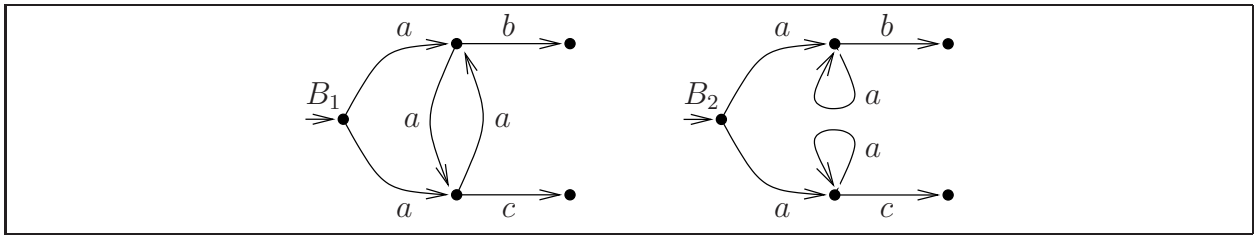


Figure 1.3: B_1 and B_2 are acceptance testing equivalent, but not after hiding a

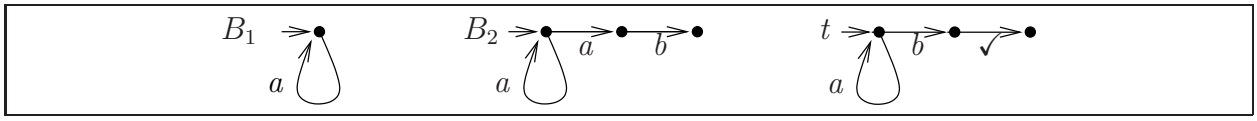


Figure 1.4: B_1 is an acceptance testing implementation of B_2, but not after hiding a

testing is precisely captured by the failures model. In contrast to must-testing, however, acceptance testing does not yield a pre-congruence with respect to *abstraction* (or *hiding*), a construction which internalises visible actions and may thereby introduce new divergences.[1] We give two examples illustrating this fact.

Figure 1.3 is taken from Bergstra et al. [2]; it shows two acceptance testing equivalent systems that differ when a is hidden. According to the acceptance testing scenario, the only observable fact is that after an arbitrary nonempty sequence of a's, either b is refused or c is refused; the difference between the two systems in Figure 1.3 is that the left-hand system alternates between allowing b and allowing c, whereas the right-hand side keeps on offering the same action after the initial choice. After hiding a this difference becomes testable, at least in the acceptance testing scenario: then the left-hand system accepts both the test for b and the test for c — meaning that in all states reachable from the initial state through internal moves, both b and c are still possible — whereas the right-hand system accepts neither, since after the first state either b or c has been disabled.

Another, even simpler, example is shown in Figure 1.4: the behaviour B_1 is an implementation of B_2 in the sense of acceptance testing (in this case the two are not equivalent). After hiding a on both sides, the implementation relation no longer holds: in the acceptance testing scenario, B_2 accepts the test for b — i.e., b is always possible — whereas B_1 will obviously refuse to perform b, meaning that it fails to accept that test. This shows that the removal of trace capabilities (taking away the a-b trace of B_2) interferes with pre-congruence with respect to hiding.

The problem of defining a suitable fair testing pre-congruence is in fact a long-standing one; in Section 8 we list some previous attempts at solving it. In the present article, we propose a solution by introducing the notion of *should*-testing, elaborating on and extending our preliminary papers [10, 11]. This notion of testing was also concurrently and independently developed by Natarajan and Cleaveland [31]; again, see Section 8 for a more detailed comparison of the respective contributions. A closely related approach based on liveness in Petri nets, giving an interesting characterization for a coarsest congruence, has been developed by Vogler in [41].

Should-testing is based on a modified definition of a successful test: process B passes test t iff every finite execution of $B \parallel t$ (the process subjected to the test, where "$- \parallel -$" denotes parallel execution of two processes, synchronized on all observable actions) has passed through or *can be extended* to pass through a successful state of t. For the behaviours in Figure 1.4, for instance, using this definition the tester process t (with 'success label' ✓) will distinguish between B_1 and B_2.

Our solution is canonical in the sense that it is the *coarsest* fair (i.e., divergence insensitive)

[1] It is actually, erroneously, claimed in [12] that reduction *is* also a pre-congruence with respect to abstraction.

testing pre-congruence, even if one chooses a notion of observability-through-testing that is much weaker than the one informally described above. In fact, no more is needed than the ability to observe whether a system under test is *alive*, in the sense of being able to do any observable action.

Structure of the paper

We proceed as follows. Section 2 sets the scene by defining our process language, its operational semantics and bisimulation, and discussing pre-congruences in general. In Section 3, we introduce the De Nicola-Hennessy testing framework in general and should-testing in particular. We discuss the fairness properties of should-testing and show that it gives rise to a pre-congruence for all our operators — but for the standard problem with choice, which is solved in the standard fashion (namely, by taking initial stability into account). This pre-congruence result is one main achievement of the present article. The pre-congruence proofs rely directly on the testing definition.

A theoretically interesting aspect is, that this pre-congruence is *not* the coarsest pre-congruence for hiding within the acceptance testing preorder, since that coarsest pre-congruence surprisingly fails to be a pre-congruence for parallel composition. As another surprise, our should-pre-congruence has to be refined further (by requiring related processes to be trace equivalent) to become a pre-congruence also for recursion. This full should-pre-congruence solves the problem of fair testing pre-orders; in particular, it is coarser than observation congruence in all cases, with the advantages discussed above.

As another main result, Section 4 gives a denotational characterization of the full should-pre-congruence, based on a generalisation of the concept of failure pairs (cf. [13]) as given in [41]. An important achievement are the suitable definitions for the operators of our language on the denotational model — which results in alternative pre-congruence proofs — and the proof of recursion pre-congruence, which is somewhat involved; a non-standard feature in the latter proof is the fact that there is not one unique least element in the denotational model space, since comparable processes must have the same language. For good measure, we have shown that the model space is optimal in that it contains "no junk" (every model describes the semantics of some system specifiable in our process language). Natarajan and Cleaveland [31] give another characterization of the should-preorder (without giving denotational constructions or discussing any pre-congruence properties).

In Section 5 we show decidability of the should-pre-congruence; an essential tool in this proof is the (far from simple) denotational, failure-type characterization of the pre-congruence. Further proof principles to demonstrate that the should-pre-congruence holds between two given behaviours are presented in Section 6; they include bisimulations and axioms inherited from observational congruence, compositional reasoning, specific axioms for should-precongruence and a special contraction lemma, which is proven on the basis of the denotational characterization. In Section 7 we show the application of should-pre-congruence and these proof principles to a number of examples, chosen so as to illustrate the practical advantages over standard observation equivalence and must- and acceptance testing. The examples include a scheduling problem, a version of the Alternating Bit-protocol, and fair communication channels. Finally, Section 8 discusses related work and presents our conclusions.

The "no junk" property of the model space, being quite technical and mainly of theoretical interest, is relegated to Appendix A. Furthermore, this report version includes proofs of all the main theorems; for readability, however, some have been deferred to Appendix B.

2 Basic process algebraic concepts

We start by defining our process algebraic language; then we present its operational semantics and other basic notins, and discuss in particular the notion of pre-congruence.

We first review some general notational issues.

- We use $A \rightharpoonup B$ to denote the space of *partial* functions from A to B; $f\colon A \rightharpoonup B$ means that f is such a partial function, in which case $dom(f)$ equals the subset of A on which f is defined. Note that if $f\colon A \rightharpoonup B$ then $f\colon dom(f) \to B$.
- We use $\vec{x}$ to denote a countable (i.e., finite or countably infinite) vector of elements. $|\vec{x}|$ ($\in \mathbb{N} \cup \{\omega\}$) denotes the length of x and for all $0 \leq i < |\vec{x}|$, x_i denotes the i'th element of x (hence $\vec{x} = x_0\, x_1\, \cdots$).

2.1 The language

We assume sufficiently large sets $\mathbf{A}$ of actions, ranged over by $a, b, \ldots$, and $\mathbf{X}$ of process names, ranged over by X, Y, Z.[2] There is also a special *invisible action* $\tau \notin \mathbf{A}$; we denote $\mathbf{A}_\tau = \mathbf{A} \cup \{\tau\}$, ranged over by $\alpha, \beta, \ldots$ Throughout the paper we consider the language $\mathbf{L}$, ranged over by $B, C, \ldots$ and generated by the following grammar:

$$\begin{array}{rcl} B & ::= & \alpha; B \mid \sum \textbf{set of } B \mid B \parallel_A B \mid B[\varphi] \mid B/A \mid X \mid \mathrm{rec}_X\, \theta \\ \theta & ::= & \textbf{set of } X := B \ . \end{array}$$

The "**set of**" construction indicates a countable set of expressions of the relevant kind. Hence terms are constructed from the following elements:

- A family of action prefix operators "$\alpha; -$" indexed by an action $\alpha \in \mathbf{A}_\tau$;
- A CCS-like infinitary summation operator "$\sum -$" whose parameter is a countable set of terms;
- A family of CSP-like parallel composition operators "$- \parallel_A -$" indexed by a set of synchronisation actions $A \subseteq \mathbf{A}$;
- A family of relabelling operators "$-[\varphi]$" indexed by a relabelling function $\varphi\colon \mathbf{A}_\tau \to \mathbf{A}_\tau$, which satisfies $\varphi(\alpha) = \tau$ if and only if $\alpha = \tau$. In actual terms, we usually only list the fragment of φ that is not the identity.
- A family of hiding operators "$-/A$" indexed by a set of actions $A \subseteq \mathbf{A}$ to be abstracted away from;
- A family of process variables "X" with $X \in \mathbf{X}$, which serve either as placeholders of subterms to be inserted later (by syntactic substitution, see below) or as process invocations. (These two uses are distinguished in [28] by using disjoint sets of so-called *agent variables* in the first, and *agent constants* in the second case.)
- Recursive terms $\mathrm{rec}_X\, \theta$, where $\theta\colon \mathbf{X} \rightharpoonup \mathbf{L}$ is a process environment (see below) and $X \in dom(\theta)$. θ can be regarded as a *vector* of equations $X := \theta(X)$, of which the recursion operator builds a fixpoint at each given coordinate X. (See also Milner [27] for another example of this operator.) A special case is $\theta = \{X := B\}$, in which case we sometimes simply write $\mathrm{rec}\, X.\, B$.

  We sometimes use $\mathrm{rec}\, \theta$ to denote the function $dom(\theta) \to \mathbf{L}$ that maps each $X \in dom(\theta)$ to the θ-fixpoint at X, $\mathrm{rec}_X\, \theta$.

[2] In the general case, both $\mathbf{A}$ and $\mathbf{X}$ should be uncountable; see Section 2.6.

A *process environment* θ is a countable set of definitions of the form $X := B_X$, with the additional constraint that all the X's are distinct; hence θ can be interpreted as a partial function $\mathbf{X} \rightharpoonup \mathbf{L}$, such that $\theta(X) = B_X$ for all $(X := B_X) \in \theta$. We sometimes write θ as $\{X := \theta(X)\}_{X \in dom(\theta)}$.

Since it takes a function as parameter, one can regard recursion as a higher-order operator. Accordingly, we sometimes refer to terms without recursion as the *first-order* fragment of $\mathbf{L}$. We denote a typical first-order operator by op, and its application to a vector of operands (of appropriate length) by $op(\vec{B})$.

Note that, in spite of the presence of two constructions with a countable number of operands (summation and recursion), the principle of induction over the structure of terms remains valid, due to the fact that syntax trees have finite depth (even if they have countable width).

An alternative presentation of recursion, which is more pleasant to use in practice but more awkward in theoretical developments, is to define the process environment θ separately and in the term under consideration just write down the variable X whose θ-defined behaviour is invoked. That is, in this alternative presentation, every term B has an implicit context, being the process environment θ that defines the behaviour of the process variables invoked in B; see the beginning of Section 7 for an example. We sometimes write B **with** θ to make the context explicit. In fact, we have the following correspondence (which relies on syntactic substitution, to be formally defined below):

$$B \textbf{ with } \theta \;=\; B[\mathrm{rec}_X\, \theta / X]_{X \in dom(\theta)}$$

We use special abbreviations for some forms of summation, synchronisation and process definition:

$$\begin{array}{rclcrcl} \mathbf{0} & = & \sum \emptyset & & B \,|||\, C & = & B \parallel_{\emptyset} C \\ B + C & = & \sum \{B, C\} & & B \parallel C & = & B \parallel_{\mathbf{A}} C \;. \end{array}$$

The reason why we allow infinitary summation and infinite process environments in $\mathbf{L}$ is to be able to capture arbitrary *tests* as terms of $\mathbf{L}$; cf. Section 4 below.

To save parenthesis, we will use the following implicit order of binding strength:

$$-[\varphi] \quad > \quad -/A \quad > \quad a;- \quad > \quad -\parallel_A - \quad > \quad \textstyle\sum - \quad > \quad \mathrm{rec}_X - \quad .$$

2.2 Sorts, free variables and substitution

We sometimes use the (syntactic) *sort* of a term, this being an approximation of the actions a system may perform during its lifetime. Since the sort is a syntactic notion, due to the Turing power of $\mathbf{L}$ it cannot precisely equal the set of actions actually performed; instead, it is an overestimate. Sorts are given by a function $\mathcal{S}\colon \mathbf{L} \to \mathcal{P}(\mathbf{A})$. The definition is inspired by Milner [28]. In particular, to define the sorts of process variables we rely on explicit sorting: for every variable $X \in \mathbf{X}$ we assume that the sort is a pre-defined countable set $\mathcal{S}_X \subseteq \mathbf{A}$. The sorting function is defined by the following set of rules:

$$\begin{array}{rclcrcl} \mathcal{S}(\tau; B) & = & \mathcal{S}(B) & & \mathcal{S}(B[\varphi]) & = & \varphi(\mathcal{S}(B)) \\ \mathcal{S}(a; B) & = & \{a\} \cup \mathcal{S}(B) & & \mathcal{S}(B/A) & = & \mathcal{S}(B) \setminus A \\ \mathcal{S}(\sum \mathbf{B}) & = & \bigcup_{B \in \mathbf{B}} \mathcal{S}(B) & & \mathcal{S}(X) & = & \mathcal{S}_X \\ \mathcal{S}(B_1 \parallel_A B_2) & = & \mathcal{S}(B_1) \cup \mathcal{S}(B_2) & & \mathcal{S}(\mathrm{rec}_X\, \theta) & = & \mathcal{S}_X \end{array}$$

For the sake of consistency, process environments have to respect the sorts of process variables. We call θ *well-sorted* if $\mathcal{S}(\theta(X)) \subseteq \mathcal{S}_X$ for all $X \in dom(\theta)$; in the remainder, we will always implicitly assume process environments to be well-sorted. Moreover, we will always assume that there are fresh actions not in the sorts of any of the terms under consideration; see Section 2.6.

The free variables of a term B, denoted $fv(B)$, are defined as usual; furthermore, $fv(\theta) = \bigcup_{X \in dom(\theta)} fv(\theta(X))$. To be precise:

$$\begin{array}{rcl} fv(op(\vec{B})) & = & \bigcup_{0 \leq i < |\vec{B}|} fv(B_i) \\ fv(X) & = & \{X\} \\ fv(\mathrm{rec}_X\, \theta) & = & fv(\theta) \setminus dom(\theta) \ . \end{array}$$

Note that $fv(B)$ is always countable. We call B *closed* if $fv(B) = \emptyset$ (and the same for θ). We use $\mathbf{L}^\bullet$ to denote the closed terms of $\mathbf{L}$. The free variables of a term B can be instantiated by *syntactic substitution*. This is denoted $B[\sigma]$, where $\sigma \colon \mathbf{X} \rightharpoonup \mathbf{L}$ is a (partial) *substitution function*. We also use $\theta[\sigma]$ to denote the instantiated process environment $\{X := \theta(X)[\sigma]\}_{X \in dom(\theta)}$. Substitution is defined as follows:

$$\begin{array}{rcll} op(\vec{B})[\sigma] & = & op(B_0[\sigma]\ B_1[\sigma]\ \cdots) & \\ X[\sigma] & = & \begin{cases} \sigma(X) & \text{if } X \in dom(\sigma) \\ X & \text{otherwise} \end{cases} & \\ (\mathrm{rec}_X\, \theta)[\sigma] & = & \mathrm{rec}_{\rho(X)}((\theta \circ \rho^{-1})[\rho \cup \sigma]) & \text{where } \rho \colon dom(\theta) \to \mathbf{Y}\ (\subseteq \mathbf{X}) \text{ is bijective} \\ & & & \text{with } \mathbf{Y} \cap (dom(\sigma) \cup fv(\sigma) \cup fv(\mathrm{rec}_X\, \theta)) = \emptyset. \end{array}$$

Note that for ρ (which we use for an α-conversion, to avoid the capture of free variables of σ) in the last equation to exist, there must be sufficiently many process variables not in $dom(\sigma) \cup fv(\sigma)$; see Section 2.6. Just as for process environments, we only consider well-sorted substitution functions, i.e., such that $\mathcal{S}(\sigma(X)) \subseteq \mathcal{S}_X$ for all $X \in dom(\sigma)$. We sometimes write $B[\sigma]$ as $B[\sigma(X)/X]_{X \in dom(\sigma)}$. (Note that process environments θ and substitution functions σ are in fact mathematically the same kind of objects. For instance, the definition of $fv(\theta)$ directly carries over to $fv(\sigma)$.)

2.3 Operational semantics

We express the behaviour of terms of $\mathbf{L}$ in terms of labelled transition systems, as usual. In general, an A-labelled transition system is a triple $\langle S, \rightarrow, q \rangle$ where S is a set of *states*, $\rightarrow \subseteq S \times A \times S$ a transition relation between states and $q \in S$ the initial state.

The operational semantics of terms of $\mathbf{L}$ is defined through structural operational rules; see Table 2.1. This gives rise to an $\mathbf{A}_\tau$-labelled transition system (without initial state) $\langle \mathbf{L}^\bullet, \rightarrow \rangle$ (where $\mathbf{L}^\bullet$ denotes the set of closed terms of $\mathbf{L}$, and $\rightarrow$ is generated by derivations from the rules in Table 2.1). The rules are entirely standard.

The consistency of the sort and the operational semantics is expressed by the following proposition, corresponding to the *subject reduction* property in typed functional languages:

Proposition 2.1 *If $B \xrightarrow{\alpha} B'$, then $\mathcal{S}(B) \supseteq \mathcal{S}(B')$ [$\cup \{\alpha\}$ if $\alpha \neq \tau$].*

We call a transition $B \xrightarrow{\alpha} B'$ closed if B is closed. An example of a non-closed transition is $a; X \xrightarrow{a} X$. For any transition, closed or not, it holds that free variables of the target term are already free in the source term, as expressed by the following proposition.

Proposition 2.2 *If $B \xrightarrow{\alpha} B'$, then $fv(B') \subseteq fv(B)$.*

If a transition is not closed, it can be further instantiated; this is expressed by the following proposition.

Proposition 2.3 *$B \xrightarrow{\alpha} B'$ implies $B[\sigma] \xrightarrow{\alpha} B'[\sigma]$ for arbitrary σ.*

Table 2.1: Structural operational semantics of **L**

$$\frac{}{\alpha; B \xrightarrow{\alpha} B} \qquad \frac{B \xrightarrow{\alpha} B' \quad B \in \mathbf{B}}{\sum \mathbf{B} \xrightarrow{\alpha} B'} \qquad \frac{B_1 \xrightarrow{\alpha} B_1' \quad \alpha \notin A}{B_1 \parallel_A B_2 \xrightarrow{\alpha} B_1' \parallel_A B_2}$$

$$\frac{B_2 \xrightarrow{\alpha} B_2' \quad \alpha \notin A}{B_1 \parallel_A B_2 \xrightarrow{\alpha} B_1 \parallel_A B_2'} \qquad \frac{B_1 \xrightarrow{\alpha} B_1' \quad B_2 \xrightarrow{\alpha} B_2' \quad \alpha \in A}{B_1 \parallel_A B_2 \xrightarrow{\alpha} B_1' \parallel_A B_2'}$$

$$\frac{B \xrightarrow{\alpha} B'}{B[\varphi] \xrightarrow{\varphi(\alpha)} B'[\varphi]} \qquad \frac{B \xrightarrow{\alpha} B' \quad \alpha \in A}{B/A \xrightarrow{\tau} B'/A} \qquad \frac{B \xrightarrow{\alpha} B' \quad \alpha \notin A}{B/A \xrightarrow{\alpha} B'/A}$$

$$\frac{\theta(X)[\mathrm{rec}_Y\, \theta/Y]_{Y \in dom(\theta)} \xrightarrow{\alpha} B'}{\mathrm{rec}_X\, \theta \xrightarrow{\alpha} B'}$$

The premise of the operational rule for recursion can be simplified if one assumes *guardedness*. A variable X is called guarded in a term B if X only occurs free in sub-terms of the form $\alpha; C$. More precisely, the definition of guardedness is as follows:

- X is guarded in $\alpha; B$;
- X is guarded in $\sum \mathbf{B}$ iff X is guarded in all $B \in \mathbf{B}$;
- X is guarded in $B \parallel_A C$ iff X is guarded in B and C;
- X is guarded in B/A and $B[\varphi]$ iff X is guarded in B;
- X is guarded in every variable $Y \neq X$, but not in X.
- X is guarded in $\mathrm{rec}_Y\, \theta$ if θ is guarded and X is guarded in $\theta(Y)$.

A process environment θ as a whole is called guarded if all $X \in dom(\theta)$ are guarded in all θ-images. The crucial property of guarded variables is that they are not regarded in the derivation of any (initial) transition of the term. The following proposition states the aforementioned simplification of the operational rule for recursion that is the essential property of guarded process environments.

Proposition 2.4 *If θ is guarded, then $\mathrm{rec}_X\, \theta \xrightarrow{\alpha} B'$ iff $\theta(X) \xrightarrow{\alpha} B''$ such that $B' = B''[\mathrm{rec}_Y\, \theta/Y]_{Y \in dom(\theta)}$.*

In the remainder of this paper, we restrict ourselves to guarded process environments.

2.4 Pre-congruence

For many purposes, the operational semantics is too fine-grained, distinguishing terms that are intuitively equivalent — such as $a; \mathbf{0}$ and $a; \mathbf{0} + a; (\mathbf{0} + \mathbf{0})$. For that reason, one usually considers an abstraction of the operational model. Such an abstraction can be obtained by defining a *semantic pre-order* over the model, which takes the form of a relation $\sqsubseteq \subseteq S \times S$ over the states of a given transition system; in particular, also over the (closed) terms in the transition system generated by Table 2.1. To be able to reason about the open terms as well, the relation $\sqsubseteq$ is extended in the usual way: two open terms are related iff all their closed instantiations are related ($B \sqsubseteq C$ iff $B[\sigma] \sqsubseteq C[\sigma]$

for all $\sigma: fv(B,C) \to \mathbf{L}$ with $fv(\sigma) = \emptyset$). Furthermore, $\sqsubseteq$ is extended pointwise to vectors of terms ($\vec{B} \sqsubseteq \vec{C}$ iff $|\vec{B}| = |\vec{C}|$ and $B_i \sqsubseteq C_i$ for all $0 \leq i < |\vec{B}|$) and also to process environments ($\theta \sqsubseteq \eta$ iff $dom(\theta) = dom(\eta)$ and $\theta(X) \sqsubseteq \eta(X)$ for all $X \in dom(\theta)$).

One common use for semantic pre-orders is to formalise a notion of *correctness* of a design or implementation, in such a way that a system correctly implements a given specification iff it is smaller with respect to the pre-order under consideration. A pre-order used for this purpose is also often called an *implementation relation*. The transitivity of an implementation relation is essential to ensure that two correct design steps, executed one after the other, still give rise to a correct implementation. If, moreover, one wants to design or verify a system in a *modular* fashion, by comparing sub-systems and putting them together, a further desirable property of the implementation relation is for it to be a *pre-congruence*. In fact, we distinguish several variants.

Definition 2.5 *Let $\sqsubseteq \subseteq \mathbf{L} \times \mathbf{L}$ be a pre-order.*

- $\sqsubseteq$ *is a* first-order pre-congruence *if for any n-ary operator op of $\mathbf{L}$ (with $n \in \mathbb{N} \cup \{\omega\}$), $\vec{B} \sqsubseteq \vec{C}$ with $|\vec{B}| = n$ implies $op(\vec{B}) \sqsubseteq op(\vec{C})$.*
- $\sqsubseteq$ *is a* recursion pre-congruence *if $\theta \sqsubseteq \eta$ implies $\mathrm{rec}_X\, \theta \sqsubseteq \mathrm{rec}_X\, \eta$ for all $X \in dom(\theta)$.*
- $\sqsubseteq$ *is a* full pre-congruence *if it is both a first-order and a recursion pre-congruence.*

Moreover, a [first-order/full/recursion] congruence *is a [first order/full/recursion] pre-congruence that is actually an equivalence relation.*

Another well-known way to characterise (pre-)congruences is through so-called *contexts*. A context is an incomplete term, containing so-called "holes"; the context can be instantiated by "filling" the holes, that is, by inserting terms into them. (The holes can also be regarded as a kind of higher-level variables; the only difference between instantiating a variable and filling a hole is that the latter may capture free variables.) For a semantic relation to be useful in the algebraic theory, one would like it to be preserved under substitution into arbitrary contexts. As we will now demonstrate, this is the case if and only if the relation is a (full) pre-congruence.

In this paper, contexts are terms Ctx, with free variables from a predefined countably infinite set $\{Z_0, Z_1, \ldots\}$ serving as holes, and no Z_i bound anywhere in Ctx. Holes can be (partially) filled by instantiating them with a vector of terms $\vec{B}$. Filling holes is different from ordinary syntactic substitution in that free variables in $\vec{B}$ may be captured in the instantiated context: we usually write $Ctx[-]$ for Ctx and $Ctx[\vec{B}]$ for $Ctx[B_i/Z_i]_{0 \leq i < |\vec{B}|}$. If Ctx contains only the hole Z_0, we usually denote it by $-$. For instance,

$$Ctx[-] \;=\; \mathrm{rec}_{X_1}\, \{X_1 := a; -,\, X_2 := - \,|||\, a; X_3,\, X_3 := b; \mathbf{0}\}$$

is a context with example instantiation

$$\begin{aligned} &Ctx[b; (X_2 + \mathrm{rec}\, Y.\, a; X_3)] \\ &\quad = \quad \mathrm{rec}_{X_1}\, \{X_1 := a; b; (X_2 + \mathrm{rec}\, Y.\, a; X_3),\, X_2 := b; (X_2 + \mathrm{rec}\, Y.\, a; X_3) \,|||\, a; X_2,\, X_3 := b; \mathbf{0}\} \;. \end{aligned}$$

Full pre-congruences can then be characterised as follows:

Proposition 2.6 *A pre-order $\sqsubseteq$ is a full pre-congruence iff for all contexts $Ctx[-]$, $\vec{B} \sqsubseteq \vec{C}$ with $|\vec{B}| = \omega$ implies $Ctx[\vec{B}] \sqsubseteq Ctx[\vec{C}]$.*

Proof: if. For an arbitrary operator op of $\mathbf{L}$, choose $Ctx = op(Z_0, Z_1, \ldots)$. It follows that $\sqsubseteq$ is a first-order pre-congruence. Now consider $\theta \sqsubseteq \eta$, and let $Y \in dom(\theta)$ be arbitrary. Let $\vec{X}$ represent an arbitrary ordering of the variables in $dom(\theta)$, and $\theta(\vec{X})$ (resp. $\eta(\vec{X})$) the vector of θ-images of $\vec{X}$. Now choose $Ctx = \mathrm{rec}_Y \{X_i := Z_i\}_{0 \leq i < |\vec{X}|}$. The property in the proposition then implies

$$\mathrm{rec}_Y\, \theta = Ctx[\theta(\vec{X})] \sqsubseteq Ctx[\eta(\vec{X})] = \mathrm{rec}_Y\, \eta \ .$$

Since Y is arbitrary, this implies $\sqsubseteq$ is a recursion pre-congruence.

Only if. Assume that $\sqsubseteq$ is a full pre-congruence. The proof obligation then follows by induction on the structure of Ctx. Assume $\vec{B} \sqsubseteq \vec{C}$ with $|\vec{B}| = \omega$:

- If $Ctx = op(\vec{D})$, then $D_i[\vec{B}] \sqsubseteq D_i[\vec{C}]$ for all $0 \leq i < |\vec{D}|$ by the induction hypothesis, implying $Ctx[\vec{B}] \sqsubseteq Ctx[\vec{C}]$ by the fact that $\sqsubseteq$ is a first-order pre-congruence;
- If $Ctx = X$ where $X \notin \{Z_0, Z_1, \ldots\}$, then $Ctx[\vec{B}] = X \sqsubseteq X = Ctx[\vec{C}]$;
- If $Ctx = Z_i$, then $Ctx[\vec{B}] = B_i \sqsubseteq C_i = Ctx[\vec{C}]$;
- If $Ctx = \mathrm{rec}_X\, \theta$, then by the induction hypothesis we have $\theta(Y)[\vec{B}] \sqsubseteq \theta(Y)[\vec{C}]$ for all $Y \in dom(\theta)$. From this we may conclude

  $$Ctx[\vec{B}] = \mathrm{rec}_X \{Y := \theta(Y)[\vec{B}]\}_{Y \in dom(\theta)} \sqsubseteq \mathrm{rec}_X \{Y := \theta(Y)[\vec{C}]\}_{Y \in dom(\theta)} = Ctx[\vec{C}]$$

  using the fact that $\sqsubseteq$ is a recursion pre-congruence. □

An example of the use of contexts is given in Ex. 3.13, where a particular context is used to show that a relation is *not* a pre-congruence.

2.5 Bisimulation

In this paper, we are concerned with relations that abstract from internal activity of the system, in the sense that finite sequences of τ-actions are simply ignored. This is achieved by using *weak* (or τ-*abstracting*) transitions, defined as follows: for all $s, s' \in S$,

$$s \xrightarrow{a_1 \cdots a_n} s' :\Leftrightarrow s \xrightarrow{\tau}{}^* \xrightarrow{a_1} \xrightarrow{\tau}{}^* \cdots \xrightarrow{\tau}{}^* \xrightarrow{a_n} \xrightarrow{\tau}{}^* s' \ .$$

In this situation, we call $a_1 \cdots a_n$ a *trace* of s. A particular, well-known class of implementation relations (in fact, equivalences) is based on the principle of *bisimulation* (see Milner [28], which boils down to a step-by-step matching of transitions of two systems under comparison. The notion of a "match" for a transition $\xrightarrow{\alpha}$ can be either strong ($\xrightarrow{\alpha}$) or weak ($\stackrel{\alpha}{\Longrightarrow}$); in the latter case, a match for $\xrightarrow{\tau}$ can be either at least one ($\stackrel{\tau}{\Longrightarrow}$) or arbitrarily many ($\stackrel{\varepsilon}{\Longrightarrow}$). Let $\hat{\cdot} : \mathbf{A}_\tau \to \mathbf{A}^*$ be a mapping defined by $\hat{a} = a$ for all $a \in \mathbf{A}$ and $\hat{\tau} = \varepsilon$.

Definition 2.7 (bisimilarity) *Let $\langle S, \to \rangle$ be a transition system.*

- Strong bisimilarity $\sim \subseteq S \times S$ *is the largest symmetrical relation such that for all* $s_1 \sim s_2$ *and* $s_1 \xrightarrow{\alpha} s_1'$ *there is a* $s_2' \sim s_1'$ *such that* $s_2 \xrightarrow{\alpha} s_2'$.
- Weak bisimilarity $\approx \subseteq S \times S$ *is the largest symmetrical relation such that for all* $s_1 \approx s_2$ *and* $s_1 \xrightarrow{\alpha} s_1'$ *there is a* $s_2' \approx s_1'$ *such that* $s_2 \stackrel{\hat{\alpha}}{\Longrightarrow} s_2'$.
- Rooted bisimilarity $\simeq_{\mathbf{bis}} \subseteq S \times S$ *is the largest symmetrical relation such that for all* $s_1 \simeq_{\mathbf{bis}} s_2$ *and* $s_1 \xrightarrow{\alpha} s_1'$ *there is a* $s_2' \approx s_1'$ *such that* $s_2 \stackrel{\alpha}{\Longrightarrow} s_2'$.

The following result is standard; see Milner [28] for the corresponding property of CCS, and [4, 3] for meta-results applying to the language $\mathbf{L}$ considered here.

Proposition 2.8 *$\sim$ and $\simeq_{\mathbf{bis}}$ are full congruences.*

Strong bisimulation is often regarded as the most distinguishing reasonable equivalence relation over transition systems; in fact, all relations considered in this paper are weaker. Since it also happens to have a relatively pleasant proof technique, we use it as a "touchstone" for the equivalence of behaviours: equalities that hold up to strong bisimilarity are certain to be valid in all semantics under consideration. For instance, with respect to process definitions, the following equivalence is easily shown to hold:

Proposition 2.9 $\mathrm{rec}_X\,\theta \sim \theta(X)[\mathrm{rec}_Y\,\theta/Y]_{Y\in dom(\theta)}$.

The following is a technical result giving sufficient conditions to conclude that a relation is a pre-congruence with respect to parallel composition.

Lemma 2.10 *If $\sqsubseteq$ is a pre-congruence for renaming such that $\sim \subseteq \sqsubseteq$, and $B_i \sqsubseteq C_i$ $(i = 1, 2)$ implies $B_1 \parallel_A B_2 \sqsubseteq C_1 \parallel_A C_2$ for all $A \supseteq (\mathcal{S}(B_1) \cap \mathcal{S}(B_2)) \cup (\mathcal{S}(C_1) \cap \mathcal{S}(C_2))$, then $\sqsubseteq$ is a pre-congruence for parallel composition (with arbitrary synchronisation sets).*

Proof. The proof strategy is to construct, for arbitrary terms $B_1 \parallel_A B_2$ and $C_1 \parallel_A C_2$, renaming functions $\varphi_1, \varphi_2, \psi$ and a set of actions A' such that

$$\begin{aligned} B_1 \parallel_A B_2 &\sim (B_1[\varphi_1] \parallel_{A'} B_2[\varphi_2])[\psi] \\ C_1 \parallel_A C_2 &\sim (C_1[\varphi_1] \parallel_{A'} C_2[\varphi_2])[\psi] \end{aligned}$$

and both $A' \supseteq \mathcal{S}(B_1[\varphi_1]) \cap \mathcal{S}(B_2[\varphi_2])$ and $A' \supseteq \mathcal{S}(C_1[\varphi_1]) \cap \mathcal{S}(C_2[\varphi_2])$. It is not difficult to see that this implies the proof obligation.

The role of the φ_i is to rename the actions $(\mathcal{S}(B_i) \cup \mathcal{S}(C_i)) \setminus A$ for $i = 1, 2$ to disjoint sets of fresh actions. For $i = 1, 2$ let $\varphi_i \colon (\mathcal{S}(B_i) \cup \mathcal{S}(C_i) \cup A) \to A_i$ be bijective such that $\varphi_1 \restriction A = \varphi_2 \restriction A$ and $A_1 \cap A_2 = A'$, where $A' = \varphi_1(A)$. (Note that $\mathbf{A}$ is assumed to be large enough so that such A_i can always be found.) Moreover, let $\psi = \varphi_1^{-1} \cup \varphi_2^{-1}$ (hence ψ is left inverse to both φ_1 and φ_2). It is straightforward to show that these satisfy the requirements; for instance, the following relation $R \subseteq \mathbf{L} \times \mathbf{L}$ is a bisimulation:

$$R = \{(\, B_1' \parallel_A B_2', (B_1'[\varphi_1] \parallel_{A'} B_2'[\varphi_2])[\psi]\,) \mid B_i' \in \mathbf{L}, \mathcal{S}(B_i') \subseteq \mathcal{S}(B_i) \text{ for } i = 1, 2\} \ .$$

□

2.6 Cardinality assumptions

The following remarks are intended to ensure the logical consistency of our assumptions and do not concern the main point of this article. In the present section, we have made the following statements regarding the cardinality of various sets and constructions:

- The sets $\mathbf{A}$ and $\mathbf{X}$ are uncountable;
- The sum operator $\sum$ has a countable number of operands;
- Process environments $\theta \colon \mathbf{X} \rightharpoonup \mathbf{L}$ have countable domains;
- Each process variables $X \in \mathbf{X}$ has a countable sort $\mathcal{S}_X$.

As a consequence, for all terms $B \in \mathbf{L}$, $\mathcal{S}(B)$ and $fv(B)$ are countable subsets of $\mathbf{A}$ and $\mathbf{X}$, respectively. Therefore, we can always assume the existence of fresh actions (not in the sort of a given term) and of a countable set of fresh process variables disjoint from any given countable subset of $\mathbf{X}$. This, in fact, is the motivation for choosing the cardinalities in this way. We have already seen one case of the "enough process variables" assumption in the definition of syntactic substitution.

Occasionally we will construct terms $\mathrm{rec}_X\,\theta$ where θ actually has an uncountable domain. However, if we define $\leq_\theta \subseteq \mathbf{X} \times \mathbf{X}$ as the smallest transitive and reflexive relation such that $Y \leq_\theta Z$ whenever $Y \in fv(\theta(Z))$, then $\downarrow_\theta Y = \{Z \in \mathbf{X} \mid Z \leq_\theta Y\}$ is clearly countable and $\theta \restriction \downarrow_\theta X$ intuitively contains all process definitions that can possibly affect the behaviour of X. In fact, if we momentarily treat $\mathrm{rec}_X\,\theta$ as if it were an allowed term we obtain $\mathrm{rec}_X\,\theta \sim \mathrm{rec}_X(\theta \restriction \downarrow_\theta X)$. For that reason, we will implicitly assume $\mathrm{rec}_X\,\theta$ to stand for $\mathrm{rec}_X(\theta \restriction \downarrow_\theta X)$.

3 Testing

The subject of this paper is the study of certain *testing pre-orders*. This entails a setup wherein systems are investigated by synchronising them with *tests*, which are systems containing a special *success action* $\checkmark \notin \mathbf{A}_\tau$. A state with an outgoing $\checkmark$-transition is called successful. We denote $A_\checkmark = A \cup \{\checkmark\}$ for arbitrary $A \subseteq \mathbf{A}_\tau$, and we reuse α to range over $\mathbf{A}_{\tau,\checkmark}$. Tests, then, are closed terms t with $\mathcal{S}(t) \subseteq \mathbf{A}_\checkmark$; their operational semantics is again determined by Table 2.1. We denote the set of tests by $\mathbf{L}_\checkmark$, ranged over by t. (Note that we will still use $\parallel$ to abbreviate $\parallel_{\mathbf{A}}$, and *not* $\parallel_{\mathbf{A}_\checkmark}$.)

Applying the test t to the (closed) term B is done by synchronising the two on all actions (except for τ and $\checkmark$), resulting in the term $B \parallel t$. Whether or not a test application as a whole is deemed successful depends on the presence of sufficiently many successful states in the behaviour of $B \parallel t$, where the notion of "sufficiently many" is itself a parameter of the testing framework, which may be called the *test modality*. The landmark paper of De Nicola and Hennessy [17] essentially defines two such modalities: *may*- and *must*-testing, which we recall here. To formulate the latter, we define a *maximal run* through a transition system to be a sequence of states $(s_i)_{i<n}$ for some $n \in \mathbb{N}^{>0} \cup \{\omega\}$ such that $s_{i-1} \xrightarrow{\alpha_i} s_i$ for all $0 < i < n$, and if $n \neq \omega$ then $\nexists \alpha\colon s_{n-1} \xrightarrow{\alpha}$.

$$\begin{aligned} B \textbf{ may } t &\;:\Leftrightarrow\; \exists w \in \mathbf{A}^*\colon (B \parallel t) \xRightarrow{w\checkmark} \\ B \textbf{ mst } t &\;:\Leftrightarrow\; \forall \text{ maximal runs } (B_i)_{i<n}\colon (B \parallel t = B_0 \text{ implies } \exists i < n\colon B_i \xrightarrow{\checkmark}) \ . \end{aligned}$$

In words, B **may** t if and only if *there is a* path from $B \parallel t$ to a successful state, whereas B **mst** t if and only if *any* path from $B \parallel t$ eventually passes through a successful state.

Any such test modality $\mathbf{mod} \subseteq \mathbf{L}^\bullet \times \mathbf{L}_\checkmark$ gives rise to an implementation relation expressing that the left-hand system (the proposed implementation) passes all the tests that the right-hand system (the given specification) passes, in the sense of the chosen test modality **mod**. (If a term does not pass a test, we say that it *fails* it.) That is, for $I, S \in \mathbf{L}^\bullet$ we define:

$$I \sqsubseteq_{\mathbf{mod}} S \;:\Leftrightarrow\; \forall t \in \mathbf{L}_\checkmark\colon S \textbf{ mod } t \Longrightarrow I \textbf{ mod } t \qquad (\mathbf{mod} = \mathbf{may}, \mathbf{mst}).$$

The resulting relations $\sqsubseteq_{\mathbf{may}}$ and $\sqsubseteq_{\mathbf{mst}}$ are studied extensively in [17]. $\sqsubseteq_{\mathbf{may}}$ is shown to be a full pre-congruence (coinciding with *inverse language inclusion*, see the next section); $\sqsubseteq_{\mathbf{mst}}$ is not a pre-congruence with respect to choice, but becomes a full pre-congruence if we strengthen it using a *stability test*, which is a unary predicate defined by:

$$B \textbf{ stb} \;:\Leftrightarrow\; B \not\xrightarrow{\tau} \ .$$

Just like test modalities, unary predicates $\mathbf{prd} \subseteq \mathbf{L}^\bullet$ naturally give rise to an ordering over $\mathbf{L}^\bullet$:

$$I \sqsubseteq_{\mathbf{prd}} S \;:\Leftrightarrow\; S \textbf{ prd} \Longrightarrow I \textbf{ prd} \ .$$

For instance, $I \sqsubseteq_{\mathbf{stb}} S$ holds if either S is stable or I is not. The known (pre-)congruence properties of may- and must-testing, then, come down to the following:

Theorem 3.1 (De Nicola and Hennessy [17]) $\sqsubseteq_{\mathbf{may}}$ *and* $\sqsubseteq_{\mathbf{mst}} \cap \sqsubseteq_{\mathbf{stb}}$ *are full pre-congruences.*

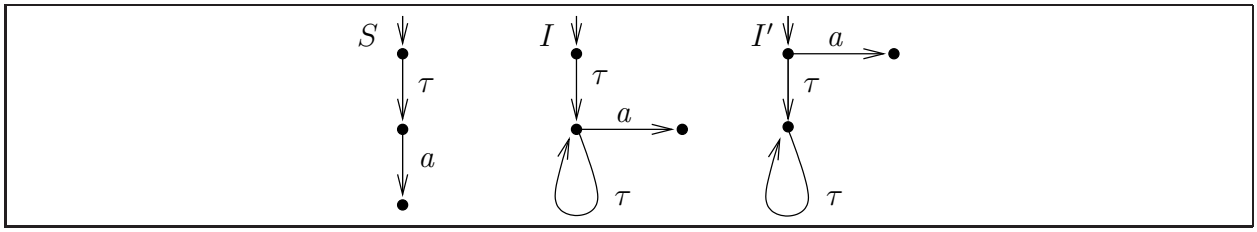


Figure 3.1: $I \not\sqsubseteq_{\mathbf{mst}} S$ and $I' \sqsubseteq_{\mathbf{mst}} I$ (Ex. 3.2)

3.1 Fairness and congruence issues

Let us recall the fairness aspects of **mst**-testing. A must-test application, B **mst** t, immediately fails if either B or t *diverges*, i.e., is able to do an infinite sequence of τ-moves that never reaches a successful state. In particular, a simple τ-loop generates a divergence. The idea is that the system may indeed choose to follow this path forever, in which case success is never attained.

For some purposes, this interpretation is too strict. For instance, the implementation technique of *busy-waiting*, when formulated in terms of a transition system, gives rise to a τ-loop; so does the assumption made in many protocols that messages may be retransmitted upon loss. In either of theses cases, there is a *fairness* assumption implicitly associated with the τ-loop, which can intuitively be understood as guaranteeing that the divergence will never actually arise; i.e., that the path which stays in the loop forever is not a possible behaviour of the system. Because of the way it deals with divergence, the **mst**-testing modality is not compatible with such an implicit fairness assumption.

Example 3.2 *Consider the systems* $S = \tau; a; \mathbf{0}$ *and* $I = \operatorname{rec} X.\, \tau; (X + a; \mathbf{0})$ *(see Figure 3.1). The visible behaviour in both cases consists of the ability to do* a *after some internal moves; however, on the face of it,* I *may elect to do internal moves forever and never execute* a*. Indeed, we have* $I \not\sqsubseteq_{\mathbf{mst}} S$ *due to* S **mst** $a; \checkmark; \mathbf{0}$ *and* $\neg(I$ **mst** $a; \checkmark; \mathbf{0})$*. A fair execution of* I*, on the other hand, should not always choose* τ*, but should eventually choose to execute* a*; hence* S *and* I *are equivalent if only fair runs are considered.*

Now consider $I' = (\operatorname{rec} X.\, \tau; X) + a; \mathbf{0}$ *(see Figure 3.1), which can choose with its first* τ *never to do* a*. Hence, if only fair runs are considered,* I' *should not implement* I*; yet* $I' \sqsubseteq_{\mathbf{mst}} I$*.*

Inspired by the above example, this implicit notion of fairness can be captured by the following *liveness* predicate:

$$B \textbf{ live} :\Leftrightarrow \forall B \stackrel{\varepsilon}{\Longrightarrow} B' \colon \exists \alpha \neq \tau : B' \stackrel{\alpha}{\Longrightarrow} \quad .$$

Hence B is called live if, after any amount of internal moves, there is always some visible behaviour left. We will call an implementation relation *liveness-preserving* if it is at least as strong as $\sqsubseteq_{\mathbf{live}}$. I and I' in Ex. 3.2 show that $\sqsubseteq_{\mathbf{mst}}$ is *not* liveness-preserving, since I is live while I' is not; neither is $\sqsubseteq_{\mathbf{may}}$ liveness-preserving, witness $\tau; \mathbf{0} + a; \mathbf{0} \sqsubseteq_{\mathbf{may}} a; \mathbf{0}$.

A liveness-preserving test modality, here called *acceptance testing*, was developed in [12, 40] (and called *reduction* in the former). It is defined as follows:

$$B \textbf{ acc } t :\Leftrightarrow \forall w \in \mathbf{A}^*, \forall B' \colon (B \parallel t) \stackrel{w}{\Longrightarrow} B' \text{ implies } \exists \alpha \in \mathbf{A}_{\checkmark} \colon B' \stackrel{\alpha}{\Longrightarrow} \quad .$$

Hence, B **acc** t does not automatically fail at a divergence; rather, it fails only if a divergent path cannot be exited by a visible action. This reflects the implicit assumption that as long as there is a visible transition reachable from the current state, that (or some other) visible transition will eventually be performed. The resulting implementation relation, $\sqsubseteq_{\mathbf{acc}}$, is indeed liveness-preserving, since B **live** iff B **acc** $\sum\{a; \checkmark; \mathbf{0} \mid a \in \mathcal{S}(B)\}$. For instance, in Ex. 3.2 we find that $I' \not\sqsubseteq_{\mathbf{acc}} I$, because I **acc** $a; \checkmark; \mathbf{0}$ whereas $\neg(I'$ **acc** $a; \checkmark; \mathbf{0})$.

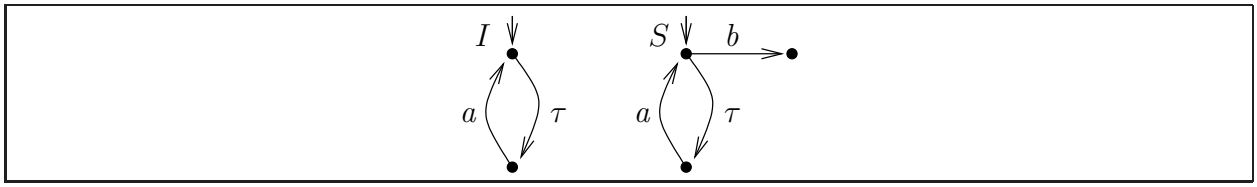


Figure 3.2: $I \sqsubseteq_{\mathbf{acc}} S$ but $I/a \not\sqsubseteq_{\mathbf{acc}} S/a$ (Ex. 3.3)

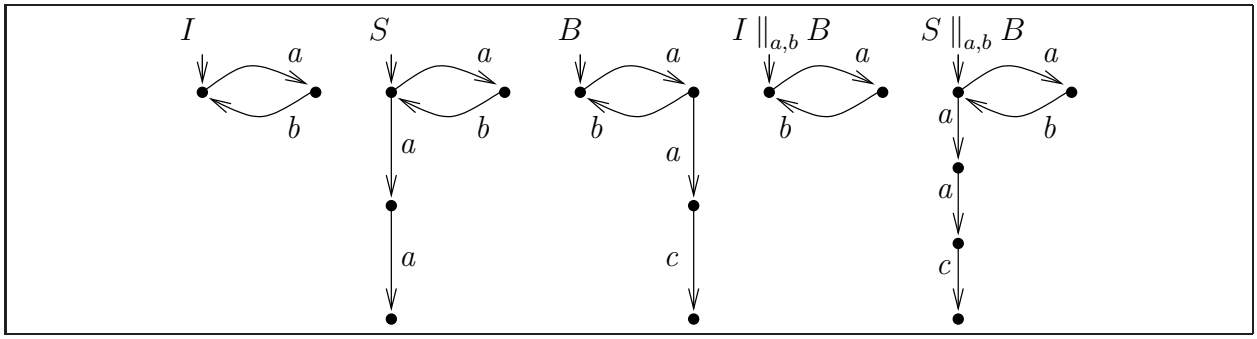


Figure 3.3: $I/A \sqsubseteq_{\mathbf{acc}} S/A$ for all $A \subseteq \mathbf{A}$, but $(I \parallel_A B)/a, b \not\sqsubseteq_{\mathbf{acc}} (S \parallel_A B)/a, b$ (Ex. 3.5)

Unfortunately, a new problem occurs that severely limits the usefulness of $\sqsubseteq_{\mathbf{acc}}$: it is not a pre-congruence for hiding. The following example, demonstrating this, is due to Rom Langerak:

Example 3.3 *Consider the systems S and I depicted in Figure 3.2. I satisfies all tests of S under* **acc***; hence $I \sqsubseteq_{\mathbf{acc}} S$. Yet after hiding a we get that S/a $\mathbf{acc}\ b; \checkmark; \mathbf{0}$ whereas $\neg(I/a\ \mathbf{acc}\ b; \checkmark; \mathbf{0})$; hence $I/a \not\sqsubseteq_{\mathbf{acc}} S/a$.*

We recall the following result.

Theorem 3.4 (Brinksma [12], Vogler [40, 41]) $\sqsubseteq_{\mathbf{acc}} \cap \sqsubseteq_{\mathbf{stb}}$ *is a liveness-preserving first-order pre-congruence for all operators of* $\mathbf{L}$ *except hiding.*

The natural technique to repair the deficit for hiding would be to formulate the coarsest pre-congruence within $\sqsubseteq_{\mathbf{acc}}$ with respect to hiding. Surprisingly, in this case this is not sufficient, since the relation thus obtained fails to be a pre-congruence with respect to parallel composition.

Example 3.5 *Consider systems S and I in Figure 3.3. We have $I/A \sqsubseteq_{\mathbf{acc}} S/A$ for arbitrary A. Yet after synchronising over $A = \{a, b\}$ with $B = \operatorname{rec} X.\ a; (b; X + a; c; \mathbf{0})$ we find that the resulting systems are* not $\sqsubseteq_{\mathbf{acc}}$*-related after arbitrary hiding, as $(S \parallel_A B)/a, b$ $\mathbf{acc}\ c; \checkmark\mathbf{0}$ whereas $\neg((I \parallel_A B)/a, b\ \mathbf{acc}\ c; \checkmark\mathbf{0})$.*

3.2 Should-testing

It follows that in constructing the coarsest pre-congruence inside acceptance testing, one has to take hiding and parallel composition into account at the same time. We now present a new test modality that does this in the correct way.

$$B\ \mathbf{shd}\ t :\Leftrightarrow \forall w \in \mathbf{A}^*, \forall B' \colon (B \parallel t) \stackrel{w}{\Longrightarrow} B' \text{ implies } \exists v \in \mathbf{A}^*.\ B' \stackrel{v\checkmark}{\Longrightarrow} \ .$$

The difference with **acc** is that there, every reachable 'pre-success' state of the system under test is merely required to be either live or successful, whereas for **shd**, every reachable state is required to be on a path to success.

Since **shd** is a stronger requirement than **acc**, it follows that the resulting pre-order $\sqsubseteq_{\mathbf{shd}}$ is also liveness-preserving. In fact, we have the following inclusions:

Proposition 3.6 *The following inclusions hold. All inclusions are strict; no inclusion exists where none is shown.*

$$\begin{array}{ccccccccc} \sim & \subset & \sqsubseteq_{\mathbf{mst}} & & & & & & \\ \subset & & & & & & & & \\ \simeq_{\mathbf{bis}} & \subset & \sqsubseteq_{\mathbf{shd}} & \subset & \sqsubseteq_{\mathbf{acc}} & \subset & \sqsubseteq_{\mathbf{may}}^{-1} & & \\ & & & & & & \subset & & \\ & & & & & & \sqsubseteq_{\mathbf{live}} & & \end{array}$$

Furthermore, for closed first-order terms, some of the inclusions collapse to equalities:

$$\sqsubseteq_{\mathbf{shd}} \;=\; \sqsubseteq_{\mathbf{acc}} \;=\; \sqsubseteq_{\mathbf{mst}}$$

Proof. We only prove the inclusions that are not standard or obvious.

$\sqsubseteq_{\mathbf{shd}} \subseteq \sqsubseteq_{\mathbf{acc}}$. Assume $I \sqsubseteq_{\mathbf{shd}} S$ and let $t \in \mathbf{L}_{\checkmark}$ be such that S **acc** t. If $\neg(I$ **acc** $t)$ then let $w \in \mathbf{A}^*$ be such that $I \parallel t \stackrel{w}{\Longrightarrow} I'$ and $I' \stackrel{\alpha}{\not\Longrightarrow}$ for all $\alpha \in \mathbf{A}_{\checkmark}$. It follows that $I' = I'' \parallel t'$ for some I'' and t'. Assume $w = a_1 \cdots a_n$ and let

$$\begin{array}{rcl} t_i & = & \checkmark; \mathbf{0} + a_i; t_{i+1} \quad \text{for } 0 \leq i < n \\ t_n & = & \sum\{a; \checkmark; \mathbf{0} \mid t' \stackrel{a}{\Longrightarrow}\} \ . \end{array}$$

By construction it follows that, on the one hand, $\neg(I$ **shd** $t_0)$ and, on the other, S **shd** t_0 and hence I **shd** t_0. By contradiction we may conclude I **acc** t.

$\sqsubseteq_{\mathbf{shd}} \supseteq \sqsubseteq_{\mathbf{acc}}$ for first-order terms. Assume $I \sqsubseteq_{\mathbf{acc}} S$ and let $t \in \mathbf{L}_{\checkmark}$ be such that S **shd** t. It follows that S **acc** t and hence I **acc** t. Now let $w \in \mathbf{A}^*$ be such that $I \parallel t \stackrel{w}{\Longrightarrow} I'$. It follows that $I' = I'' \parallel t'$ for some I'' and t'. Due to the fact that I is a first-order term, there are $\hat{I}$ and $\hat{t}$ such that $I'' \parallel t' \stackrel{v}{\Longrightarrow} \hat{I} \parallel \hat{t}$ for some $v \in \mathbf{A}^*$ and $\neg(\hat{I} \parallel \hat{t}) \stackrel{a}{\Longrightarrow}$ for all $a \in \mathbf{A}^*$. Due to I **acc** t it follows that $\hat{I} \parallel \hat{t} \stackrel{\checkmark}{\Longrightarrow}$, implying $I' \stackrel{v\checkmark}{\Longrightarrow}$. We may conclude I **shd** t.

□

Moreover, as we will show, $\sqsubseteq_{\mathbf{shd}}$ is a pre-congruence w.r.t. the entire first-order fragment of $\mathbf{L}$ except for choice. For the choice operator, the same problem occurs that we already encountered with must-testing: to obtain a pre-congruence, initial stability has to be preserved as well. For that reason, we define

$$\sqsubseteq^{+}_{\mathbf{shd}} \quad := \quad \sqsubseteq_{\mathbf{shd}} \cap \sqsubseteq_{\mathbf{stb}}$$

We now set out to show that $\sqsubseteq^{+}_{\mathbf{shd}}$ is indeed a first-order pre-congruence. Before giving the actual proofs, we first need some auxiliary notation.

- For any $A \subseteq \mathbf{A}_{\checkmark}$, we use $\bar{A} = \mathbf{A}_{\checkmark} \setminus A$ to denote the complement of A with respect to $\mathbf{A}_{\checkmark}$.
- Any partial function $\varphi \colon \mathbf{A}_{\checkmark} \rightharpoonup \mathbf{A}_{\checkmark}$ gives rise to a (total) string homomorphism $\hat{\varphi} \colon \mathbf{A}^*_{\checkmark} \to \mathbf{A}^*_{\checkmark}$ with $\varphi(a) = \varepsilon$ whenever $\varphi(a)$ is undefined originally; this in turn gives rise, through pointwise extension, to a (total) function $\varphi \colon \mathcal{P}(\mathbf{A}^*_{\checkmark}) \to \mathcal{P}(\mathbf{A}^*_{\checkmark})$.
- A special class of such partial functions φ are the *projections* $\pi_A \colon \mathbf{A}_{\checkmark} \rightharpoonup \mathbf{A}_{\checkmark}$ with $A \subseteq \mathbf{A}$, which are such that $\pi_A(a) = a$ if $a \in A$, and $\pi(a)$ is undefined otherwise. Hence, π_A applied to a string $w \in \mathbf{A}^*_{\checkmark}$ has the effect of dropping all non-A-actions from w.

We now present a series of lemmas stating the pre-congruence properties of $\sqsubseteq_{\mathbf{shd}}$ and $\sqsubseteq^{+}_{\mathbf{shd}}$ with respect to the first-order operators of $\mathbf{L}$.

Lemma 3.7 *$\sqsubseteq_{\mathbf{shd}}$ and $\sqsubseteq^{+}_{\mathbf{shd}}$ are pre-congruences for prefixing.*

Proof. We show that $\sqsubseteq_{\mathbf{shd}}$ is a pre-congruence for prefixing; since $\alpha; I \sqsubseteq_{\mathbf{stb}} \alpha; S$ holds irregardless of I and S, the result for $\sqsubseteq^{+}_{\mathbf{shd}}$ follows immediately.

For this purpose, we show that, for all $\alpha \in \mathbf{A}_\tau$,

$$B \ \mathbf{shd}\ t \quad \text{iff} \quad \alpha; B \ \mathbf{shd}\ \alpha; t \ .$$

This suffices to conclude the lemma.

if. Assume $B \parallel t \stackrel{w}{\Longrightarrow} B'$ for some $w \in \mathbf{A}^*$. It follows that $\alpha; B \parallel \alpha; t \stackrel{\alpha\, w}{\Longrightarrow} B'$ if $\alpha \neq \tau$ or $\alpha; B \parallel \alpha; t \stackrel{w}{\Longrightarrow} B'$ otherwise, implying in either case (due to $\alpha; B \ \mathbf{shd}\ \alpha; t$) that $B' \stackrel{v\checkmark}{\Longrightarrow}$ for some $v \in \mathbf{A}^*$.

only if. Assume $\alpha; B \parallel \alpha; t \stackrel{w}{\Longrightarrow} B'$ for some $w \in \mathbf{A}^*$. If $B' = \alpha; B \parallel \alpha; t$ then (since $B \parallel t \stackrel{\varepsilon}{\Longrightarrow} B \parallel t$, and hence $\exists v \in \mathbf{A}^* : B \parallel t \stackrel{v\checkmark}{\Longrightarrow}$ due to $B \ \mathbf{shd}\ t$) $B' \stackrel{\alpha\, v\checkmark}{\Longrightarrow}$ if $\alpha \neq \tau$ or $B' \stackrel{v\checkmark}{\Longrightarrow}$ otherwise.

Otherwise $B \| t \stackrel{w'}{\Longrightarrow} B'$ where $w = \alpha\, w'$ if $\alpha \neq \tau$ or $w = w'$ otherwise; hence $\exists v \in \mathbf{A}^* : B' \stackrel{v\checkmark}{\Longrightarrow}$ due to $B \ \mathbf{shd}\ t$.

□

Lemma 3.8 *$\sqsubseteq^{+}_{\mathbf{shd}}$ is a pre-congruence for choice.*

The proof for this case is deferred to Appendix B (Page 65).

Lemma 3.9 *$\sqsubseteq_{\mathbf{shd}}$ and $\sqsubseteq^{+}_{\mathbf{shd}}$ are pre-congruences for renaming.*

The proof for this case is deferred to Appendix B (Page 66).

Lemma 3.10 *$\sqsubseteq_{\mathbf{shd}}$ and $\sqsubseteq^{+}_{\mathbf{shd}}$ are pre-congruences for synchronisation.*

Proof. We show that $\sqsubseteq_{\mathbf{shd}}$ is a pre-congruence for synchronisation; since $\sqsubseteq_{\mathbf{stb}}$ is also clearly a pre-congruence, the result for $\sqsubseteq^{+}_{\mathbf{shd}}$ follows immediately.

Assume $I \sqsubseteq_{\mathbf{shd}} S$; to be proved is $I \parallel_A B \sqsubseteq_{\mathbf{shd}} S \parallel_A B$ for arbitrary $B \in \mathbf{L}$ and $A \subseteq \mathbf{A}$. Since $\sqsubseteq_{\mathbf{shd}} \supseteq \sim$ (Proposition 3.6) and $\sqsubseteq_{\mathbf{shd}}$ is a pre-congruence for renaming (Lemma 3.9), due to Lemma 2.10 we only have to regard the case where $A \supseteq (\mathcal{S}(I) \cup \mathcal{S}(S)) \cap \mathcal{S}(B)$.

For this case, it can be shown that for arbitrary C with $\mathcal{S}(C) \cap \mathcal{S}(B) \subseteq A$:

$$(C \parallel_A B) \ \mathbf{shd}\ t \quad \text{iff} \quad C \ \mathbf{shd}\ ((B \parallel_{\mathcal{S}(B) \cup A} t)/(\mathcal{S}(B) \setminus A)) \ .$$

which suffices to conclude the desired property.

In order to prove the above characterisation, note that

$$(C \parallel_A B) \parallel t \sim C \parallel_{\overline{\mathcal{S}(B)} \cup A} (B \parallel_{\mathcal{S}(B) \cup A} t) \ .$$

For the purpose of should-testing, the right hand term has the same "failure capabilities" as $C \parallel ((B \parallel_{\mathcal{S}(B) \cup A} t)/(\mathcal{S}(B) \setminus A)$, in the sense that one term satisfies

$$\exists B' : - \stackrel{w}{\Longrightarrow} B' \wedge \neg \exists v \in \mathbf{A}^* : B' \stackrel{v\checkmark}{\Longrightarrow}$$

iff the other does. □

Lemma 3.11 *$\sqsubseteq_{\mathbf{shd}}$ and $\sqsubseteq^{+}_{\mathbf{shd}}$ are pre-congruences for hiding.*

Proof. For all $A \subseteq \mathbf{A}$, let $R_A \in \mathbf{X}$ be a process variable with $\mathcal{S}(R_A) = A$, and for all $t \in \mathbf{L}_{\surd}$, let t_A be a derived test defined by

$$t_A = (t \parallel \mathrm{rec}\, R_{\bar{A}}.\, \textstyle\sum\{a; R_{\bar{A}} \mid a \in \bar{A}\}) \parallel\!\parallel\!\parallel \mathrm{rec}\, R_A.\, \textstyle\sum\{a; R_A \mid a \in A\} \ .$$

Note that for all $B, t \in \mathbf{L}$ and $A \subseteq \mathbf{A}$, then

$$(B/A)\ \mathbf{shd}\ t \quad \text{iff} \quad B\ \mathbf{shd}\ t_A \ .$$

(This follows by the fact that for all such B and t the following holds for all $w \in \mathbf{A}^{*}_{\surd}$:

$$B/A \parallel t \stackrel{w}{\Longrightarrow} B'/A \parallel t' \quad \text{iff} \quad \exists v \in \mathbf{A}^{*}_{\surd} : \pi_{\bar{A}}(v) = w \wedge B \parallel t_A \stackrel{v}{\Longrightarrow} B' \parallel t'_A$$

Using this fact, any failure of B/A w.r.t. t can be converted to a failure of B w.r.t. t_A, and vice versa.)

Now assume $I \sqsubseteq_{\mathbf{shd}} S$, and let $A \subseteq \mathbf{A}$ be arbitrary. If $S/A\ \mathbf{shd}\ t$ for some arbitrary t then $S\ \mathbf{shd}\ t_A$, hence $I\ \mathbf{shd}\ t_A$, hence $I/A\ \mathbf{shd}\ t$. We may conclude $I/A \sqsubseteq_{\mathbf{shd}} S/A$; hence $\sqsubseteq_{\mathbf{shd}}$ is a pre-congruence.

Finally, if $I \sqsubseteq^{+}_{\mathbf{shd}} S$ and $S/A\ \mathbf{stb}$ for some $A \subseteq \mathbf{A}$, then $S\ \mathbf{stb}$ and $S\ \mathbf{shd}\ \surd; \mathbf{0} + a; \mathbf{0}$ for all $a \in A$. It follows that $I\ \mathbf{stb}$ and $I\ \mathbf{shd}\ \surd; \mathbf{0} + a; \mathbf{0}$ for all $a \in A$, and hence $I/A\ \mathbf{stb}$. We may conclude $I/A \sqsubseteq_{\mathbf{stb}} S/A$; hence $\sqsubseteq^{+}_{\mathbf{shd}}$ is also a pre-congruence. □

The following is one of the main results of this paper.

Theorem 3.12 *The following properties hold for $\sqsubseteq_{\mathbf{shd}}$ and $\sqsubseteq^{+}_{\mathbf{shd}}$, regarded as relations over $\mathbf{L}$:*

- *$\sqsubseteq_{\mathbf{shd}}$ is the coarsest liveness-preserving pre-congruence for all operators except $+$;*
- *$\sqsubseteq^{+}_{\mathbf{shd}}$ is the coarsest liveness-preserving first-order pre-congruence.*

Proof.

- The fact that $\sqsubseteq_{\mathbf{shd}}$ and $\sqsubseteq^{+}_{\mathbf{shd}}$ are liveness-preserving is due to the following observation: If $I \sqsubseteq_{\mathbf{shd}} S$ and $S\ \mathbf{live}$, then $S\ \mathbf{shd}\ t$ for $t = \sum\{a; \surd; \mathbf{0} \mid a \in \mathcal{S}(I) \cup \mathcal{S}(S)\}$; hence (due to $I \sqsubseteq_{\mathbf{shd}} S$) $I\ \mathbf{shd}\ t$, implying $I\ \mathbf{live}$.

- The fact that $\sqsubseteq_{\mathbf{shd}}$ and $\sqsubseteq^{+}_{\mathbf{shd}}$ are pre-congruences for $\mathbf{L}$ follows from Lemmas 3.7–3.11.

- The fact that $\sqsubseteq_{\mathbf{shd}}$ is the *coarsest* liveness-preserving pre-congruence can be proved as follows. Assume $I \not\sqsubseteq_{\mathbf{shd}} S$, and let t be such that $S\ \mathbf{shd}\ t$ and $\neg(I\ \mathbf{shd}\ t)$. Let $a \notin \mathcal{S}(S) \cup \mathcal{S}(I) \cup \mathcal{S}(t)$ (i.e., a is fresh w.r.t. S, I and t; such an action has been assumed to exist always) and $A = \mathbf{A} \setminus \{a\}$. Now let B_t be a term obtained from t by syntactically replacing all occurrences of $\surd$ by a and all occurrences of $\parallel$ by $\parallel_A$. It can be proved by induction on the structure of t that $B_t \sim t[a/\surd]$ would hold if $a/\surd$ were a valid relabelling function. Now let

  $$Ctx[-] = (Z_0 \parallel_A Z_1)/A \ .$$

  It follows that $Ctx[C, B_t]\ \mathbf{live}$ iff $C\ \mathbf{shd}\ t$ whenever $a \notin \mathcal{S}(C)$; hence $Ctx[S, B_t]\ \mathbf{live}$ but $\neg Ctx[I, B_t]\ \mathbf{live}$.

- For the analogous claim about $\sqsubseteq^+_{\mathbf{shd}}$, again assume $I \not\sqsubseteq^+_{\mathbf{shd}} S$. It now suffices to consider the case that $I \not\sqsubseteq_{\mathbf{stb}} S$, i.e., S **stb** and $\neg I$ **stb**. Let $a \in \mathbf{A} \setminus (\mathcal{S}(S) \cup \mathcal{S}(I))$ and $Ctx[-] = (- + a; \mathbf{0}) \parallel_{\mathcal{S}(S)\cup\mathcal{S}(I)} \mathbf{0}$. It follows that $Ctx[S]$ **live** whereas $\neg Ctx[I]$ **live**. □

It came as a surprise to the authors that the coarsest liveness-preserving first-order pre-congruence, $\sqsubseteq^+_{\mathbf{shd}}$, is not a full pre-congruence.[3] The original observation of this fact is due to Rom Langerak.

Example 3.13 *Take $B = \mathbf{0}$ and $C = a; \mathbf{0} + \tau; \mathbf{0}$. It is clear that $B \sqsubseteq_{\mathbf{stb}} C$ and $B \sqsubseteq_{\mathbf{acc}} C$; since both systems are finite, it follows (due to Proposition 3.6) that $B \sqsubseteq^+_{\mathbf{shd}} C$. Now consider the context*

$$Ctx[-] = \mathrm{rec}\, X.\, ((\tau; X + a; b; \mathbf{0}) \parallel_a -)/a \ .$$

*It can be seen that $Ctx[C] \sim \mathrm{rec}\, X.\tau; (X \,|||\, \tau; \mathbf{0}) + \tau; b; \mathbf{0} + \tau; \tau; X$ and hence $Ctx[C]$ **shd** $b; \checkmark$ whereas $Ctx[B] \sim \mathrm{rec}\, X.\, \tau; X$ and hence $\neg(Ctx[B]$ **shd** $b; \checkmark)$. It follows that $Ctx[B] \not\sqsubseteq_{\mathbf{shd}} Ctx[C]$.*

A closer investigation shows that $Ctx[-]$ in the above example "codes for" the trace a in the term to be plugged in: $Ctx[C]$ loops around internally, restarting C just as long as C has not done a. Using this principle, we can build such "coding context" for any trace, and therefore the following holds.

Proposition 3.14 *If $Ctx[I] \sqsubseteq_{\mathbf{shd}} Ctx[S]$ for all contexts Ctx, then $I \sqsubseteq_{\mathbf{may}} S$.*

Proof. Let $A = \mathcal{S}(I) \cup \mathcal{S}(S)$. Let t be arbitrary such that S **may** t. It follows that $S \parallel t \stackrel{w\checkmark}{\Longrightarrow}$ where (due to Proposition 2.1) $w = a_1 \cdots a_n \in A^*$ for some n. Now let $b \in \mathbf{A} \setminus A$ be arbitrary (by assumption, such a b always exists) and define

$$Ctx[-] = \mathrm{rec}\, X.\, ((\tau; X + a_1; \ldots; a_n; b; \mathbf{0}) \parallel_A -)/A \ .$$

It can be seen that for any B with $\mathcal{S}(B) \subseteq A$, $Ctx[B]$ **shd** $b; \checkmark; \mathbf{0}$ iff $B \stackrel{w}{\Longrightarrow}$. Due to $Ctx[I] \sqsubseteq_{\mathbf{shd}} Ctx[S]$, therefore, $I \stackrel{w}{\Longrightarrow}$; hence I **may** t. □

It follows that in order to obtain a full pre-congruence within $\sqsubseteq_{\mathbf{shd}}$, it is necessary to test for $\sqsubseteq_{\mathbf{may}}$ as well. As it turns out, this is in fact also *sufficient*, at least if we restrict the language to finite summation. Let

$$\sqsubseteq^c_{\mathbf{shd}} \ := \ \sqsubseteq_{\mathbf{shd}} \cap \sqsubseteq_{\mathbf{stb}} \cap \sqsubseteq_{\mathbf{may}} \ ;$$

we then have the following result, which is in fact the crucial theoretical contribution of this paper.

Theorem 3.15 *$\sqsubseteq^c_{\mathbf{shd}}$ is the coarsest liveness-preserving full pre-congruence over $\mathbf{L}$ restricted to finite summation.*

Note that it already follows from Theorems 3.1 and 3.12 that $\sqsubseteq^c_{\mathbf{shd}}$ is a first-order pre-congruence. The proof of the recursion pre-congruence property relies on the denotational characterisation of $\sqsubseteq_{\mathbf{shd}}$, presented in Section 4 below (see Theorem 4.25); the technical reasons for the restriction to finite summation are also explained there.

[3] We were not yet aware of this while writing [10], and although we did not make an explicit claim of this kind, that paper does reflect our impression at the time that $\sqsubseteq^+_{\mathbf{shd}}$ is a full pre-congruence.

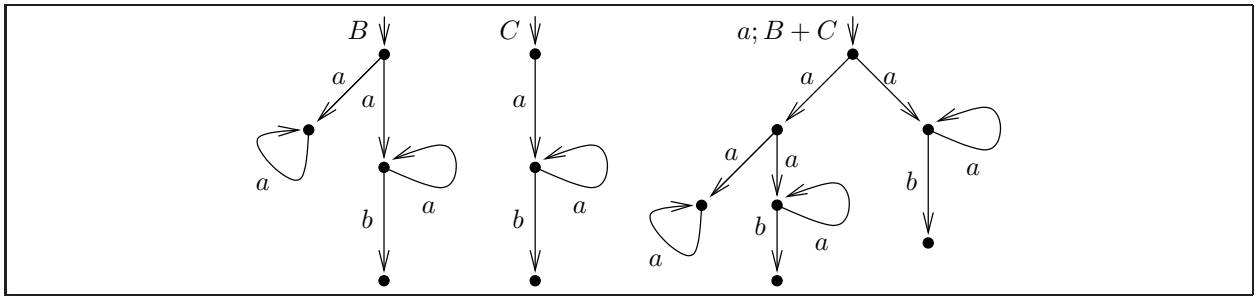


Figure 3.4: KFAR is violated: $B \simeq^c_{\mathbf{shd}} a;B + C$ but $B/a \not\simeq^c_{\mathbf{shd}} \tau;C/a$

3.3 Fairness properties of should-testing

We call $\sqsubseteq^c_{\mathbf{shd}}$ 'fair testing', but so far we have not presented any results that justify this usage of the word 'fair'. We will now address this issue. First we show that $\sqsubseteq^c_{\mathbf{shd}}$ satisfies a weakened version of what is known as *Kooman's Fair Abstraction Rule* (KFAR); see e.g. Baeten and Weijland [1].

KFAR algebraically captures the interpretation of divergences found in weak and rooted bisimilarity (see Definition 2.7). Essentially, since the only observations taken into account are visible transitions, τ-loops in a system are ignored by $\simeq_{\mathbf{bis}}$. A general form of the rule is:

$$\frac{B_i = a_i;B_{i+1} + C_i \quad a_i \in A}{B_i/A = \tau;\sum_{i\in\mathbb{N}_n}(C_i/A)}\ (i \in \mathbb{N}_n) \tag{1}$$

where $\mathbb{N}_n$ denotes the natural numbers modulo n, and $B_i, C_i \in \mathbf{L}$ are arbitrary terms for $i \in \mathbb{N}_n$. It is a standard result that $\simeq_{\mathbf{bis}}$ satisfies (1).

Unfortunately, $\simeq^c_{\mathbf{shd}}$ does *not* satisfy (1), for reasons discussed below. However, it does satisfy a weaker variant, which we call KFAR$^-$:

$$\frac{X_i := a_i;X_{i+1} + C_i \quad a_i \in A}{X_i/A = \tau;\sum_{i\in\mathbb{N}_n}(C_i/A)}\ (i \in \mathbb{N}_n) \tag{2}$$

The difference with (1) is in the premise: where KFAR requires that certain equations hold (under the equivalence relation being studied), KFAR$^-$ assumes that the variables X_i are *defined* according to those equations. Clearly, if a certain relation satisfies (1) then it certainly satisfies (2); for instance, this is the case with $\simeq_{\mathbf{bis}}$.

Corollary 3.16 $\sqsubseteq^c_{\mathbf{shd}}$ *satisfies (2).*

Proof. Immediate from the fact that $\simeq_{\mathbf{bis}} \subseteq \sqsubseteq^c_{\mathbf{shd}}$ and $\simeq_{\mathbf{bis}}$ satisfies (2). □

As to the question why $\sqsubseteq^c_{\mathbf{shd}}$ fails to satisfy KFAR: Figure 3.4 shows a counterexample with $n = 1$. In this example, $B \simeq^c_{\mathbf{shd}} a;B+C$ (as can be checked formally using the denotational characterisation presented in Section 4), but $B/a \not\simeq^c_{\mathbf{shd}} \tau;C/a$ since $\tau;C/a\ \mathbf{shd}\ b;\checkmark;\mathbf{0}$ but $\neg(B/a\ \mathbf{shd}\ b;\checkmark;\mathbf{0})$.

An analysis of why $\sqsubseteq^c_{\mathbf{shd}}$ fails to satisfy the reasonable looking KFAR shows that the problem actually lies in the failure of the so-called *recursive specification principle*, which states that recusive equations have unique solutions. Indeed, it is *not* the case that, e.g., the equation $X = a;X + C$ (where C is as in Figure 3.4) has a unique solution modulo $\simeq^c_{\mathbf{shd}}$; viz. both B from Figure 3.4 and $\mathrm{rec}\,X.\ a;X + C$ are solutions. However, if $t = \mathrm{rec}\,Y.\ a;Y + b;\checkmark;\mathbf{0}$ then $\mathrm{rec}\,X.\ a;X + C\ \mathbf{shd}\ t$ whereas $\neg(B\ \mathbf{shd}\ t)$; see Figure 3.5.

The built-in fairness assumption of should-testing can also be expressed in another, more classical way. The *strong fairness assumption* states that, if a state is encountered infinitely often, then

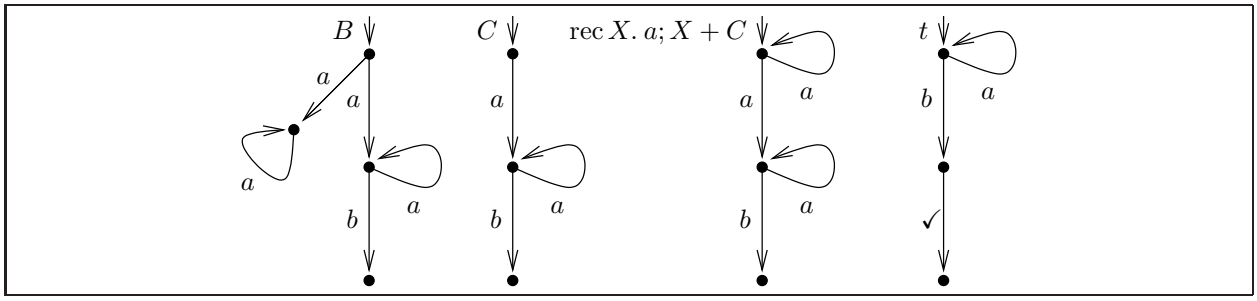


Figure 3.5: RSP is violated: $B \simeq^c_{\mathbf{shd}} a;B + C$ but $(\mathrm{rec}\, X.\, a; X + C)$ **shd** t whereas $\neg(B$ **shd** $t)$

all its outgoing transitions will eventually be taken. To make this precise we define for $B_0 \in \mathbf{L}$: $B_0 \xrightarrow{\alpha_0} B_1 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_{n-1}} B_n \ldots$ is a *fair run* of B_0 if it is maximal and contains infinitely often each transition $B \xrightarrow{\alpha} B'$ for which B occurs infinitely often. Moreover, we call a process B *finite state* if there are only finitely many reachable B' (i.e., with $\exists w \in \mathbf{A}^*.\, B \stackrel{w}{\Longrightarrow} B'$).

Lemma 3.17 *Let $B \in \mathbf{L}$ be a finite state process. If for every B' reachable from B there is some $v \in \mathbf{A}^*$ with $B' \stackrel{v\checkmark}{\Longrightarrow}$ then every fair run of B contains a $\checkmark$-transition.*

The proof is by considering the minimal disctance from a state visited infinitely often in a fair run to a state with an outgoing $\checkmark$-transition; it is omitted here. The condition of the lemma is obviously connected to the **shd**-relation. The following makes the connection explicit.

Corollary 3.18 *Let $B \in \mathbf{L}$ and $t \in \mathbf{L}_\checkmark$ be finite state processes. B* **shd** *t if and only if every fair run of $B \parallel t$ contains a $\checkmark$-transition.*

4 Denotational characterisation

The testing relations of the previous section can also be characterised denotationally. With the exception of **mst**-testing, we recall these (standard) characterisations here. We then show that should-testing can be captured by an extension of the model for acceptance testing. We demonstrate that this really gives rise to a denotational model for should-testing, including alternative proofs for the pre-congruence results in the previous section; these proofs are for some of the operators considerably easier than the operational ones.

We use the following notations for the *prefix closure* and *suffix closure* of a set of strings W, as well as the concatenation of V and W and the *remainder* of W after a word v; $\sqsubseteq$ denotes the prefix relation.

$$\begin{aligned} \downarrow W &:= \{w \in \mathbf{A}^* \mid \exists v \in W : w \sqsubseteq v\} \\ \uparrow W &:= \{w \in \mathbf{A}^* \mid \exists v \in W : v \sqsubseteq w\} \\ V\,W &:= \{v\,w \mid v \in V, w \in W\} \\ v^{-1}W &:= \{w \in \mathbf{A}^* \mid v\,w \in W\}\ . \end{aligned}$$

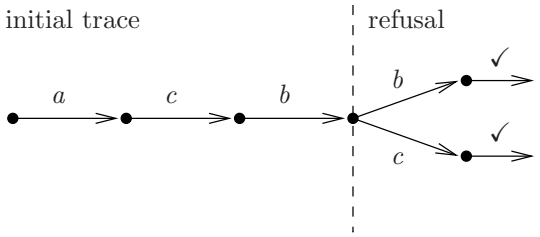
4.1 Language and failures

We now define the *language* $\mathcal{L}$ and the *failures* $\mathcal{F}$ of a transition system $T = \langle S, \rightarrow, q\rangle$.

$$\begin{aligned} \mathcal{L}(T) &:= \{w \in \mathbf{A}^* \mid \exists (q \stackrel{w}{\Longrightarrow} s)\} \\ \mathcal{F}(T) &:= \{(w, A) \in \mathbf{A}^* \times \mathcal{P}(\mathbf{A}) \mid \exists (q \stackrel{w}{\Longrightarrow} s) : \mathcal{L}(s) \cap A = \emptyset\} \end{aligned}$$

(Note that the definition of $\mathcal{F}(T)$ uses $\mathcal{L}(s)$ with $s \in S$; this is an abbreviation for $\mathcal{L}(\langle S, \rightarrow, s\rangle)$, i.e., the language of T with the initial state changed to s.) In words, the meaning of a failure $(w, A) \in \mathcal{F}(T)$ is that the initial trace w can lead to a state from which none of the actions in A can be done, or in other words, the entire A can be *refused*. Intuitively, A can be understood as describing a partial deadlock. Note that A need not be finite. The language can be derived from the failures: $\mathcal{L}(T) = \{w \mid (w, \emptyset) \in \mathcal{F}(T)\}$ for all T.

Example 4.1 *Assume* $\mathbf{A} = \{a, b, c\}$. *The terms* $B = a; b; \mathbf{0} \,|||\, c; a; \mathbf{0}$ *and* $C = a; c; (b; \mathbf{0} + b; b; \mathbf{0})$ *share the failure* $(a\,c\,b, \{b, c\})$ *(among many others). One may depict this failure by the following "broomstick" picture:*



Based on the language and failures, we define the following orderings over $\mathbf{L}$:

$$\begin{aligned} I \sqsubseteq_{\mathcal{L}} S &:\Leftrightarrow \mathcal{L}(I) \subseteq \mathcal{L}(S) \\ I \sqsubseteq_{\mathcal{F}} S &:\Leftrightarrow \mathcal{F}(I) \subseteq \mathcal{F}(S) \end{aligned}$$

We also use $\simeq_{\mathcal{L}}$ to denote the equivalence generated by $\sqsubseteq_{\mathcal{L}}$, i.e., $\simeq_{\mathcal{L}} = \sqsubseteq_{\mathcal{L}} \cap \sqsubseteq_{\mathcal{L}}^{-1}$. We now recall, without proof, the characterisations of may- and acceptance testing in terms of language and failures, respectively.

$$
\begin{array}{rcl}
\mathcal{F}(a;B) & = & \{(\varepsilon, A) \mid (\varepsilon, a^{-1}A) \in \mathcal{F}(B)\} \cup \{(a\,w, A) \mid (w, A) \in \mathcal{F}(B)\} \\
\mathcal{F}(\tau;B) & = & \mathcal{F}(B) \\
\mathcal{F}(\sum \mathbf{B}) & = & \{(w, A) \mid \forall B \in \mathbf{B}\colon (w, A) \in \mathcal{F}(B)\} \\
 & & \cup\, \{(w, A) \in \mathcal{F}(B) \mid B \in \mathbf{B}, w \in \mathbf{A}^+ \vee \neg B\ \mathbf{stb}\} \\
\mathcal{F}(B[\varphi]) & = & \{(\varphi(w), A) \mid (w, \varphi^{-1}(A)) \in \mathcal{F}(B)\} \\
\mathcal{F}(B \parallel_A C) & = & \{(w, A_B \cup A_C) \mid (\pi_{\overline{\mathcal{S}(C)\setminus A}}(w), \pi_{\overline{\mathcal{S}(C)\setminus A}}(A_B)) \in \mathcal{F}(B), \\
 & & \qquad (\pi_{\overline{\mathcal{S}(B)\setminus A}}(w), \pi_{\overline{\mathcal{S}(B)\setminus A}}(A_C)) \in \mathcal{F}(C)\} \quad \text{if } \mathcal{S}(B) \cap \mathcal{S}(C) \subseteq A
\end{array}
$$

Table 4.1: Compositional construction of standard failure sets

Proposition 4.2 (cf. [12, 17, 40]) $\sqsubseteq_{\mathbf{may}} = \sqsubseteq_{\mathcal{L}}^{-1}$ *and* $\sqsubseteq_{\mathbf{acc}} = \sqsubseteq_{\mathcal{F}}$.

Combined with Theorems 3.1 and 3.4, this has the following consequence:

Corollary 4.3

1. $\sqsubseteq_{\mathcal{L}}$ *is a full pre-congruence and* $\simeq_{\mathcal{L}}$ *a full congruence;*
2. $\sqsubseteq_{\mathcal{F}} \cap \sqsubseteq_{\mathbf{stb}}$ *is a pre-congruence for all operators except hiding.*

The second clause is equivalent to saying that all first-order **L**-operators, with the exception of hiding, give rise to $\subseteq$- and $\Longleftarrow$-monotonic constructions on the failure sets and stability predicate. In Table 4.1, we have recalled the construction of the failure sets; monotonicity follows from the fact that all of these are pointwise extensions of constructions on individual failure pairs.

Some remarks about the equations in Table 4.1 are in order.

- The equation for "$a;-$" specifies the union of two sets, the first of which has the condition $\mathcal{L}(B) \cap a^{-1}A = \emptyset$. Since $A \subseteq \mathbf{A}$, either $a^{-1}A = \emptyset$ (if $a \notin A$) or $a^{-1}A = \{\varepsilon\}$ (if $a \in A$); hence this condition is equivalent to the simpler $a \notin A$. However, in the form given here the construction extends more smoothly to tree failures (see Table 4.2 below).

- The equation for "$- \parallel_A -$" is defined only for the case where A includes all actions that the operands have in common. As we have seen in Lemma 2.10, with the help of well-chosen relabellings this suffices to cover the entire family of synchronisation operators. The failures of the synchronised term are constructed from the failures of the operands.

  The condition $(\pi_{\overline{\mathcal{S}(C)\setminus A}}(w), \pi_{\overline{\mathcal{S}(C)\setminus A}}(A_B)) \in \mathcal{F}(B)$ (and also the symmetrical condition on C) are also formulated so as to extend smoothly to tree failures (Table 4.2). The condition is equivalent to $(\pi_{\overline{\mathcal{S}(C)\setminus A}}(w), A_B) \in \mathcal{F}(B) \wedge A_B \cap \mathcal{S}(C) \subseteq A$. This reflects the fact that in a synchronised term $B \parallel_A C$, B can only decide about the refusal of actions that C cannot do independently; i.e., actions that are in the synchronisation set or outside C's alphabet.

  To see this equivalence, observe for the reverse implication that $A_B \cap \mathcal{S}(C) \subseteq A$ implies $\pi_{\overline{\mathcal{S}(C)\setminus A}}(A_B) = A_B$. On the other hand, if $A_B \cap \mathcal{S}(C) \subseteq A$ fails, then $\varepsilon \in \pi_{\overline{\mathcal{S}(B)\setminus A}}(A_B)$, but ε is never in a refusal set.

Proposition 4.4 *All equations in Table 4.1 are sound.*

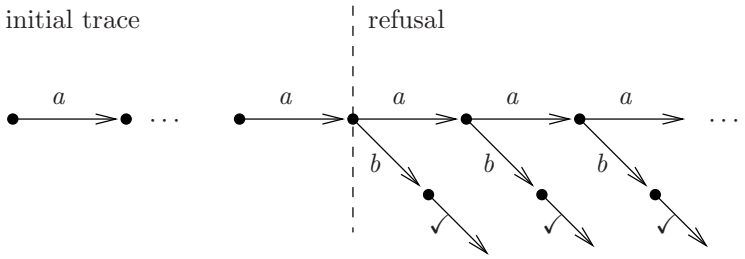
4.2 Tree failures

Fairness can only be captured if we introduce information about infinite behaviour into our model; for it is only "in the infinity" that the notion of fairness exists at all. If we take ordinary failures as our starting point, one way to add information about infinite behaviour, due to Vogler [41], is to extend the refusal information in a failure from sets of actions $A \subseteq \mathbf{A}$, as above, to sets of *words* $V \subseteq \mathbf{A}^+$. This results in the following definition:

$$\mathcal{F}^+(T) = \{(w, V) \in \mathcal{L}(T) \times \mathcal{P}(\mathbf{A}^+) \mid \exists q \stackrel{w}{\Longrightarrow} s \colon \mathcal{L}(s) \cap V = \emptyset\} \ .$$

Since the set V of an extended failure $(w, V) \in \mathcal{F}^+(T)$ can be interpreted as a *tree* with nodes $\downarrow V$ and success nodes V (corresponding to the nodes with outgoing $\surd$-transitions in Ex. 4.5), we also call the elements of $\mathcal{F}^+$ *tree failures*. Due to the fact that the elements of V can be of unbounded length, tree failures provide information about infinite behaviour, or in other words, about certain liveness properties of systems.

Example 4.5 *Consider* $B = \operatorname{rec} X.\, a; X$ *and* $C = \sum_i \tau; a^i; b$ *(where* $a^i; B$ *is recursively defined by* $a^0; B = B$ *and* $a^{i+1}; B = a; a^i; B$*). It follows that* $\mathcal{F}(B) = \{(a^i, A) \mid i \in \mathbb{N}, a \notin A\} \subseteq \mathcal{F}(C)$*; hence* $B \sqsubseteq_{\mathbf{acc}} C$*. Thus,* B *is considered a valid implementation for* C *up to* **acc** *despite the fact that* C *will always eventually be able to execute* b *(which is a liveness property). This difference does show up in the extended failure* $(a^n, a^*b) \in \mathcal{F}^+(B) \setminus \mathcal{F}^+(C)$ *(for arbitrary* $n \in \mathbb{N}$*). One may depict this extended failure as follows:*



In principle, it looks as if the additional information provided by tree failures is precisely what we need to determine fairness. Unfortunately, it turns out that on the whole, tree failures provide too much distinguishing power to be a faithful model for should-testing.

Example 4.6 $(a, \{b\,c\})$ *is a tree failure of* $B = a; b; c; \mathbf{0} + a; b; d; \mathbf{0}$ *but not of* $C = a; (b; c; \mathbf{0} + b; d; \mathbf{0})$*; hence* B *and* C *can be distinguished using* $\mathcal{F}^+$*-pairs, whereas they are equivalent under* $\simeq_{\mathbf{acc}}$ *and hence (being finite) also under* $\simeq_{\mathbf{shd}}$ *(see Proposition 3.6).*

As this example shows, tree failures give too much information about the moment of choice: $(a, \{b\,c\})$ tells us that already after the action a, the trace $b\,c$ can be refused. This information surplus can be removed by *closing up* or *saturating* under an ordering over tree failures. We define

$$(v, V) \preceq (w, W) \ \ :\Leftrightarrow \ \ \exists u \in \{\varepsilon\} \cup \downarrow V \colon w = v\,u, u^{-1}V = W \ .$$

For instance, in Ex. 4.6, $(a, \{b\,c\}) \preceq (a\,b, \{c\}) \in \mathcal{F}^+(C)$. Note that if (w, W) is a tree failure then $u \notin V$, since otherwise $\varepsilon \in W$ which is ruled out by the definition of $\mathcal{F}^+$; hence u is in fact chosen from the *proper* prefixes of V. The additional choice of $u = \varepsilon$ is only relevant if $V = \emptyset$, since in all other cases $\varepsilon \in \downarrow V$. Without this addition, $\preceq$ would not be reflexive for tree failures with an empty refusal set, i.e., of the form $(v, \emptyset)$. Just as with standard failures, tree failures with empty refusal sets are very useful, for instance to capture the *language* of a term.

Closing up the tree failures under $\preceq$ means that B is considered to be smaller than C if $\downarrow_{\preceq}\mathcal{F}^+(B) \subseteq \downarrow_{\preceq}\mathcal{F}^+(C)$ (where $\downarrow_{\preceq}$ builds the $\preceq$-downward-closure of its argument) rather than $\mathcal{F}^+(B) \subseteq \mathcal{F}^+(C)$; or equivalently (for arbitrary transition systems T, T'):

$$T \sqsubseteq_{\mathcal{F}^+} T' \;:\Leftrightarrow\; \forall\,(v,V) \in \mathcal{F}^+(T)\colon \exists\,(w,W) \in \mathcal{F}^+(T')\colon (v,V) \preceq (w,W)\ .$$

(Note that, in contrast to $\sqsubseteq_{\mathcal{L}}$ and $\sqsubseteq_{\mathcal{F}}$, $\sqsubseteq_{\mathcal{F}^+}$ is *not* defined as the direct inclusion of $\mathcal{F}^+$-sets.) Combining the definitions of $\preceq$ and $\sqsubseteq_{\mathcal{F}^+}$, we obtain the following simplified characterisation:

$$T \sqsubseteq_{\mathcal{F}^+} T' \iff \forall\,(v,V) \in \mathcal{F}^+(T)\colon \exists u \in \{\varepsilon\} \cup \downarrow V\colon (v\,u, u^{-1}V) \in \mathcal{F}^+(T')\ . \tag{3}$$

For instance, in Ex. 4.6 we have $B \sqsubseteq_{\mathcal{F}^+} C \sqsubseteq_{\mathcal{F}^+} B$; in other words, the systems presented there cannot be distinguished by $\sqsubseteq_{\mathcal{F}^+}$. We arrive at the following hierarchy of pre-orders.

Proposition 4.7 *$\mathcal{F}^+$-inclusion implies $\sqsubseteq_{\mathcal{F}^+}$ implies $\sqsubseteq_{\mathcal{F}}$ implies $\sqsubseteq_{\mathcal{L}}$.*

Proof. The only non-trivial statement is that $\sqsubseteq_{\mathcal{F}^+}$ implies $\sqsubseteq_{\mathcal{F}}$. This can be seen by observing that $\mathcal{F}(T) \subseteq \mathcal{F}^+(T)$ and that any standard failure is always a $\preceq$-maximum within a given set of tree failures: for if $(v,A) \preceq (w,W)$ for some tree failure (w,W) then $w = v\,u$ for some $u \in \{\varepsilon\} \cup \downarrow A = \{\varepsilon\} \cup A$ and $W = u^{-1}A$. If $u \in A$ then $W = \{\varepsilon\}$, which contradicts $W \subseteq \mathbf{A}^+$. It follows that $u = \varepsilon$, hence $(v,A) = (w,W)$. □

The following theorem states that the closure of $\mathcal{F}^+$ under $\preceq$ provides precisely the necessary abstraction to capture should-testing.

Theorem 4.8 $\sqsubseteq_{\mathbf{shd}} = \sqsubseteq_{\mathcal{F}^+}$.

Proof. Below, we use $\mathcal{L}(t)$ to denote the language of a test $t \in \mathbf{L}_{\checkmark}$. In this case, the words $w \in \mathcal{L}(t)$ range over $\mathbf{A}^*_{\checkmark}$ rather than $\mathbf{A}^*$.

$\subseteq$ This is due to the fact that we can mimic the failure of a should-test quite closely by the presence of a tree failure. Recursively define a family of tests $t_{v,V}$ with $v \in \mathbf{A}^*$ and $V \subseteq \mathbf{A}^*$ by

$$\begin{aligned} t_{\varepsilon,V} &= X_V \\ t_{a\,w,V} &= \checkmark;\mathbf{0} + a; t_{w,V} \end{aligned}$$

where the process variables X_V are to be interpreted according to the following definition:

$$X_V := \textstyle\sum\{\checkmark;\mathbf{0} \mid \varepsilon \in V\} + \sum\{a; X_{a^{-1}V} \mid a \in \downarrow V\}$$

It follows that $\mathcal{L}(X_V) = \downarrow(V\checkmark)$ for all $V \subseteq \mathbf{A}^*$.[4] Now assume $I \sqsubseteq_{\mathbf{shd}} S$ and $(v,V) \in \mathcal{F}^+(I)$. Then $I \stackrel{v}{\Longrightarrow} I'$ such that $\mathcal{L}(I') \cap V = \emptyset$; hence $I \| t_{v,V} \stackrel{v}{\Longrightarrow} I' \| X_V$ and $\nexists v'\colon I' \| X_V \stackrel{v'\checkmark}{\Longrightarrow}$. It follows that $\neg(I\ \mathbf{shd}\ t_{v,V})$, and hence $\neg(S\ \mathbf{shd}\ t_{v,V})$. The latter implies that $S \,\|\, t_{v,V} \stackrel{w}{\Longrightarrow} S' \,\|\, t'$ such that $\nexists w'\colon S' \| t' \stackrel{w'\checkmark}{\Longrightarrow}$. This implies $w = v\,u$ for some $u \in \{\varepsilon\} \cup \downarrow V$ such that $t' = X_{u^{-1}V}$. Since $t' \stackrel{w'\checkmark}{\Longrightarrow} t''$ iff $w' \in u^{-1}V$, we may conclude $\mathcal{L}(S') \cap u^{-1}V = \emptyset$; hence $(w, u^{-1}V) \in \mathcal{F}^+(S)$.

[4] Note that this is a case as discussed in Section 2.6, where $dom(\theta)$ is uncountable; as described there, the invocations X_V are not to be interpreted as $\mathrm{rec}_{X_V}\,\theta$ but rather as $\mathrm{rec}_{X_V}(\theta \upharpoonright \downarrow_\theta X_V)$. To be exact, we have to restrict attention to languages V over a countable subset of $\mathbf{A}$, which also makes the sum in the definition of X_V well-defined.

$$
\begin{array}{rcl}
\mathcal{F}^+(a;B) & = & \{(\varepsilon,V) \mid (\varepsilon,a^{-1}V) \in \mathcal{F}^+(B)\} \cup \{(a\,w,V) \mid (w,V) \in \mathcal{F}^+(B)\} \\
\mathcal{F}^+(\tau;B) & = & \mathcal{F}^+(B) \\
\mathcal{F}^+(\sum \mathbf{B}) & = & \{(w,V) \mid \forall B \in \mathbf{B}\colon (w,V) \in \mathcal{F}^+(B)\} \\
 & & \cup\, \{(w,V) \in \mathcal{F}^+(B) \mid B \in \mathbf{B}, w \in \mathbf{A}^+ \vee \neg B\ \mathbf{stb}\} \\
\mathcal{F}^+(B[\varphi]) & = & \{(\varphi(w),V) \mid (w,\varphi^{-1}(V)) \in \mathcal{F}^+(B)\} \\
\mathcal{F}^+(B/A) & = & \{(\pi_{\bar{A}}(w),V) \mid (w,\pi_{\bar{A}}^{-1}(V)) \in \mathcal{F}^+(B)\} \\
\mathcal{F}^+(B \parallel_A C) & = & \{(w, V_B \cup V_C) \mid (\pi_{\overline{\mathcal{S}(C)\setminus A}}(w), \pi_{\overline{\mathcal{S}(C)\setminus A}}(V_B)) \in \mathcal{F}^+(B), \\
 & & \qquad (\pi_{\overline{\mathcal{S}(B)\setminus A}}(w), \pi_{\overline{\mathcal{S}(B)\setminus A}}(V_C)) \in \mathcal{F}^+(C)\} \quad \text{if } \mathcal{S}(B) \cap \mathcal{S}(C) \subseteq A
\end{array}
$$

Table 4.2: Compositional construction of tree failures

$\supseteq$ Assume $I \sqsubseteq_{\mathcal{F}^+} S$ and $\neg(I\ \mathbf{shd}\ t)$. It follows that $I \parallel t \stackrel{v}{\Longrightarrow} I' \parallel t'$ such that $\nexists v'\colon I' \parallel t' \stackrel{v'\surd}{\Longrightarrow}$. Let $V = \{v' \mid v'\surd \in \mathcal{L}(t')\}$; then $(v,V) \in \mathcal{F}^+(I)$. Hence $(w,W) \in \mathcal{F}^+(S)$ such that $w = v\,u$ for some $u \in \{\varepsilon\} \cup {\downarrow}V$ and $W = u^{-1}V$. This implies $S \stackrel{w}{\Longrightarrow} S'$ such that $\mathcal{L}(S') \cap W = \emptyset$. Since $u \in \mathcal{L}(t')$, there is a $t' \stackrel{u}{\Longrightarrow} t''$; it follows that $S \parallel t \stackrel{w}{\Longrightarrow} S' \parallel t''$. Due to $\{u' \mid u'\surd \in \mathcal{L}(t'')\} \subseteq u^{-1}V = W$ we have $\nexists u'\colon S' \parallel t'' \stackrel{u'\surd}{\Longrightarrow}$, implying $\neg(S\ \mathbf{shd}\ t)$. $\Box$

Note that the construction of $t_{w,V}$ in the first part of the proof uses both infinitary summation and an infinite process environment. In fact, this theorem is the reason why we included those constructs in our language in the first place.

In analogy with Corollary 4.3, which tranfers the results about $\sqsubseteq_{\mathbf{may}}$ and $\sqsubseteq_{\mathbf{acc}}$ in the testing framework to $\sqsubseteq_{\mathcal{L}}$ and $\sqsubseteq_{\mathcal{F}}$ in the denotational setting, using Theorem 4.8 we can also transfer Theorem 3.12 about $\sqsubseteq_{\mathbf{shd}}$ to $\sqsubseteq_{\mathcal{F}^+}$:

Corollary 4.9 *$\sqsubseteq_{\mathcal{F}^+} \cap \sqsubseteq_{\mathbf{stb}}$ is a first-order pre-congruence.*

In fact, the denotational construction of the tree failures appears as a straightforward extension of the failure constructions. They are given in Table 4.2, which should be compared with Table 4.1.

The following is straightforward to prove:

Proposition 4.10 *All equations in Table 4.2 are sound.*

Note that this in itself does not yet imply that $\sqsubseteq_{\mathcal{F}^+}$ is a pre-congruence, merely (because all the constructions are pointwise) that $\mathcal{F}^+$-inclusion is one. The denotational pre-congruence proof of $\sqsubseteq_{\mathcal{F}^+}$ is delayed until we have fully introduced the denotational model.

Alternative definition of $\sqsubseteq_{\mathcal{F}^+}$. An alternative characterisation $\sqsubseteq_{\mathcal{F}^+}$ is obtained by using the $\preceq$-downward-closure of $\mathcal{F}^+$ as a model and the subset relation as the corresponding ordering. This is in fact the original formulation in [41], which we also used in [10, 11]. More precisely, one defines

$$\mathcal{F}^{++}(T) = \{(w,V) \in \mathcal{L}(T) \times \mathcal{P}(\mathbf{A}^+) \mid \exists v \in {\downarrow}V\colon \exists (q \stackrel{w\,v}{\Longrightarrow} s)\colon \mathcal{L}(s) \cap v^{-1}V = \emptyset\}$$

and regards set inclusion over this model as a semantic relation:

$$B \sqsubseteq_{\mathcal{F}^{++}} C \ :\Leftrightarrow\ \mathcal{F}^{++}(B) \subseteq \mathcal{F}^{++}(C)\ .$$

It is not difficult to see that $\mathcal{F}^{++}(B) = {\downarrow_{\preceq}}\mathcal{F}^+(B)$ and hence $\sqsubseteq_{\mathcal{F}^+} = \sqsubseteq_{\mathcal{F}^{++}}$. The reason why we chose the simpler model with the more complex ordering is to have better model constructions (see Table 4.2) directly extending those for standard failures (Table 4.1)— although the price is a more involved proof of pre-congruence, especially for parallel composition (Lemma 4.18 below). Also, $\sqsubseteq_{\mathcal{F}^+}$ corresponds more closely to the way the decision algorithm works (Section 5 below).

4.3 The denotational model

After having given the intuitions and preliminary definitions, we now switch to a purely denotational perspective. We have seen that the denotation of terms has to take three factors into account:

- A system satisfies more tests if and only if it has fewer tree failures.
- In order to obtain a pre-congruence with respect to choice, initial stability has to be preserved.
- In order to obtain a recursion pre-congruence, language equivalence has to be preserved.

We first consider tree failures sets as objects in their own right.

The space of tree failures. We now give an explicit definition of the sets of tree failures that constitute valid models. This *space* of tree failures is denoted $\mathbf{T}$; we use $\mathcal{F}, \mathcal{G}$ to range over $\mathbf{T}$.

In order for $\mathbf{T}$ to be adequate, we impose a number of saturation conditions on elements $\mathcal{F} \in \mathbf{T}$, which extend the closure conditions on failure sets; see, e.g., [13]:

- $\mathcal{F}$ may not be empty: even a completely deadlocked system has failures (namely, at least $(\varepsilon, \emptyset)$ is a failure).
- The refusal part of $\mathcal{F}$ is saturated in the sense that the upward-closure $\uparrow V$ of the refusal set V of any tree failure $(v, V) \in \mathcal{F}$ again gives rise to a tree failure, namely $(v, \uparrow V) \in \mathcal{F}$. We call this particular property *suffix saturation*; it reflects the fact that if the traces in V can be refused in a given state, then all *longer* traces can naturally also be refused.
- The refusal part of $\mathcal{F}$ is also saturated in the sense that the refusal set V of any $(v, V) \in \mathcal{F}$ can be decreased arbitrarily: that is, $(v, W) \in \mathcal{F}$ for all $W \subseteq V$.
- The refusal part of $\mathcal{F}$ is also saturated in the sense that the refusal set V of any $(v, V) \in \mathcal{F}$ can be extended to any set of traces w such that $(v\,w, w^{-1}V) \notin \mathcal{F}$. (Note that all $w \in V$ themselves satisfy this criterion, since then $\varepsilon \in w^{-1}V$ and no refusal set can contain ε.) We call this particular property *extension saturation*. The intuition behind it is the following: if state at which V can be refused had such a w-path, this would apparently lead to a state where some $u \in w^{-1}V$ is enabled; this, however, contradicts the refusal of V. Hence there is no such w-path.
- The trace part of $\mathcal{F}$ is saturated in the sense that for any tree failure $(v, V) \in \mathcal{F}$ and any prefix $w \sqsubseteq v$, there is also a tree failure $(w, \emptyset) \in \mathcal{F}$.

We thus arrive at the following definition of the model space:

$$\begin{aligned}\mathbf{T} = \{\mathcal{F} \subseteq \mathbf{A}^* \times \mathcal{P}(\mathbf{A}^+) \mid\ & \mathcal{F} \neq \emptyset \text{ and for all } (v, V) \in \mathcal{F}: \\ & (v, \uparrow V) \in \mathcal{F}, \\ & \forall W \subseteq V: (v, W) \in \mathcal{F}, \\ & \forall W \supseteq V: (\forall w \in W: (v\,w, w^{-1}V) \notin \mathcal{F}) \Rightarrow (v, W) \in \mathcal{F}, \\ & \forall w \sqsubseteq v: (w, \emptyset) \in \mathcal{F}\ \}\end{aligned}$$

$\mathbf{T}$ is interpreted under the pre-order $\sqsubseteq_{\mathbf{T}}$, which carries the definition of $\sqsubseteq_{\mathcal{F}^+}$ over to $\mathbf{T}$ (see (3)):

$$\mathcal{F} \sqsubseteq_{\mathbf{T}} \mathcal{G} \ :\Leftrightarrow\ \forall (v, V) \in \mathcal{F}: \exists u \in \{\varepsilon\} \cup \downarrow V: (v\,u, u^{-1}V) \in \mathcal{G}\ .$$

The following proposition (the proof of which is left to the reader) states that we have indeed correctly characterised the model space.

Proposition 4.11 $\mathcal{F}^+(B) \in \mathbf{T}$ *for all closed* $B \in \mathbf{L}$.

The dual property (i.e., that we have also *completely* characterised the model space, in the sense that all $\mathcal{F} \in \mathbf{T}$ correspond to the set of tree failures of some closed $B \in \mathbf{L}$), is discussed in Appendix A; we regard this to be of limited interest, and the proof is very technical.

The full model. As recalled above, we already know that, apart from the tree failures, the initial stability of terms must be encoded in their denotations; moreover, language equality is a necessary condition for semantic inclusion. This gives rise to the following definition of $\mathbf{M}$, the space of denotational models. Moreover, in the remainder of this section we use $\mathbf{W}$ to denote the universe of non-empty prefix-closed sets of words.

$$\begin{array}{rcl} \mathbf{M} & = & \mathbf{T} \times \mathbb{B} \\ \mathbf{W} & = & \{W \subseteq \mathbf{A}^* \mid W \neq \emptyset, {\downarrow}W = W\} \ . \end{array}$$

The ordering over $\mathbf{M}$ accordingly extends $\sqsubseteq_{\mathbf{T}}$ with the requirement of stability preservation and language equivalence.

$$\langle \mathcal{F}_1, P_1 \rangle \sqsubseteq_{\mathbf{M}} \langle \mathcal{F}_2, P_2 \rangle \ :\Leftrightarrow \ (\mathcal{F}_1 \sqsubseteq_{\mathbf{T}} \mathcal{F}_2) \wedge (P_1 \Leftarrow P_2) \wedge (\mathcal{L}(\mathcal{F}_1) = \mathcal{L}(\mathcal{F}_2)) \ .$$

We can derive the tree failure semantics operationally as follows (for any $B \in \mathbf{L}^\bullet$):

$$\mathcal{O}(B) \ := \ \langle \mathcal{F}^+(B), B \ \mathbf{stb} \rangle \ .$$

By combining Proposition 4.2, Proposition 4.7 and Theorem 4.8, we get:

Theorem 4.12 *For all closed* $B, C \in \mathbf{L}$, $B \sqsubseteq^c_{\mathbf{shd}} C$ *iff* $\mathcal{O}(B) \sqsubseteq_{\mathbf{M}} \mathcal{O}(C)$.

In the remainder of this section, essentially $\langle \mathbf{M}, \sqsubseteq_{\mathbf{M}} \rangle$ will be our denotational semantic domain. However, this domain deviates from what is usual in denotational semantics in several respects, which are potentially troublesome in the fixpoint construction needed to define the semantics of recursion:

1. $\mathbf{M}$ does not have all least upper bounds with respect to $\sqsubseteq_{\mathbf{M}}$;
2. In particular, $\mathbf{M}$ does not have a bottom (smallest) element with respect to $\sqsubseteq_{\mathbf{M}}$;
3. $\sqsubseteq_{\mathbf{M}}$ is a pre-order rather than a partial order, so least upper bounds are in general not uniqe even if they do exist.

The last point turns out to be less of a problem: obviously least upper bounds (when they exist) are unique up to the kernel of the pre-order, $\simeq_{\mathbf{M}} = \sqsubseteq_{\mathbf{M}} \cap \sqsupseteq_{\mathbf{M}}$, and since due to Theorem 4.12 this coincides with our semantic equivalence over terms, $\simeq^c_{\mathbf{shd}}$, this is good enough for our purpose.

The other points can be solved by restricting the fixpoint constructions to sub-domains of $\mathbf{M}$ in which all tree failure sets have the same language. That is, for all $W \in \mathbf{W}$, we define a subset of models, $\mathbf{M}_W$, that contains just the models with language W; moreover, in $\mathbf{M}_W$ we distinguish a special element $\bot_W$, that will turn out to be a smallest element (in $\mathbf{M}_W$).

$$\begin{array}{rcl} \mathbf{M}_W & = & \{\langle \mathcal{F}, P \rangle \in \mathbf{M} \mid \mathcal{L}(\mathcal{F}) = W\} \\ \bot_W & = & (\{(w, V) \mid w \in W, V \cap w^{-1}W = \emptyset\}, \mathtt{tt}) \ . \end{array}$$

First we state that $\bot_W$ is indeed a smallest element. The following is straightforward to prove.

Proposition 4.13 *For every $W \in \mathbf{W}$ and $\langle \mathcal{F}, P\rangle \in \mathbf{M}_W$, $\bot_W \sqsubseteq_{\mathbf{M}} \langle \mathcal{F}, P\rangle$.*

Note that therefore $\bot_W$ is a least upper bound of the empty set. For non-empty I we define

$$\bigsqcup_{i\in I} \langle \mathcal{F}_i, P_i\rangle = \langle \bigcup_{i\in I} \mathcal{F}_i, \bigwedge_{i\in I} P_i\rangle$$

The following proposition states that this yields a least upper bound.

Proposition 4.14 *Let $I \neq \emptyset$ and $\langle \mathcal{F}_i, P_i\rangle_i \in \mathbf{M}_W$ for all $i \in I$; then $\bigsqcup_{i\in I} \langle \mathcal{F}_i, P_i\rangle \in \mathbf{M}_W$ is a least upper bound with respect to $\sqsubseteq_{\mathbf{M}}$.*

Proof. First note that $\bigcup_i \mathcal{F}_i \in \mathbf{T}$ and $\mathcal{L}(\bigcup_i \mathcal{F}_i) = \bigcup_i \mathcal{L}(\mathcal{F}_i) = W$, and hence $\bigsqcup_i \langle \mathcal{F}_i, P_i\rangle \in \mathbf{M}_W$.

Let $\langle \mathcal{F}, P\rangle = \bigsqcup_i \langle \mathcal{F}_i, P_i\rangle$. It is clear that $\langle \mathcal{F}_i, P_i\rangle \sqsubseteq_{\mathbf{M}} \langle \mathcal{F}, P\rangle$ for all i; hence $\langle \mathcal{F}, P\rangle$ is an upper bound. Now assume $\forall i : \langle \mathcal{F}_i, P_i\rangle \sqsubseteq_{\mathbf{M}} \langle \mathcal{G}, Q\rangle$ for some $\langle \mathcal{G}, Q\rangle \in \mathbf{M}$. Let $(v, W) \in \bigcup_i \mathcal{F}_i$ be arbitrary. It follows that $(v, W) \in \mathcal{F}_i$ for some i; hence (due to $\mathcal{F}_i \sqsubseteq_{\mathbf{T}} \mathcal{G}$) we have $\exists u \in \{\varepsilon\} \cup \downarrow V$ such that $(v\,u, u^{-1}V) \in \mathcal{G}$. We may conclude $\mathcal{F} \sqsubseteq_{\mathbf{T}} \mathcal{G}$. Furthermore, it is also straightforward to see that $Q \Rightarrow P$; hence $\langle \mathcal{F}, P\rangle \sqsubseteq_{\mathbf{M}} \langle \mathcal{G}, Q\rangle$. □

It follows from the propositions above that, for every $W \in \mathbf{W}$, the *quotient model* $\mathbf{M}_W/{\simeq_{\mathbf{M}}}$ (which is the set of $\simeq_{\mathbf{M}}$-equivalence classes $[\langle \mathcal{F}, P\rangle]_{\simeq_{\mathbf{M}}} = \{\langle \mathcal{G}, Q\rangle \mid \langle \mathcal{G}, Q\rangle \simeq_{\mathbf{M}} \langle \mathcal{F}, P\rangle\}$ for all $\langle \mathcal{F}, P\rangle \in \mathbf{M}_W$) is a complete lattice under the ordering $\sqsubseteq$ defined by $[\langle \mathcal{F}, P\rangle]_{\simeq_{\mathbf{M}}} \sqsubseteq [\langle \mathcal{G}, Q\rangle]_{\simeq_{\mathbf{M}}}$ iff $\langle \mathcal{F}, P\rangle \sqsubseteq_{\mathbf{M}} \langle \mathcal{G}, Q\rangle$. This observation can serve as an intuition and justification for our denotational semantics, even though, for the sake of simplicity, we carry out the actual constructions and proofs in $\mathbf{M}$. In particular, the $\bigsqcup$-construction defined above, lifted to the quotient model, indeed yields the (unique) least upper bound.

First-order operators. Since we have already proved, using only the operational test framework, that $\sqsubseteq^c_{\mathbf{shd}}$ is a first-order pre-congruence (Theorems 3.1 and 3.12), this is an indication that there exist constructions over $\mathbf{M}$ that correspond to the first-order $\mathbf{L}$-operators and are monotonic with respect to $\sqsubseteq_{\mathbf{M}}$. The definitions are given in Table 4.3.

The constructions over $\mathbf{T}$ in Table 4.3 are in fact strongly analogous to the equalities in Table 4.2. The first task is to show that all these constructions stay within the denotational model space. The proof is left to the reader.

Lemma 4.15 *The constructions $a;\mathcal{F}$, $\bigoplus_i \langle \mathcal{F}_i, P_i\rangle$, $\mathcal{F}[\varphi]$ and $\mathcal{F} \parallel_A \mathcal{G}$ in Table 4.3 map into $\mathbf{T}$.*

Note that the construction for parallel composition in Table 4.3 yields the same result when we replace $\mathcal{A}(\mathcal{F})$ throughout by some $A_1 \supseteq \mathcal{A}(\mathcal{F})$ and $\mathcal{A}(\mathcal{G})$ by some $A_2 \supseteq \mathcal{A}(\mathcal{G})$, provided $A_1 \cap A_2 \subseteq A$. Hence, we can also apply this construction to $\mathcal{F}^+(B)$ and $\mathcal{F}^+(C)$ if we only know the static sorts $\mathcal{S}(B)$ and $\mathcal{S}(C)$ and not the (potentially smaller) dynamic alphabets $\mathcal{A}(\mathcal{F}^+(B))$ and $\mathcal{A}(\mathcal{F}^+(C))$.

The constructions in Table 4.3 are all clearly monotonic with respect to set inclusion on the set of tree failures. Since, however, $\sqsubseteq_{\mathbf{M}}$ is not based on a simple set inclusion (like $\sqsubseteq_{\mathcal{F}}$) but relies on a form of saturation, it does not immediately follow from this observation that these constructions preserve $\sqsubseteq_{\mathbf{M}}$. Nevertheless, a more careful analysis shows that the desired property does hold. (In terms of the quotient model discussed above, this preservation property is necessary to establish that the constructions are well-defined in $\mathbf{M}/{\simeq_{\mathbf{M}}}$.)

Proposition 4.16 *The constructions over $\mathbf{M}$ defined in Table 4.3 are monotonic with respect to $\sqsubseteq_{\mathbf{M}}$.*

Constructions over $\mathbf{T}$

$$
\begin{aligned}
a;\mathcal{F} &= \{(\varepsilon, V) \mid (\varepsilon, a^{-1}V) \in \mathcal{F}\} \cup \{(a\,w, V) \mid (w, V) \in \mathcal{F}\} \\
\textstyle\bigoplus_i \langle \mathcal{F}_i, P_i\rangle &= \textstyle\bigcap_i \mathcal{F}_i \cup \bigcup_i \{(w, V) \in \mathcal{F}_i \mid w \in \mathbf{A}^+ \vee \neg P_i\} \\
\mathcal{F}[\varphi] &= \{(\varphi(w), V) \mid (w, \varphi^{-1}(V)) \in \mathcal{F}\} \\
\mathcal{F}/A &= \{(\pi_{\bar{A}}(w), V) \mid (w, \pi_{\bar{A}}^{-1}(V)) \in \mathcal{F}\} \\
\mathcal{F} \parallel_A \mathcal{G} &= \{(w, V \cup W) \mid (\pi_{\overline{\mathcal{A}(\mathcal{G})\setminus A}}(w), \pi_{\overline{\mathcal{A}(\mathcal{G})\setminus A}}(V)) \in \mathcal{F}, \\
&\qquad\qquad (\pi_{\overline{\mathcal{A}(\mathcal{F})\setminus A}}(w), \pi_{\overline{\mathcal{A}(\mathcal{F})\setminus A}}(W)) \in \mathcal{G}\} \quad \text{if } \mathcal{A}(\mathcal{F}) \cap \mathcal{A}(\mathcal{G}) \subseteq A
\end{aligned}
$$

Constructions over $\mathbf{M}$

$$
\begin{aligned}
a;\langle \mathcal{F}, P\rangle &= \langle a;\mathcal{F}, \mathtt{tt}\rangle \\
\tau;\langle \mathcal{F}, P\rangle &= \langle \mathcal{F}, \mathtt{ff}\rangle \\
\textstyle\sum_i \langle \mathcal{F}_i, P_i\rangle &= \langle \textstyle\bigoplus_i \langle \mathcal{F}_i, P_i\rangle, \bigwedge_i P_i\rangle \\
\langle \mathcal{F}, P\rangle[\varphi] &= \langle \mathcal{F}[\varphi], P\rangle \\
\langle \mathcal{F}, P\rangle/A &= \langle \mathcal{F}/A, P \wedge (A \cap \mathcal{L}(\mathcal{F}) = \emptyset)\rangle \\
\langle \mathcal{F}, P\rangle \parallel_A \langle \mathcal{G}, Q\rangle &= \langle \mathcal{F} \parallel_A \mathcal{G}, P \wedge Q\rangle
\end{aligned}
$$

Table 4.3: Model constructions on $\mathbf{T}$ and $\mathbf{M}$

For the proof, we restrict ourselves to the tree failures component of $\mathbf{M}$; monotonicity w.r.t. stability is straightforward to prove, whereas preservation of language equality follows from standard theory (see Corollary 4.3). We show the required property for relabelling and synchronisation; the case for hiding is analogous to relabelling.

Lemma 4.17 *If $\mathcal{F} \sqsubseteq_{\mathbf{T}} \mathcal{G}$ then $\mathcal{F}[\varphi] \sqsubseteq_{\mathbf{T}} G[\varphi]$.*

Proof. Assume $(v, W) \in \mathcal{F}[\varphi]$.

- By definition (Table 4.3), $v = \varphi(w)$ for some $(w, \varphi^{-1}(W)) \in \mathcal{F}$.
- $\mathcal{F} \sqsubseteq_{\mathbf{M}} \mathcal{G}$ then implies $(w\,u, u^{-1}\varphi^{-1}(W)) \in \mathcal{G}$ for some $u \in \{\varepsilon\} \cup {\downarrow}\varphi^{-1}(W)$.
- It can be shown that $\varphi^{-1}(\varphi(u)^{-1}W) = u^{-1}\varphi^{-1}(W)$.
- By definition (Table 4.3), it follows that $(\varphi(w\,u), \varphi(u)^{-1}W) \in \mathcal{G}[\varphi]$.
- Since $\varphi(w\,u) = \varphi(w)\varphi(u)$ and $\varphi(u) \in \{\varepsilon\} \cup {\downarrow}W$, we are done. □

The monotonicity for synchronisation is by far the most complex to prove. Let us regard for a moment the special case of full synchronisation (i.e., synchronisation set $\mathbf{A}$) so that the string homomorphisms $\pi_{\overline{\mathcal{A}(\mathcal{F})\setminus A}}$ and $\pi_{\overline{\mathcal{A}(\mathcal{G})\setminus A}}$ in Table 4.3 equal the identity function and can hence be omitted. The first proof idea is as follows. If $\mathcal{F}_i \sqsubseteq_{\mathbf{M}} \mathcal{G}_i$ for $i = 1, 2$, then any tree failure $(w, V) \in \mathcal{F}_1 \parallel \mathcal{F}_2$ is constructed from $(w, V_i) \in \mathcal{F}_i$ such that $V = V_1 \cup V_2$; these in turn imply the existence of elements of the $\mathcal{G}_i$ of the form

$$(w\,u_1, u_1^{-1}V_1) \in \mathcal{G}_1 \qquad (w\,u_2, u_2^{-1}V_2) \in \mathcal{G}_2$$

where $u_i \in \{\varepsilon\}{\downarrow}V_i$ for $i = 1, 2$. However, it is *not* automatically the case that $u_1 = u_2$ or even that one is a prefix of the other, and hence these $\mathcal{G}_i$-elements do not immediately combine into an element of $\mathcal{G}_1 \parallel \mathcal{G}_2$. Consequently, the monotonicity proof is a good deal more involved than this simple idea, and we refer it to Appendix B (Page 68).

Lemma 4.18 *If $\mathcal{F}_i \sqsubseteq_{\mathbf{T}} \mathcal{G}_i$ for $i = 1, 2$ with $\mathcal{A}(\mathcal{G}_1) \cap \mathcal{A}(\mathcal{G}_2) \subseteq A$, then $\mathcal{F}_1 \parallel_A \mathcal{F}_2 \sqsubseteq_{\mathbf{T}} \mathcal{G}_1 \parallel_A \mathcal{G}_2$.*

4.4 The denotational semantics

We now set out to develop the actual denotational semantics. For this purpose, we first prove continuity of our **M**-constructions. Unfortunately, here it turns out that the infinite choice operator is *not* continuous in any obvious sense, and for that reason, we prove the results of this subsection for the language with finite sums only.

Continuity. First let us clarify what we mean by continuity in a setting where least upper bounds are not unique. We define this as preservation of the construction $\bigsqcup$, defined above, that picks a particular representative from the set of (equivalent) least upper bounds, modulo the kernel equivalence $\simeq_{\mathbf{M}}$. Let us call a subset of $\mathbf{M}$ *consistent* if it has an upper bound (as we have seen above, this is the case if and only if it is actually a subset of some $\mathbf{M}_W$).

Definition 4.19 (continuity) *A function $f\colon \mathbf{M} \to \mathbf{M}$ is* continuous *if for any consistent subset $\{x_i\}_{i\in I} \subseteq \mathbf{M}$*

$$f(\bigsqcup_i x_i) \;\simeq_{\mathbf{M}}\; \bigsqcup_i f(x_i) \ .$$

f is chain-continuous *if the above property holds for all cases where $\{x_i\}_{i\in I}$ is a* chain*, i.e., is linearly ordered according to $\sqsubseteq_{\mathbf{M}}$.*

Note that this is a necessary and sufficient condition for "real" [chain-]continuity of f lifted to the quotient model $\mathbf{M}/\simeq_{\mathbf{M}}$. In practice we will always use subsets of $\mathbb{N}$ as index sets, and the chain will correspond to the ordering of the indices; that is, $x_i \sqsubseteq_{\mathbf{M}} x_j$ iff $i \leq j$. The desired property is stated in the following proposition.

Proposition 4.20 *The constructions over $\mathbf{M}$ defined in Table 4.3 are chain-continuous.*

Proof. Since all our operators have finite arity (recalling that we have restricted summation to finite sets), they are essentially functions $g\colon \mathbf{M}^n \to \mathbf{M}$ (n being the arity of the function), which can be split into $f\colon \mathbf{T}^n \times \mathbb{B}^n \to \mathbf{T}$ and $p\colon \mathbf{T}^n \times \mathbb{B}^n \to \mathbb{B}$ such that

$$g(\langle \mathcal{F}_1, P_1\rangle, \ldots, \langle \mathcal{F}_n, P_n\rangle) = \langle f(\mathcal{F}_1, \ldots, \mathcal{F}_n, P_1, \ldots, P_n), p(\mathcal{F}_1, \ldots, \mathcal{F}_n, P_1, \ldots, P_n)\rangle \ .$$

Note that (taking $\mathbb{B}$ to be ordered in the standard fashion, with `ff` smaller than `tt`), the f are monotonic in their $\mathcal{F}_k$-parameters and anti-monotonic in the P_k, whereas the p are anti-monotonic in the $\mathcal{F}_k$ and monotonic in the P_k. This is consistent with the definition of $\sqsubseteq_{\mathbf{M}}$, which corresponds to set inclusion for the tree failure sets but *reverse* implication for the stability predicate.

We show continuity for f; that of p is proved along the same lines. Furthermore, since the P_k can only change once (from `tt` to `ff`), monotonicity actually implies continuity in these parameters. Thus, the proof can be reduced to showing that the f are $\sqsubseteq_{\mathbf{T}}$-chain-continuous in the $\mathcal{F}_k$.

For the purpose of this proof we write $\vec{\mathcal{F}} \in \mathbf{T}^n$ for vectors of tree failure sets, with elements $\mathcal{F}_k \in \mathbf{T}$ for $1 \leq k \leq n$; the elements of a vector $\vec{\mathcal{F}}_i$ are likewise denoted $\mathcal{F}_{k,i}$. We also write $\mathbf{Q} = \mathbf{A}^* \times \mathcal{P}(\mathbf{A}^*)$ for the set of which tree failures are subsets.

The crux of the proof is the observation that the functions f are all defined *pointwise* on the individual tree failures, in the sense that, for every vector $\vec{P} \in \mathbb{B}^n$, there is a pointwise construction $\varphi_{\vec{P}}\colon \mathbf{Q}^n \to \mathcal{P}(\mathbf{Q})$ satisfying, for all $\vec{\mathcal{F}} \in \mathbf{M}^n$

$$f(\vec{\mathcal{F}}, \vec{P}) = \bigcup \{\varphi_{\vec{P}}(\vec{q}) \mid \vec{q} \in \vec{\mathcal{F}}\}$$

where $\vec{q} \in \vec{\mathcal{F}}$ has to be understood componentwise. (In fact, with the exception of $\bigoplus$, the $\varphi_{\vec{p}}$ are actually insensitive to $\vec{p}$.)

Over $\mathbf{T}^n$ we define $\sqsubseteq$ as the component-wise extension over the subset relation (i.e., $\vec{\mathcal{F}} \sqsubseteq \vec{\mathcal{G}}$ iff $\mathcal{F}_k \subseteq \mathcal{G}_k$ for $1 \leq k \leq n$) and $\bigsqcup$ as the component-wise union (i.e., $\bigsqcup_i \vec{\mathcal{F}}_i = (\bigcup_i \mathcal{F}_{1,i}, \ldots, \bigcup_i \mathcal{F}_{n,i})$). We now show that for all $\sqsubseteq$-chains of vectors $\{\vec{\mathcal{F}}_i \in \mathbf{T}^n\}_i$,

$$f(\textstyle\bigsqcup_i \vec{\mathcal{F}}_i, \vec{P}) = \bigcup_i f(\vec{\mathcal{F}}_i, \vec{P}) \ .$$

- $\subseteq$ Assume $\mathcal{G} \in f(\bigsqcup_i \vec{\mathcal{F}}_i, \vec{P})$; hence $\mathcal{G} \in \varphi_{\vec{P}}(\vec{q})$ for some $\vec{q} \in \bigsqcup_i \vec{\mathcal{F}}$. It follows that for all $k \in \{1, \ldots, n\}$ there is some i_k such that $q_k \in \mathcal{F}_{k,i_k}$. Let $j = \max\{i_k \mid 1 \leq k \leq n\}$; due to the fact that all $\{\mathcal{F}_{k,i}\}_i$ are $\subseteq$-chains it follows that $q_k \in \mathcal{F}_{k,j}$ for all $1 \leq k \leq n$. But then $\vec{q} \in \vec{\mathcal{F}}_j$, hence $\mathcal{G} \in \bigcup\{\varphi_{\vec{P}}(\vec{q}) \mid \vec{q} \in \vec{\mathcal{F}}_j\} = f(\vec{\mathcal{F}}_j, \vec{P})$, implying $\mathcal{G} \in \bigcup_i f(\vec{\mathcal{F}}_i, \vec{P})$.
- $\supseteq$ Assume $G \in \bigcup_i f(\vec{\mathcal{F}}_i, \vec{P})$; hence $\mathcal{G} \in \varphi_{\vec{P}}(\vec{q})$ for some i and some $\vec{q} \in \vec{\mathcal{F}}_i$. Since clearly $\vec{\mathcal{F}}_i \sqsubseteq \bigsqcup_i \vec{\mathcal{F}}_i$, it follows that $\vec{q} \in \bigsqcup_i \vec{\mathcal{F}}_i$, hence $\mathcal{G} \in \bigcup\{\varphi_{\vec{P}}(\vec{q}) \mid \vec{q} \in \bigsqcup_i \vec{\mathcal{F}}_i\} = f(\bigsqcup_i \vec{F}_i, \vec{P})$.

This shows that f is $\sqsubseteq$-chain-continuous in the "tree failures" parameters. However, for the continuity of g we actually need $\sqsubseteq_{\mathbf{T}}$-chain-continuity of f. Fortunately, we can turn any $\sqsubseteq_{\mathbf{T}}$-chain of vectors $\{\vec{\mathcal{F}}_i\}_i$ into a $\sqsubseteq$-chain of equivalent vectors $\{\vec{\mathcal{G}}_i\}_i$ (i.e., with $\mathcal{G}_i \simeq_{\mathbf{T}} \mathcal{F}_i$ for all $i \in \mathbb{N}$) and the same component-wise union ($\bigsqcup_i \mathcal{F}_i = \bigsqcup_i \mathcal{G}_i$), as follows:

$$\mathcal{G}_{k,0} = \mathcal{F}_{k,0} \qquad \mathcal{G}_{k,i+1} = \mathcal{G}_{k,i} \cup \mathcal{F}_{k,i+1} \ .$$

$\mathcal{G}_{k,i} \simeq_{\mathbf{T}} \mathcal{F}_{k,i}$ follows by induction on i, using transitivity of $\sqsubseteq_{\mathbf{T}}$ and the fact that $\subseteq$ over $\mathbf{T}$ is stronger than $\sqsubseteq_{\mathbf{T}}$. It is not difficult to see (recalling that f is monotonic in the $\vec{\mathcal{F}}_k$-parameters) that necessarily $\bigcup_i f(\vec{\mathcal{F}}_i, \vec{P}) \simeq_{\mathbf{T}} \bigcup_i f(\vec{\mathcal{G}}_i, \vec{P})$. □

Fixpoints. In the following we discuss a fixpoint construction for recursion that is essentially standard domain theory, except that we keep all our constructions in the concrete, pre-ordered set $\mathbf{M}_W$. Let us first discuss the necessary concepts not for the specific domain $\mathbf{M}_W$ but for an arbitrary domain $\langle A, \sqsubseteq, \bigsqcup \rangle$ with $\sqsubseteq$ a pre-order and $\bigsqcup$ a construction that selects a representative least upper bound w.r.t. $\sqsubseteq$ for each $B \subseteq A$.

Let $\mathbf{Y}, \mathbf{Z}$ denote sets of variables. First recall that there is a standard extension of $\sqsubseteq$ from A to functions $\mathbf{Y} \to A$, according to which $f \sqsubseteq g$ for $f, g\colon \mathbf{Y} \to A$ iff $f(x) \sqsubseteq g(x)$ for all $x \in \mathbf{Y}$, and likewise for $\bigsqcup$. The notions of monotonicity and [chain-]continuity are extended accordingly to functions $(\mathbf{Z} \to A) \to (\mathbf{Y} \to A)$.

We use functions in $\Phi\colon (\mathbf{Z} \to A) \to (\mathbf{Y} \to A)$ to model the semantics of process environments defined on $\mathbf{Y}$ and with free variables in $\mathbf{Z}$ (where one may imagine that usually $\mathbf{Y} \subseteq \mathbf{Z}$). When such a process environment is closed recursively, essentially this provides values for the free variables in $x \in \mathbf{Y} \cap \mathbf{Z}$, by "feeding $\Phi(x)$ back into Φ".

To define this precisely, we alternatively portray Φ as a function $\Psi\colon (\mathbf{Z} \to A) \to (\mathbf{Y} \to A) \to (\mathbf{Y} \to A)$ defined by $\Psi(f)(g) = \Phi(f \lhd g)$, where $f \lhd g$, pronounced "f overwritten by g", acts as g where that is defined, and as f elsewhere — in other words, f is ignored whenever g offers an alternative. For every $f\colon \mathbf{Z} \to A$, $\Psi(f)$ is a function from $\mathbf{Y} \to A$ to $\mathbf{Y} \to A$, and this is where we can finally apply fixpoint theory. A function $g\colon \mathbf{Y} \to A$ is a *pre-fixpoint* of $\Psi(f)$ if $g \sqsubseteq \Psi(f)(g)$, and a *fixpoint* if $g \simeq \Psi(f)(g)$, where $\simeq = \sqsubseteq \cap \sqsupseteq$. A *least* fixpoint (if it exists) is a fixpoint that is "equal to or below" all others (w.r.t. $\sqsubseteq$).

As usual, if $\Psi(f)$ is continuous, then one can show that every least upper bound of all pre-fixpoints is a least fixpoint of $\Psi(f)$. This implies that if our original Φ is continuous, then every $\Psi(f)$ has a least fixpoint. In our case, where we have only chain-continuity, we construct a particular chain of pre-fixpoints and we define a mapping $\mu\Phi\colon (\mathbf{Z} \to A) \to (\mathbf{Y} \to A)$ yielding a particular least fixpoint $\mu\Phi(f)$ of $\Psi(f)$ for every $f\colon \mathbf{Z} \to A$. In terms of the original function Φ, $\mu\Phi$ satisfies, for every $f\colon \mathbf{Z} \to A$:

$$\mu\Phi(f) = \Phi(f \lhd \mu\Phi(f)) \ .$$

A more constructive characterisation of a least fixpoint is through approximation: define $\bot = \bigsqcup \emptyset$ and (for all $f\colon \mathbf{Z} \to A$)

$$\begin{aligned} \Phi^0 : f &\mapsto \{(x, \bot) \mid x \in \mathbf{Y}\} \\ \Phi^{i+1} : f &\mapsto \Phi(f \lhd \Phi^i(f)) \quad (i \geq 0) \\ \mu\Phi : f &\mapsto \textstyle\bigsqcup_i \Phi^i(f) \ . \end{aligned}$$

Proposition 4.21 *$\Phi\colon (\mathbf{Z} \to A) \to (\mathbf{Y} \to A)$ is a monotonic and chain-continuous function, then for every $f\colon \mathbf{Z} \to A$, $\mu\Phi(f)$ is a least fixpoint of $\Psi_f\colon g \mapsto \Phi(f \lhd g)$.*

Proof. First of all, due to the fact that $\bot$ is a smallest elements of A and hence $\Phi^0(f)$ is a smallest element of $\mathbf{Y} \to A$, plus monotonicity of Φ, it can be proved by induction on i that $\Phi^i(f) \sqsubseteq \Phi^{i+1}(f)$ for all i; hence $\{\Phi^i(f)\}_i$ is a chain, and hence so is $\{f \lhd \Phi^i(f)\}_i$.

The following sequence of equalities then shows that $\mu\Phi(f)$ is a fixpoint for Ψ_f:

$$\begin{aligned} \Psi_f(\mu\Phi(f)) = \Phi(f \lhd \textstyle\bigsqcup_i \Phi^i(f)) = \Phi(\bigsqcup_i (f \lhd \Phi^i(f))) &\simeq \textstyle\bigsqcup_i \Phi(f \lhd \Phi^i(f)) \\ &= \textstyle\bigsqcup_i \Phi^{i+1}(f) = \bigsqcup_i \Phi^i(f) = \mu\Phi(f) \end{aligned}$$

where the last step but one is due to the fact that, for any element $g\colon \mathbf{Y} \to A$, $\Phi^0(f) \sqcup g = g$ (where in this particular case $g = \bigsqcup_i \Phi^{i+1}(f)$).

If g is a fixpoint of Ψ, then clearly $\Phi^0(f) \sqsubseteq g$. By induction on i it can then be shown (using monotonicity of Φ and the fixpoint property of g) that $\Phi^i(f) \sqsubseteq g$ for all i. But then g is an upper bound to $\{\Phi^i(f)\}_i$, and hence $\mu\Phi(f) \sqsubseteq g$ by the properties of least upper bounds. This proves that $\mu\Phi(f)$ is a *least* fixpoint of Ψ_f. □

Proposition 4.22 *Let Φ be as in Proposition 4.21; then $\mu\Phi$ is monotonic and chain-continuous.*

Proof. By induction on i it can be proved that all Φ^i are monotonic and chain-continuous. Now assume $f, g\colon \mathbf{Z} \to A$ such that $f \sqsubseteq g$; then

$$\mu\Phi(f) = \textstyle\bigsqcup_i \Phi^i(f) \sqsubseteq \bigsqcup_i \Phi^i(g) = \mu\Phi(g) \ .$$

Similarly, let $\{f_k\colon \mathbf{Z} \to A\}_{k \in K}$ be a chain; then

$$\mu\Phi(\textstyle\bigsqcup_k f_k) = \bigsqcup_i \Phi^i(\bigsqcup_k f_k) \simeq \bigsqcup_i \bigsqcup_k \Phi^i(f_k) = \bigsqcup_k \bigsqcup_i \Phi^i(f_k) = \bigsqcup_k \mu\Phi(f_k) \ .$$

□

The semantic mapping. We are now ready to define the denotational semantics of $\mathbf{L}$ directly, rather than through the operational semantics as in Theorem 4.12. As usual, the denotational semantics takes the form of a mapping $[\![-]\!]_-: \mathbf{L} \to (\mathbf{X} \rightharpoonup \mathbf{M}) \rightharpoonup \mathbf{M}$, which for any term $B \in \mathbf{L}$ yields a function $[\![B]\!]_-: (fv(B) \to \mathbf{M}) \to \mathbf{M}$, the parameter of which provides the interpretation of the free variables of the term. Actually, in the sequel we will allow any function $\Omega: \mathbf{X} \rightharpoonup \mathbf{M}$ for which $dom(\Omega) \supseteq fv(B)$ to play the role of such an interpretation. We write $[\![B]\!]_\Omega$ ($\in \mathbf{M}$) for the semantics of B in the context Ω. If B is closed, we may omit the paramter Ω.

In addition, for the semantics of process environments θ we will use the same notation, where now we impose $[\![\theta]\!]_-: (\mathbf{Z} \to A) \to (\mathbf{Y} \to A)$ with $\mathbf{Y} = dom(\theta)$ and $\mathbf{Z} = fv(\theta)$. Thus, $[\![\theta]\!]_-$ has the shape of Φ in Propositions 4.21 and 4.22 above. This means that we can put Propositions 4.13 and 4.14 on the one hand, and Propositions 4.16 and 4.20 on the other, to good use: indeed for each $W \in \mathbf{W}$, $\langle \mathbf{M}_W, \sqsubseteq_{\mathbf{M}}, \bigsqcup \rangle$ is just such a domain as $\langle A, \sqsubseteq, \bigsqcup \rangle$ in Proposition 4.21, and functions $[\![\theta]\!]_-$ satisfy the required properties to construct least fixpoints.

However, we have cheated in one respect. Obviously the interpretations Ω are to play the role of the functions $f: \mathbf{Z} \to A$, but in fact Ω does not range over a single domain A; rather, for any $X \in fv(B)$ the interpretation has a different associated $W_X \in \mathbf{W}$ such that $\Omega(X) \in \mathbf{M}_{W_X}$. Though this makes no difference to our least upper bound construction, which as we have shown in Proposition 4.14 is the same irregardless of the trace set W, it does influence the construction of the bottom element, $\bot_{W_X}$. This does raise a problem, for where does W_X come from? Here we rely on another semantic mapping $\mathcal{L}[\![-]\!]_-$ that yields the *trace language* of terms.

Proposition 4.23 *There is a function $\mathcal{L}[\![-]\!]_-: \mathbf{L} \to (\mathbf{X} \rightharpoonup \mathbf{W}) \rightharpoonup \mathbf{W}$ such that for every closed $\sigma: \mathbf{X} \rightharpoonup \mathbf{L}$ with $dom(\sigma) \supseteq fv(B)$,*

$$\mathcal{L}[\![B]\!]_{\mathcal{L}[\![\sigma]\!]} = \mathcal{L}(B[\sigma]) \ .$$

We omit the actual definition of this function, since it is standard (in fact, it coincides with the "initial trace" part of the tree failures).

Thus the definition of the approximants and least fixpoint becomes, for every $\Omega: \mathbf{X} \rightharpoonup \mathbf{M}$ with $dom(\Omega) \supseteq fv(\theta) \setminus dom(\theta)$:

$$\begin{array}{rcl} [\![\theta]\!]^0_\Omega & = & \{(X, \bot_W) \mid X \in dom(\theta), W = \mathcal{L}[\![\mathrm{rec}_X\, \theta]\!]_{\mathcal{L}(\Omega)}\} \\ [\![\theta]\!]^{i+1}_\Omega & = & [\![\theta]\!]_{\Omega \lhd [\![\theta]\!]^i_\Omega} \quad (i \geq 0) \\ \mu[\![\theta]\!]_\Omega & = & \bigsqcup_i [\![\theta]\!]^i_\Omega \ . \end{array}$$

The following is an immediate corollary of Propositions 4.16 and 4.22.

Theorem 4.24 *For all $B \in \mathbf{L}$, $[\![B]\!]_-$ is $\sqsubseteq_{\mathbf{M}}$-monotonic.*

The remainder of this section is spent in proving the following characterisation theorem, which states the correspondence between the denotational semantics in Table 4.4 and the operational semantics of the previous section. Together with Theorems 4.12 and 4.24 this proves Theorem 3.15, i.e., in particular the recursive pre-congruence of $\sqsubseteq^c_{\mathbf{shd}}$.

Theorem 4.25

1. *For all $B \in \mathbf{L}$ and $\sigma: \mathbf{X} \rightharpoonup \mathbf{L}^\bullet$ with $dom(\sigma) \supseteq fv(B)$, $[\![B]\!]_{\mathcal{O}(\sigma)} \simeq_{\mathbf{M}} \mathcal{O}(B[\sigma])$.*
2. *For all $B \in \mathbf{L}^\bullet$, $[\![B]\!] \simeq_{\mathbf{M}} \mathcal{O}(B)$.*

$$
\begin{array}{c}
\hline
[\![\mathbf{0}]\!]_\Omega = \bot_{\{\varepsilon\}} \\
[\![\alpha; B]\!]_\Omega = \alpha; [\![B]\!]_\Omega \\
[\![\sum_i B_i]\!]_\Omega = \sum_i [\![B_i]\!]_\Omega \\
[\![B[\varphi]]\!]_\Omega = [\![B]\!]_\Omega[\varphi] \\
[\![B/A]\!]_\Omega = [\![B]\!]_\Omega/A \\
[\![B \parallel_A C]\!]_\Omega = [\![B]\!]_\Omega \parallel_A [\![C]\!]_\Omega \\
[\![X]\!]_\Omega = \Omega(X) \\
[\![\mathrm{rec}_X\, \theta]\!]_\Omega = \mu[\![\theta]\!]_\Omega(X) \\
\hline
\end{array}
$$

Table 4.4: Denotational semantic function

Obviously, the second clause is a direct consequence of the first. The proof is given below; it is by induction on the structure of B and, not surprisingly, the hard case is recursion. To prepare this, we essentially need an operational counterpart of the approximations $[\![\theta]\!]^i$. For arbitrary $\theta\colon \mathbf{X} \rightharpoonup \mathbf{L}$ with $fv(\theta) \subseteq dom(\theta)$ let

$$
\begin{array}{rcl}
\theta_{\mathbf{W}} & = & [\sum_{a\in W} a; Z_{a^{-1}W}/Z_W]_{W\in\mathbf{W}} \\
\theta^0 & = & [\mathrm{rec}_{Z_W}\, \theta_{\mathbf{W}}/X]_{X\in dom(\theta), W=\mathcal{L}(\mathrm{rec}_X\,\theta)} \\
\theta^{i+1} & = & \theta[\theta^i]\ .
\end{array}
$$

Intuitively, each Z_W recursively defined in $\theta_{\mathbf{W}}$[5] is a stable and completely deterministic process (no reachable state has an outgoing τ-transition or more than one outgoing a-transition for any given $a \in \mathbf{A}$) with W as its language; in other words, it is the operational equivalent of $\bot_W$. This is formulated in the following lemma.

Lemma 4.26 *For all $W \in \mathbf{W}$, $\mathcal{O}(\mathrm{rec}_{Z_W}\, \theta_{\mathbf{W}}) = \bot_W$.*

Proof. In the proof we write Z_W for $\mathrm{rec}_{Z_W}\, \theta_{\mathbf{W}}$. We first establish $Z_W \stackrel{w}{\Longrightarrow} B$ if and only if $w \in W$ and $B = Z_{w^{-1}W}$, by induction on the length of w. This implies $\mathcal{L}(Z_W) = W$.

Base case. $w = \varepsilon$. We have $Z_W \stackrel{w}{\Longrightarrow} B$ iff $B = Z_W$ since $Z_W \stackrel{\tau}{\not\longrightarrow}$; on the other hand, $w \in W$ due to $W \neq \emptyset$ and $W = {\downarrow}W$.

Induction step. $w = v\,a$ for some v and a.

If. If $w \in W$ then $v \in W$ by ${\downarrow}W = W$, and $a \in v^{-1}W$. By induction, $Z_W \stackrel{v}{\Longrightarrow} Z_{v^{-1}W}$, and it follows that $Z_{v^{-1}W} \stackrel{a}{\longrightarrow} Z_{a^{-1}v^{-1}W} = Z_{w^{-1}W}$; hence $Z_W \stackrel{w}{\longrightarrow} Z_{w^{-1}W}$.

Only if. If $Z_W \stackrel{w}{\longrightarrow} B$ then $Z_W \stackrel{v}{\Longrightarrow} B' \stackrel{a}{\Longrightarrow} B$. By induction, $v \in W$ and $B' = Z_{v^{-1}W}$. It follows that $Z_{v^{-1}W} \stackrel{a}{\Longrightarrow} B$; hence (by the definition of $Z_{v^{-1}W}$) $a \in v^{-1}W$ and $B = Z_{a^{-1}v^{-1}W} = Z_{w^{-1}W}$. The former implies $w = v\,a \in W$.

Z_W **stb** is immediate for all W. It remains to be proved that $\mathcal{F}_W = \mathcal{F}^+(Z_W)$.

$\subseteq$ Assume $w \in W$ and $w^{-1}W \cap V = \emptyset$. As proved above, $Z_W \stackrel{w}{\Longrightarrow} B$; it follows that $\mathcal{L}(B) \cap V \subseteq w^{-1}\mathcal{L}(Z_W) \cap V = w^{-1}W \cap V = \emptyset$.

[5] Interpreted as the countable restriction of $\theta_{\mathbf{W}}$ as discussed in Section 2.6. Again, we really should restrict ourselves to languages W that only use a countable subalphabet.

$\supseteq$ Assume $(w, V) \in \mathcal{F}^+(Z_W)$; i.e., there is a B such that $Z_W \stackrel{w}{\Longrightarrow} B$ and $\mathcal{L}(B) \cap V = \emptyset$. As proved above, $B = Z_{w^{-1}W}$, thus $\mathcal{L}(B) = w^{-1}W$. Thus, $w^{-1}W \cap V = \emptyset$.

□

The following is an important element in the proof of Theorem 4.25. (Note that we freely apply $\mathcal{O}(-)$ to mappings $\mathbf{X} \rightharpoonup \mathbf{L}^\bullet$; the result is a mapping $\mathbf{X} \rightharpoonup \mathbf{M}$ defined in the obvious way.)

Lemma 4.27 *Let $\theta\colon \mathbf{X} \rightharpoonup \mathbf{L}$ be such that $[\![\theta]\!]_{\mathcal{O}(\sigma)} \simeq_{\mathbf{M}} \mathcal{O}(\theta[\sigma])$ for all $\sigma\colon \mathbf{X} \rightharpoonup \mathbf{L}^\bullet$ with $dom(\sigma) \supseteq fv(\theta)$; then $[\![\theta]\!]^i_{\mathcal{O}(\sigma)} \simeq_{\mathbf{M}} \mathcal{O}(\theta[\sigma]^i)$ for all $i \geq 0$ and all $\sigma\colon \mathbf{X} \rightharpoonup \mathbf{L}^\bullet$ with $dom(\sigma) \supseteq fv(\theta) \setminus dom(\theta)$ and $dom(\sigma) \cap dom(\theta) = \emptyset$.*

Proof. By induction on i.

Base case. For $i = 0$ the property follows from

$$\begin{array}{rcl} [\![\theta]\!]^0_{\mathcal{O}(\sigma)} & = & \{(X, \bot_W) \mid X \in dom(\theta), W = \mathcal{L}[\![\mathrm{rec}_X\, \theta]\!]_{\mathcal{L}(\mathcal{O}(\sigma))}\} \\ & = & \{(X, \mathcal{O}(\mathrm{rec}_{Z_W}\, \theta_{\mathbf{W}})) \mid X \in dom(\theta), W = \mathcal{L}(\mathrm{rec}_X\, \theta[\sigma])\} \\ & = & \mathcal{O}(\theta[\sigma]^0) \end{array}$$

where the first equality is by definition of $[\![\theta]\!]^0$, the last by definition of $\theta[\sigma]^0$ and the second is due to a combination of Lemma 4.26 and

$$\mathcal{L}[\![\mathrm{rec}_X\, \theta]\!]_{\mathcal{L}(\mathcal{O}(\sigma))} = \mathcal{L}[\![\mathrm{rec}_X\, \theta]\!]_{\mathcal{L}[\![\sigma]\!]} = \mathcal{L}(\mathrm{rec}_X\, \theta[\sigma]) \ .$$

This, in turn, follows from two successive applications of Proposition 4.23, where for the first application we also use $\mathcal{L}(\mathcal{F}^+(B)) = \mathcal{L}(B)$ $(= \mathcal{L}[\![B]\!]$ by Proposition 4.23).

Induction step. For the induction step, we have

$$[\![\theta]\!]^{i+1}_{\mathcal{O}(\sigma)} = [\![\theta]\!]_{\mathcal{O}(\sigma) \triangleleft [\![\theta]\!]^i_{\mathcal{O}(\sigma)}} \simeq_{\mathbf{M}} [\![\theta]\!]_{\mathcal{O}(\sigma) \triangleleft \mathcal{O}(\theta[\sigma]^i)} \simeq_{\mathbf{M}} \mathcal{O}(\theta[\sigma \triangleleft \theta[\sigma]^i]) = \mathcal{O}(\theta[\sigma][\theta[\sigma]^i]) = \mathcal{O}(\theta[\sigma]^{i+1})$$

where the first step is by definition of $[\![\theta]\!]^{i+1}$, the second by the induction hypothesis (also using Theorem 4.24), the third by the assumption on θ, the fourth by the properties of substitution and the last by definition of θ^i.

□

Lemma 4.28 *Let $\theta\colon \mathbf{X} \rightharpoonup \mathbf{L}$ be such that $fv(\theta) \subseteq dom(\theta)$. For every $X \in dom(\theta)$ and $i \geq 0$ let X_i be a distinct, fresh name. Define $\theta^\omega, \theta^*\colon \{X_i \mid X \in dom(\theta), i \geq 0\} \to \mathbf{L}$ by*

$$\begin{array}{rcl} \theta^\omega\colon X_i & \mapsto & \mathrm{rec}_X\, \theta \\ \theta^*\colon X_i & \mapsto & \theta^i(X) \ . \end{array}$$

Let $n \geq m \geq 0$ and let $B \in \mathbf{L}$ be a term such that $X_i \in fv(B)$ implies $i \geq n$. For every sequence of transitions

$$B[\theta^\omega] \stackrel{\alpha_1}{\longrightarrow} \cdots \stackrel{\alpha_m}{\longrightarrow} C$$

there is a term C' for which $X_i \in fv(C')$ implies $i \geq n - m$, such that $C = C'[\theta^\omega]$ and

$$B[\theta^*] \stackrel{\alpha_1}{\longrightarrow} \cdots \stackrel{\alpha_n}{\longrightarrow} C'[\theta^*] \ .$$

Proof. By induction on m.

$m = 0$**.** In this case, $C = B[\theta^\omega]$ and hence $C' = B$ satisfies the requirements.

$m = 1$**.** This is proved by induction on the derivation of $B[\theta^\omega] \xrightarrow{\alpha_1} C$. Thus, there will be one case for each of the operational rules in Table 2.1. The cases for the first-order operators really follow from the fact that the corresponding rules are all in the SOS format of De Simone [18]. As an example, we show the case of the synchronisation rule.

- $B = B_1 \parallel_A B_2$ such that $B_k[\theta^\omega] \xrightarrow{\alpha_1} C_k$ for $k = 1, 2$ and $C = C_1 \parallel_A C_2$. By the induction hypothesis, for $k = 1, 2$ there are transitions

$$B_k[\theta^*] \xrightarrow{\alpha_1} C'_k[\theta^*]$$

such that $C_k = C'_k[\theta^\omega]$ for $k = 1, 2$. It follows that $C' = C'_1 \parallel_A C'_2$ satisfies the criteria, since clearly

$$B[\theta^*] = B_1[\theta^*] \parallel_A B_2[\theta^*] \xrightarrow{\alpha_1} C'_1[\theta^*] \parallel_A C'_2[\theta^*] = C'[\theta^*] \ .$$

Now we consider the most interesting case, namely the recursion rule. Note that, for the entire term $B[\theta^\omega]$ to be of the form $\mathrm{rec}_Y\, \eta$, there are two sub-cases: B may equal one of the X_i, or B may itself be of the form $\mathrm{rec}_Y\, \eta$.

- $B = X_j$ for some $X \in dom(\theta)$ and $j \geq n \geq 1$; hence $B[\theta^\omega] = \theta^\omega(X_j) = \mathrm{rec}_X\, \theta$. Due to guardedness of θ, by Proposition 2.4 the premise of the rule can be given as $\theta(X) \xrightarrow{\alpha} D$ such that $C = D[\mathrm{rec}_X\, \theta / X]_{X \in dom(\theta)}$. By the definition of θ^j and Proposition 2.3, it follows that

$$B[\theta^*] = \theta^j(X) = \theta(X)[\theta^{j-1}] \xrightarrow{\alpha_1} D[\theta^{j-1}] \ .$$

Then $C' = D[X_{j-1}/X]_{X \in fv(D)}$ satisfies the requirements of the lemma, due to $C'[\theta^*] = D[\theta^{j-1}(X)/X]_{X \in fv(D)} = D[\theta^{j-1}]$.

- $B = \mathrm{rec}_Y\, \eta$. W.l.o.g. assume $dom(\theta^\omega) \cap dom(\eta) = \emptyset$. Since θ^ω and θ^* are closed and $fv(B) \cap dom(\eta) = \emptyset$, it follows that

$$\begin{aligned} B[\theta^\omega] &= \mathrm{rec}_Y(\eta[\theta^\omega]) \\ B[\theta^*] &= \mathrm{rec}_Y(\eta[\theta^*]) \ . \end{aligned}$$

Since η is guarded (by default assumption), due to Proposition 2.4 we can write the premise of the recursion rule as

$$\eta(Y)[\theta^\omega] \xrightarrow{\alpha_1} C \ .$$

By the induction hypothesis, it follows that there is a C' such that $X_i \in fv(C')$ implies $i \geq n - 1$, and

$$\eta(Y)[\theta^*] \xrightarrow{\alpha_1} C'[\theta^*]$$

and $C'[\theta^\omega] = C$. But then also

$$B[\theta^*] = \mathrm{rec}_Y(\eta[\theta^*]) \xrightarrow{\alpha_1} C'[\theta^*]$$

and hence C' satisfies the requirements of the lemma.

$m > 1$**.** This follows immediately from putting together the case of $m = 1$ and the induction hypothesis. Assume

$$B[\theta^\omega] \xrightarrow{\alpha_1} D \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_m} C \ ;$$

then by the case for $m = 1$ we know that there is a term D' for which $X_i \in fv(D')$ implies $i \geq n-1$ such that $D = D'[\theta^\omega]$ and

$$B[\theta^*] \xrightarrow{\alpha_1} D'[\theta^*] \ ,$$

from which, by the induction hypothesis, we can deduce the existence of term C' for which $X_i \in fv(C')$ implies $i \geq (n-1)-(m-1)$ and hence $i \geq n-m$ such that $C = C'[\theta^\omega]$ and

$$D'[\theta^*] \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_m} C'[\theta^*] \ .$$

□

Proposition 4.29 *Let θ satisfy $fv(\theta) \subseteq dom(\theta)$, and let $X \in dom(\theta)$. For every sequence of transitions*

$$\mathrm{rec}_X\,\theta \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} B$$

there is a sequence of transitions

$$\theta^n(X) \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} B'$$

such that $\mathcal{L}(B) = \mathcal{L}(B')$.

Proof. This is a consequence of Lemma 4.28, as follows: Define the term B in Lemma 4.28 to be X_n; then $X_n[\theta^\omega] = \mathrm{rec}_X\,\theta$ and $X_n[\theta^*] = \theta^n(X)$. Let C' be the term whose existence is guaranteed in the lemma, such that $B = C'[\theta^\omega]$ and $X_n[\theta^*] \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} C'[\theta^*]$. One can prove $\mathcal{L}(\theta^i(X)) = \mathcal{L}(\mathrm{rec}_X\,\theta)$ for all $X \in dom(\theta)$ and $i \geq 0$ (by induction, using Proposition 4.23 and $\mathcal{L}(\mathrm{rec}_X\,\theta) = \mathcal{L}(\theta(X)[\mathrm{rec}_X\,\theta]_{X \in dom(\theta)}$ — cf. Proposition 2.9); hence it follows from Proposition 4.23 that

$$\mathcal{L}(C'[\theta^\omega]) = \mathcal{L}[\![C']\!]_{\mathcal{L}[\![\theta^\omega]\!]} = \mathcal{L}[\![C']\!]_{\mathcal{L}[\![\theta^*]\!]} = \mathcal{L}[\![C'[\theta^*]]\!] \ .$$

It follows that the proof obligation is fulfilled if we take $B' = C'[\theta^*]$. □

Proposition 4.30 *Let $\theta\colon \mathbf{X} \rightharpoonup \mathbf{L}$ be such that $fv(\theta) \subseteq dom(\theta)$; then $\mathcal{O}(\mathrm{rec}_X\,\theta) \sqsubseteq_{\mathbf{M}} \bigsqcup_i \mathcal{O}(\theta^i(X))$ for all $X \in dom(\theta)$.*

Proof. We show (1) $\mathcal{F}^+(\mathrm{rec}_X\,\theta) \subseteq \bigcup_i \mathcal{F}^+(\theta^i(X))$ and (2) $\mathrm{rec}_X\,\theta$ **stb** if $\bigwedge_i \theta^i(X)$ **stb**.

1. Assume $\mathrm{rec}_X\,\theta \stackrel{w}{\Longrightarrow} B$ and $V \cap \mathcal{L}(B) = \emptyset$. It follows that there is a sequence of transitions $\mathrm{rec}_X\,\theta \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} B$ such that $w = \pi_{\mathbf{A}}(\alpha_1 \cdots \alpha_n)$ (in words, w is the concatenation of the non-τ-actions). Due to Proposition 4.29 we have $\theta^n(X) \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} B'$ with $\mathcal{L}(B) = \mathcal{L}(B')$; hence $\theta^n(X) \stackrel{w}{\Longrightarrow} B'$. Since clearly $V \cap \mathcal{L}(B') = \emptyset$ it follows that $(w, V) \in \mathcal{F}(\theta^n(X))$.

2. Assume $\neg\,\mathrm{rec}_X\,\theta$ **stb**. It follows that $\mathrm{rec}_X\,\theta \xrightarrow{\tau} B$ for some B. Due to Proposition 4.29 it follows that $\theta^1(X) \xrightarrow{\tau} B'$ for some B', and hence $\neg\theta^1(X)$ **stb**.

□

Proof of Theorem 4.25.1. By induction on the structure of B. For first-order operators, the proof obligation follows immediately from a comparison of the equalities in Table 4.2 with the definitions in Tables 4.3 and 4.4. The only interesting cases are $B = X$ and $B = \mathrm{rec}_X\,\theta$.

$B = X$**.** This follows from the following sequence of equalities:

$$[\![B]\!]_{\mathcal{O}(\sigma)} = \mathcal{O}(\sigma)(X) = \mathcal{O}(\sigma(X)) = \mathcal{O}(B[\sigma]) \ .$$

$B = \mathrm{rec}_X\, \theta$**.** W.l.o.g. assume $dom(\sigma) \cap dom(\theta) = \emptyset$. We show $[\![B]\!]_{\mathcal{O}(\sigma)} \simeq_{\mathbf{M}} \mathcal{O}(B[\sigma])$ by splitting the equivalence in two parts.

$\sqsubseteq_{\mathbf{M}}$ Consider the function Ψ defined by $\Psi : \Omega \mapsto [\![\theta]\!]_{\mathcal{O}(\sigma) \lhd \Omega}$ for all $\Omega \colon \mathbf{X} \rightharpoonup \mathbf{M}$ with $dom(\Omega) \supseteq fv(\theta) \setminus dom(\sigma)$. Then the following sequence of equalities holds (where $\mathrm{rec}\_\,\theta$ is shorthand for $[\mathrm{rec}_X\, \theta / X]_{X \in dom(\theta)}$):

$$\begin{array}{rcl} \Psi(\mathcal{O}(\mathrm{rec}\_\,\theta[\sigma])) & = & [\![\theta]\!]_{\mathcal{O}(\sigma) \lhd \mathcal{O}(\mathrm{rec}\_\,\theta[\sigma])} \\ & = & \mathcal{O}(\theta[\sigma \lhd \mathrm{rec}\_\,\theta[\sigma]]) \\ & \simeq_{\mathbf{M}} & \mathcal{O}(\theta[\sigma][\mathrm{rec}\_\,\theta[\sigma]]) \\ & \simeq_{\mathbf{M}} & \mathcal{O}(\mathrm{rec}\_\,\theta[\sigma]) \end{array}$$

The first step is by definition of Ψ, the second by the induction hypothesis, and the third by Proposition 2.9 (which proves this equivalence up to $\sim$) in combination with the observation that $\sim \subseteq \simeq_{\mathbf{M}}$ due to Proposition 3.6, Theorem 4.12 and the fact that $\sim$ is stability-preserving.

This proves that $\mathcal{O}(\mathrm{rec}\_\,\theta[\sigma])$ is a fixpoint of Ψ. But then $[\![\mathrm{rec}_X\, \theta]\!]_{\mathcal{O}(\sigma)} \sqsubseteq_{\mathbf{M}} \mathcal{O}(\mathrm{rec}_X\, \theta[\sigma])$ for all $X \in dom(\theta)$ due to the fact that $[\![\mathrm{rec}\_\,\theta]\!]_{\mathcal{O}(\sigma)} = \mu[\![\theta]\!]_{\mathcal{O}(\sigma)}$ is a *least* fixpoint of Ψ (Proposition 4.21).

$\sqsupseteq_{\mathbf{M}}$ This follows from the following sequence of (in)equalities:

$$[\![\mathrm{rec}_X\, \theta]\!]_{\mathcal{O}(\sigma)} = \bigsqcup_i [\![\theta]\!]^i_{\mathcal{O}(\sigma)}(X) \simeq_{\mathbf{M}} \bigsqcup_i \mathcal{O}(\theta[\sigma]^i(X)) \sqsupseteq_{\mathbf{M}} \mathcal{O}(\mathrm{rec}_X\, \theta[\sigma])$$

where the first step is by definition of $[\![\mathrm{rec}_X\, \theta]\!]$, the second by Lemma 4.27 and the third by Proposition 4.30.

□

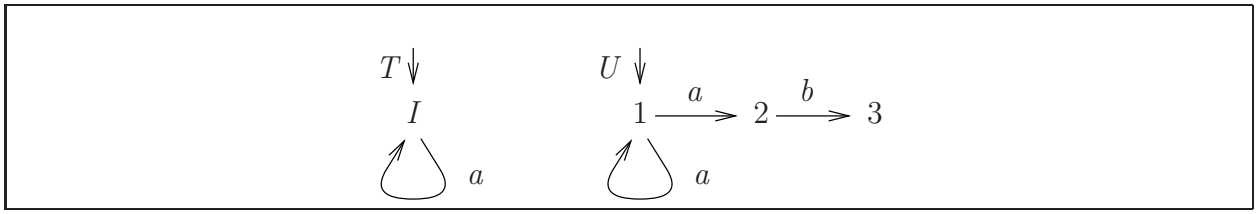


Figure 5.1: Example transition systems for the decision algorithms

5 Decidability and complexity

In this section, we will show that $\sqsubseteq_{\mathbf{shd}}$ is decidable, with linear exponential time complexity. Since $\sqsubseteq_{\mathcal{L}}^{-1}$ and $\sqsubseteq_{\mathbf{stb}}$ are known or obvious to be decidable, with a complexity that is no worse, it follows that $\sqsubseteq_{\mathbf{shd}}^{c}$ is decidable in linear exponential time.

The relation we eventually want to check is $T \sqsubseteq_{\mathcal{F}^+} U$ for some finite transition systems T and U; that is,

$$\forall\, (v, V) \in \mathcal{F}^+(T) \colon \exists\, (w, W) \in \mathcal{F}^+(U) \colon (v, V) \preceq (w, W) \ .$$

As an exercise, we will first show in Section 5.2 how to check $\mathcal{F}^+(T) \subseteq \mathcal{F}^+(U)$. As a running example, we will treat the transition systems T and U shown in Figure 5.1 where neither $\mathcal{F}^+(T) \subseteq \mathcal{F}^+(U)$ nor $T \sqsubseteq_{\mathcal{F}^+} U$ is true.

5.1 Preliminary concepts and constructions

Throughout this section, we assume that two finite transition systems T and U with initial states p_0 and q_0 are given such that $\mathcal{L}(T) \subseteq \mathcal{L}(U)$, which is known to be decidable. We denote the transition relations with $\rightarrow_T$ and $\Rightarrow_T$ etc., and by abuse of notation we use $\mathbf{A}$ for the finite set of visible actions occurring in T or U; hence, in the running example $\mathbf{A} = \{a, b\}$.

We can view each finite transition system as a finite automaton where all states are final. For an automaton A with some state s (we write $s \in A$), $\mathcal{L}_A(s)$ denotes the language of the automaton if we change the initial state to s; observe that the arc-label τ corresponds to the empty word in automata theory. We call a state *productive*, if it lies on a path from the initial state to a final state, i.e. if it is used by the automaton when accepting a word.

As a first step, we extend the automaton T to an *automaton of automata* AA by adding a family of deterministic automata A_p, $p \in T$, such that for each $p \in T$ the language of A_p is the set $\mathbf{A}^* \setminus \mathcal{L}_T(p)$ of traces that T cannot perform from p. The following holds:

$$(v, V) \in \mathcal{F}^+(T) \text{ if and only if } \exists p_0 \stackrel{v}{\Rightarrow}_{AA} p \colon V \subseteq \mathcal{L}(A_p) \ .$$

Thus, the automata A_p represent some tree failures $(v, V) \in \mathcal{F}^+(T)$ in the sense that there is a $p \in T$ with $p_0 \stackrel{v}{\Rightarrow}_{AA} p$ and $V = \mathcal{L}(A_p)$; in particular, they represent all maximal tree failures, i.e. all those $(v, V) \in \mathcal{F}^+(T)$ with maximal V. Since in a finite transition system T there exists for each $(v, W) \in \mathcal{F}^+(T)$ a maximal $(v, V) \in \mathcal{F}^+(T)$ with $W \subseteq V$, maximal tree failures of $\mathcal{F}^+(T)$ are all we have to consider when checking $\mathcal{F}^+(T) \subseteq \mathcal{F}^+(U)$ or $\mathcal{F}^+(U) \subseteq \mathcal{F}^+(T)$.

Similarly, we construct an automaton of automata for U, but this time, we additionally make U deterministic more or less by the usual power set construction. This results in a *deterministic automaton of automata* BB, which is a deterministic automaton extended with a family BB_Q, $Q \in BB$, where each BB_Q is a set of automata: for each state Q (being a set of states of U) BB_Q consists of deterministic automata B_q, $q \in Q$, with $\mathcal{L}(B_q) = \mathbf{A}^* \setminus \mathcal{L}_U(q)$.

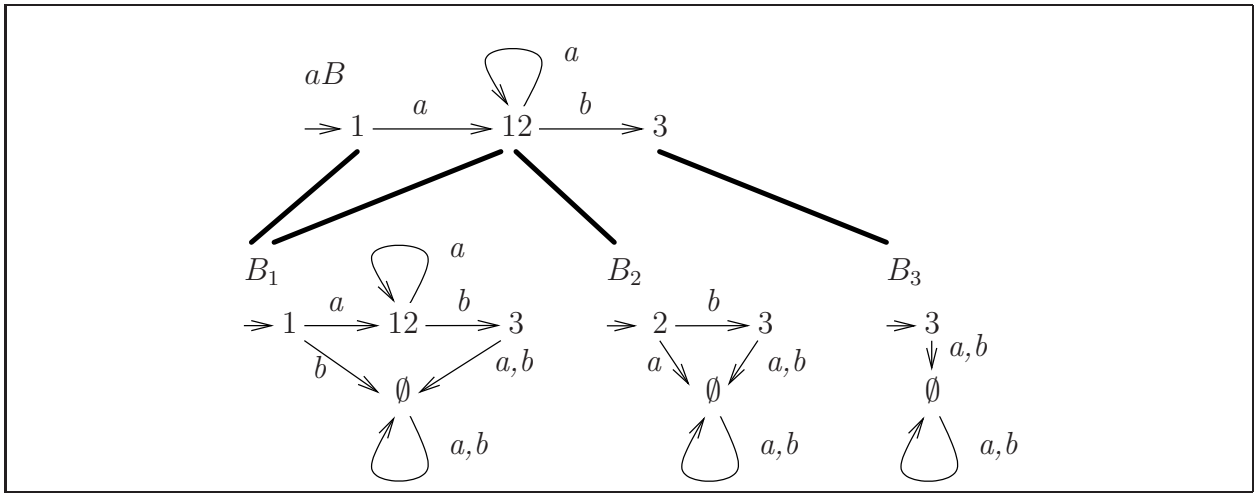


Figure 5.2: BB for the example transition system U in Figure 5.1

More in detail, the automaton part of BB is defined as follows. The initial state of BB is $Q_0 = \{q \mid q_0 \xrightarrow{\tau^*}_U q\}$; the transition relation is defined by $Q \xrightarrow{a}_{BB} Q'$ if $Q' = \{q' \mid \exists q \in Q\colon q \xrightarrow{a}_U \xrightarrow{\tau^*}_U q'\}$. We restrict BB to the nonempty states reachable from Q_0 and let each state of BB be final.[6] As a consequence, all states of BB are productive and $\mathcal{L}(BB) = \mathcal{L}(U)$.

This way, $Q_0 \overset{v}{\Longrightarrow}_{BB} Q$ iff $Q = \{q \mid q_0 \overset{v}{\Longrightarrow}_U q\}$ for all $v \in \mathbf{A}^*$ and

$$(v, V) \in \mathcal{F}^+(U) \text{ if and only if } \exists Q_0 \overset{v}{\Longrightarrow}_{BB} Q,\ B \in BB_Q\colon V \subseteq \mathcal{L}(B)\ .$$

Example 5.1 *Figure 5.2 shows BB for U as in Figure 5.1, where all states are final, and the associated automata, where only $\emptyset$ is final. Each state Q of BB is connected with thick lines to the automata in BB_Q. (We write a state $\{1,2\}$ as 12 etc.) Note that these automata are also obtained from a power set construction and complementation of final states, and that one only has to construct one automaton for each state q of U.*

5.2 Decision for $\mathcal{F}^+$-inclusion

As an exercise, we will now show how to check $\mathcal{F}^+(T) \subseteq \mathcal{F}^+(U)$. For this, from AA and BB, we construct the following (partial) product automaton S, which can also be seen as the minimal simulation from AA to BB (where a simulation is a relation between the states of AA and that of BB).[7]

- $(p_0, Q_0) \in S$ is the initial state of S and all states are final.
- If $(p, Q) \in S$, $a \in \mathbf{A}$ and $p \xrightarrow{a}_{AA} p'$, then by language inclusion and definition of BB, there is a unique $Q' \in BB$ such that $Q \xrightarrow{a}_{BB} Q'$; we add (p', Q') and the transition $(p, Q) \xrightarrow{a} (p', Q')$ to S.
- If $(p, Q) \in S$ and $p \xrightarrow{\tau}_{AA} p'$, then we add (p', Q) and the transition $(p, Q) \xrightarrow{\tau} (p', Q)$ to S. (Observe that BB has no τ-arcs.)

[6] Note that BB is deterministic in the sense that there are no τ-labelled arcs and for each state Q and each $a \in \mathbf{A}$ there is *at most* one outgoing a-labelled arc.

[7] In fact, initial state, final states and arcs of S are technically irrelevant, but we consider them as intuitively helpful.

The algorithm for checking $\mathcal{F}^+(T) \subseteq \mathcal{F}^+(U)$ now is based on the following principle. For every sequence $p_0 \stackrel{v}{\Longrightarrow}_{AA} p$, $V = \mathcal{L}(A_p)$ is a maximal refusal set for v in T (i.e., a maximal set for which $(v, V) \in \mathcal{F}^+(T)$); moreover, all refusal sets are a subset of some maximal refusal set that is encoded this way. Since BB is deterministic, v also determines a unique state Q with $Q_0 \stackrel{v}{\Longrightarrow}_{BB} Q$; by construction we have $(p, Q) \in S$. The automata in BB_Q likewise represent all maximal refusal sets for v in U. Hence, we simply have to check for each $(p, Q) \in S$ whether there is some $B \in BB_Q$ with $\mathcal{L}(A_p) \subseteq \mathcal{L}(B)$.

Example 5.2 *Figure 5.3 shows S for T (i.e. the corresponding AA) and BB as in Figures 5.1 and 5.2, as well as the only associated automaton A_I of AA, where $\emptyset$ is the only final state. Since $(I, 1) \in S$ and $ab \in \mathcal{L}(A_I) \setminus \mathcal{L}(B_1)$, we conclude that $\mathcal{F}^+(T) \subseteq \mathcal{F}^+(U)$ fails.*

5.3 Decision for $\sqsubseteq_{\mathcal{F}^+}$

Checking $\sqsubseteq_{\mathcal{F}^+}$ entails checking whether for all $(v, V) \in \mathcal{F}^+(T)$ with $V \neq \emptyset$ we have $(v\,u, u^{-1}V) \in \mathcal{F}^+(U)$ for some $u \in {\downarrow}V$. (Recall that, by language inclusion, we do not have to check pairs $(v, \emptyset)$.) As above, we construct AA, BB and S. This time, we have to check for each $(p, Q) \in S$ and each $\emptyset \neq V \subseteq \mathcal{L}(A_p)$ that

$$\exists u \in {\downarrow}V,\ Q' \in BB,\ B \in BB_{Q'} \colon Q \stackrel{u}{\Longrightarrow}_{BB} Q' \text{ and } u^{-1}V \subseteq \mathcal{L}(B)\ . \tag{4}$$

Let us fix (p, Q); we now show how to check (4) for all $\emptyset \neq V \subseteq \mathcal{L}(A_p)$. This means that we have to compare runs of A_p (u in (4)) with runs of BB. To do this, we construct another (partial) product automaton P, similar to the one above, but this time between the automaton A_p (whose initial state we also denote by p) and BB where the initial state is changed to Q. Another difference with the case above is that, this time, we do not necessarily have $\mathcal{L}(A_p) \subseteq \mathcal{L}(BB)$ — i.e., BB might not be able to simulate A_p — but still we want to represent all of $\mathcal{L}(A_p)$ in order to check the inclusion in (4). Therefore, P is constructed as follows (here $*$ is a dummy element, not appearing anywhere else):

- $(p, Q) \in P$ is the initial state;
- if $(p', Q') \in P$ and $p' \xrightarrow{a}_{A_p} p''$
  - and $Q' \xrightarrow{a}_{BB} Q''$, we add the state (p'', Q'') and the transition $(p', Q') \xrightarrow{a} (p'', Q'')$ to P;
  - and $Q' \not\xrightarrow{a}_{BB}$ (in particular, if $Q' = *$), we add $(p'', *)$ and the transition $(p', Q') \xrightarrow{a} (p'', *)$ to P;
- (p', Q') is final iff p' is.

Since A_p and BB are deterministic, P is also deterministic, and we have $\mathcal{L}(P) = \mathcal{L}(A_p)$ by construction. We will call R a *productive sub-automaton* of P, if R is obtained from P by restricting all components (in particular also the final states) to a subset M of the state set such that each state of R is productive (in R).[8] We will show that (4) is satisfied for all $\emptyset \neq V \subseteq \mathcal{L}(P)$ if and only if for each productive sub-automaton R of P

$$\exists (p', Q') \in R,\ Q' \in BB,\ B \in BB_{Q'} \colon \mathcal{L}_R(p', Q') \subseteq \mathcal{L}(B)\ . \tag{5}$$

Since the latter clearly is decidable, it then follows that that $\sqsubseteq_{\mathcal{F}^+}$ is decidable. (Note that $Q' \in BB$ in (5) is equivalent to $Q' \neq *$.)

[8] Recall that a state of an automaton is called productive if it lies on a path from the initial to some final state.

So assume (4) is satisfied for all $\emptyset \neq V \subseteq \mathcal{L}(P)$. If R is a productive sub-automaton, then $\emptyset \neq \mathcal{L}(R) \subseteq \mathcal{L}(P)$. Hence (due to (4)), $\exists u \in \downarrow\mathcal{L}(R)$, $Q' \in BB$, $B \in BB_{Q'}$: $Q \stackrel{u}{\Longrightarrow}_{BB} Q'$ and $u^{-1}\mathcal{L}(R) \subseteq \mathcal{L}(B)$. Then $(p, Q) \stackrel{u}{\Longrightarrow}_R (p', Q')$ for some p'; since R is deterministic, (p', Q') is uniquely determined by u, and therefore $u^{-1}\mathcal{L}(R) = \mathcal{L}_R(p', Q')$. Thus, (p', Q') and B are the state and the automaton whose existence is asserted in (5).

Vice versa, assume that (5) holds for each productive sub-automaton R and take some $\emptyset \neq V \subseteq \mathcal{L}(P)$. The set of states that are needed in P to accept the words of V defines a productive sub-automaton R with $V \subseteq \mathcal{L}(R)$. Take $(p', Q') \in R$ and $B \in BB_{Q'}$ that satisfy (5). Then there is some $u \in \downarrow V$ with $(p, Q) \stackrel{u}{\Longrightarrow}_P (p', Q')$ by choice of R, and $Q \stackrel{u}{\Longrightarrow}_{BB} Q'$ by construction of P and since $Q' \in BB$. Now $u^{-1}V \subseteq u^{-1}\mathcal{L}(R) = \mathcal{L}_R(p', Q')$ by determinism of R, and we conclude that $u^{-1}V \subseteq \mathcal{L}(B)$.

Therefore we have shown:

Theorem 5.3 *$\sqsubseteq_{\mathcal{F}^+}$, i.e. $\sqsubseteq_{\mathbf{shd}}$, is decidable.*

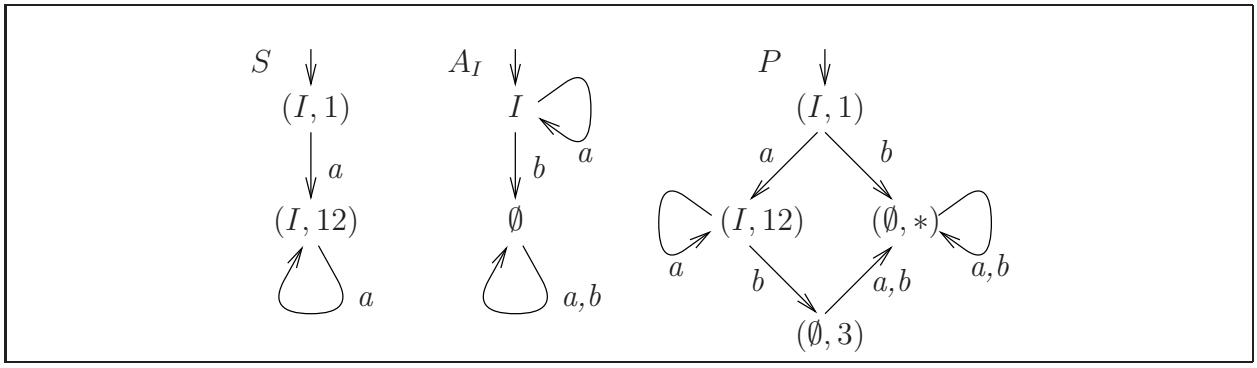


Figure 5.3: S, P and the only A_p for the example transition systems

Example 5.4 *Figure 5.3 shows S for T (i.e., its corresponding AA) and BB as in Figures 5.1 and 5.2, as well as the only associated automaton A_I of AA, where $\emptyset$ is the only final state. Moreover, for the case $(p, Q) = (I, 1)$ it shows the product automaton P, where $(\emptyset, 3)$ and $(\emptyset, *)$ are final states. If we omit $(\emptyset, *)$ from P, we get a productive sub-automaton R for which (5) fails:*

- $(p', Q') = (I, 1)$*:* $ab \in \mathcal{L}_R(I, 1)$, $ab \notin \mathcal{L}(B_1)$*;*
- $(p', Q') = (I, 12)$*:* $b, ab \in \mathcal{L}_R(I, 12)$, $ab \notin \mathcal{L}(B_1)$, $b \notin \mathcal{L}(B_2)$*;*
- $(p', Q') = (\emptyset, 3)$*:* $\varepsilon \in \mathcal{L}_R(\emptyset, 3)$, $\varepsilon \notin \mathcal{L}(B_3)$.

Since we reach $(I, 1)$ in S by ε and $\mathcal{L}(R) = a^+b$, the fact that (5) fails corresponds to $(\varepsilon, a^+b) \in \mathcal{F}^+(T)$ which is not $\preceq$-dominated by any element of $\mathcal{F}^+(U)$; hence $T \not\sqsubseteq_{\mathcal{F}^+} U$.

5.4 Complexity

The above decision algorithm for $\sqsubseteq_{\mathcal{F}^+}$ builds BB and P which could have in the order of 2^m states, where m is the number of states of U; then, P could have in the order of 2^{2^m} productive sub-automata. It follows that one would expect the algorithm to take at least double exponential

time in the worst case. We will now refine the algorithm and show that $\sqsubseteq_{\mathcal{F}^+}$ can be decided in exponential, even linear exponential time.

Consider P as productive sub-automaton of itself, and consider respective $(p', Q') \in P$ and $B \in BB_{Q'}$ that make P satisfy (5). If some productive sub-automaton R contains (p', Q'), then clearly $\mathcal{L}_R(p', Q') \subseteq \mathcal{L}_P(p', Q') \subseteq \mathcal{L}(B)$. Thus, once we have found (p', Q'), we have to check (5) further only for productive sub-automata R' of P', where P' is obtained from P by removing (p', Q') and all states that then are not on a path from (p, Q) to a final state, i.e. are not productive anymore. Now P' can be treated in the same way, and all in all we only have to check (5) for a number of productive sub-automata bounded by the number of states of P. We arrive at the following result, whose proof can be found in Appendix B (Page 69).

Theorem 5.5 *For a fixed finite alphabet size, $T \sqsubseteq_{\mathbf{shd}} U$ and $T \sqsubseteq^c_{\mathbf{shd}} U$ can be decided in time linear exponential in the numbers of states of T and U.*

Actually, our algorithm can be further simplified: one can show that the states $(p, *)$ of P are not needed. Since this does not improve the complexity in the worst case, we do not go into details here.

B	$=$	$\sum\{B\}$	(6)
$\sum_{i\in I}\sum_{j\in J} B_{ij}$	$=$	$\sum\{B_{ij} \mid i\in I, j\in J\}$	(7)
$B \parallel_A C$	$=$	$\sum\{\beta_i;(B_i \parallel_A C) \mid \beta_i \notin A\}$	(8)
		$+\sum\{\gamma_k;(B \parallel_A C_k) \mid \gamma_k \notin A\}$	
		$+\sum\{\beta_i;(B_i \parallel_A C_k) \mid \beta_i = \gamma_k \in A\}$	
$B[\varphi]$	$=$	$\sum\{\varphi(\beta_i);(B_i[\varphi]) \mid i\in I\}$	(9)
B/A	$=$	$\sum\{\tau;(B_i/A) \mid \beta_i \in A\}$	(10)
		$+\sum\{\beta_i;(B_i/A) \mid \beta_i \notin A\}$	
$\mathrm{rec}_X\,\theta$	$=$	$\theta(X)[\mathrm{rec}_\theta]$	(11)
$\alpha;\tau;B$	$=$	$\alpha;B$	(12)
$B+\tau;B$	$=$	$\tau;B$	(13)
$\alpha;(B+\tau;C)+\alpha;C$	$=$	$\alpha;(B+\tau;C)$	(14)

Table 6.1: Bisimulation axioms, where $B=\sum_{i\in I}\beta_i;B_i$ and $C=\sum_{k\in K}\gamma_k;C_k$.

6 Proof principles

We discuss a number of general methods to prove that the $\sqsubseteq^c_{\mathbf{shd}}$-preorder holds between a given pair of systems, avoiding the direct use of the (costly) denotational characterisation as much as possible. In the next section, we use these methods in a number of examples of system specifications and their implementations on the basis of should-testing.

6.1 The bisimulation inheritance

Since observation congruence is stronger than all our testing pre-orders (see Proposition 3.6), all known methods to prove $\simeq_{\mathbf{bis}}$ are valid for proving $\simeq^c_{\mathbf{shd}}$. This is an advantage because, as mentioned before, the proof techniques for observation congruence are relatively simple; hence, if it holds, it is cheaper to show $\simeq_{\mathbf{bis}}$ than to try to show $\simeq^c_{\mathbf{shd}}$ directly. Of course, it may be that observation congruence fails to hold between two given $\sqsubseteq^c_{\mathbf{shd}}$-related systems, in which case the inherited proof techniques obviously cannot work. If this is so, then one can still try methods more directly tuned to the testing notion one is actually interested in. (This point is made very forcefully by Valmari [34] in the context of transition system reduction.)

Of the proof techniques available for observation congruence we mention two: *constructing a bisimulation relation* and *applying the equational theory*. The details of bisimulation relations are omitted here; see [28] for an exposition. With respect to the equational theory, we recall the axioms in Table 6.1, adapted from Milner to our setting.

Axioms (8)–(10) explain, respectively, synchronisation, renaming and hiding in terms of action prefix and choice. In fact, using (6)–(10) one may rewrite every finite term into a term of the form $\sum_{i\in I}\alpha_i;B_i$ where the B_i are again of this form. Axiom (11), on the other hand, states that we may always unfold fixpoint terms.

6.2 The testing theory

We do not have a complete equational theory of $\sqsubseteq^c_{\mathbf{shd}}$. However, there are a number of axioms that this relation satisfies beyond those of observation congruence. For one thing, although $\sqsubseteq^c_{\mathbf{shd}}$ is incomparable to the standard must-testing of De Nicola and Hennessy [17] (see Proposition 3.6),

$\alpha;B+\alpha;C$	$=$	$\alpha;(\tau;B+\tau;C)$	(15)
$\tau;(B+C)$	$\sqsubseteq$	$B+\tau;C$	(16)
$\alpha;B+\tau;(\alpha;C+D)$	$=$	$\tau;(\alpha;B+\alpha;C+D)$	(17)
B	$\sqsubseteq$	$\tau;B$	(18)
if $B \sqsubseteq C$	then	$C \sqsubseteq \tau;B+\tau;C$	(19)
if $\theta[\sigma]=\sigma$	then	$\mathrm{rec}\_\,\theta \sqsubseteq \sigma$	(20)

Table 6.2: Should-testing axioms and rules.

most of their axioms dealing with nondeterminism do hold in our setting as well. Furthermore, the fact that recursion builds smallest fixpoints (Proposition 4.21), can be lifted to the language and transferred to $\sqsubseteq^c_{\mathbf{shd}}$ (Theorem 4.25). Table 6.2 contains the resulting axioms and rules. Rule (19) comes in the place of $B \sqsubseteq \tau;B+\tau;C$, which is an important axiom of must-testing that is *not* satisfied by $\sqsubseteq^c_{\mathbf{shd}}$ (it contradicts the language equality implicit in $\sqsubseteq^c_{\mathbf{shd}}$). Rule (20) is a weaker version of the *recursive specification principle* that we have shown in Figure 3.5 not to hold for $\simeq^c_{\mathbf{shd}}$: even though a recursive set of equations θ does not generate a unique solution, every solution is $\sqsubseteq^c_{\mathbf{shd}}$-related to the fixpoint generated by $\mathrm{rec}\,\theta$.

Theorem 6.1 *$\sqsubseteq^c_{\mathbf{shd}}$ and $\simeq^c_{\mathbf{shd}}$ satisfy the axioms and rules in Table 6.2.*

Proof. Clearly, language equality and reverse implication for stability are satisfied in all cases. In the cases (15), (16), (18) and (19), we even have equality or inclusion of $\mathcal{F}^+$-semantics (in (19) without any assumption on B and C) and are done by Proposition 4.7; as an example, we treat (16).

If $(w,V) \in \mathcal{F}^+(\tau;(B+C))$ arises from $\tau;(B+C) \stackrel{w}{\Longrightarrow} \tau;(B+C)$ or $\tau;(B+C) \stackrel{w}{\Longrightarrow} B+C$, then $w=\varepsilon$ and $V \cap \mathcal{L}(\tau;(B+C)) = V \cap \mathcal{L}(B+C) = \emptyset$; thus, $(w,V) \in \mathcal{F}^+(B+\tau;C)$ since $\mathcal{L}(\tau;(B+C)) = \mathcal{L}(B+\tau;C)$. If it arises from some other $\tau;(B+C) \stackrel{w}{\Longrightarrow} D$, then $B \stackrel{w}{\Longrightarrow} D$ or $C \stackrel{w}{\Longrightarrow} D$, hence also $B+\tau;C \stackrel{w}{\Longrightarrow} D$; therefore $(w,V) \in \mathcal{F}^+(B+\tau;C)$ in this case, too.

For (17), we have $\mathcal{F}^+(\tau;(\alpha;B+\alpha;C+D)) \subseteq \mathcal{F}^+(\alpha;B+\tau;(\alpha;C+D))$ with a similar argument, and we almost have the other inclusion as well. The only exception is $(w,V) \in \mathcal{F}^+(\alpha;B+\tau;(\alpha;C+D))$ arising from $\alpha;B+\tau;(\alpha;C+D) \xrightarrow{\tau} \alpha;C+D$, i.e. $w=\varepsilon$ and $\mathcal{L}(\alpha;C+D) \cap V = \emptyset$. If $\mathcal{L}(\alpha;B) \cap V = \emptyset$, we have $(w,V) \in \mathcal{F}^+(\tau;(\alpha;B+\alpha;C+D))$; otherwise $\alpha \in \downarrow V$ and $(\alpha,\alpha^{-1}V) \in \mathcal{F}^+(\tau;(\alpha;B+\alpha;C+D))$ by $\tau;(\alpha;B+\alpha;C+D) \stackrel{\alpha}{\Longrightarrow} C$ since $\mathcal{L}(C) \cap \alpha^{-1}V \subseteq \alpha^{-1}(\mathcal{L}(\alpha;C+D)) \cap \alpha^{-1}V = \emptyset$. □

Note that the axioms in Table 6.2 (together with the idempotence of choice, which is derivable in our setting) imply the τ-laws of observation congruence in Table 6.1.

In addition to these axioms, we recall from Section 3.3 that $\simeq^c_{\mathbf{shd}}$ also satisfies the weakened fairness rule KFAR$^-$ (see Page 22). For instance, KFAR$^-$ in combination with Axiom (12) can be used to show that τ-loops at non-initial states can be ignored: if a is an arbitrary action not occurring in B then

$$\alpha;(\mathrm{rec}\,X.\,\tau;X+B) \simeq^c_{\mathbf{shd}} \alpha;((\mathrm{rec}\,X.\,a;X+B)/a) \simeq^c_{\mathbf{shd}} \alpha;\tau;(B/a) \simeq^c_{\mathbf{shd}} \alpha;B$$

where the second step is by KFAR$^-$. It may be worthwile noting that, in combination with Rule (20), KFAR$^-$ can actually be adapted somewhat to approach the original KFAR (1) — where, with respect to Rule (20), $\theta = [a_i;X_{i+1}/X_i]_{i\in\mathbb{N}_n}$ and $\sigma = [X_i/B_i]_{i\in\mathbb{N}_n}$:

$$\frac{B_i = a_i;B_{i+1}+C_i \quad a_i \in A}{B_i/A \sqsupseteq \tau;\sum_{i\in\mathbb{N}_n}(C_i/A)}\ (i \in \mathbb{N}_n) \tag{21}$$

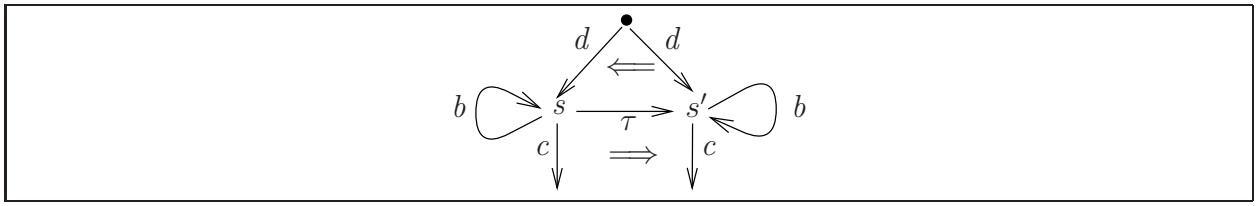


Figure 6.1: preconditions for contraction

6.3 Denotational arguments

As remarked before, the equational theory presented above is not complete. This means that occasionally one may be forced to show that two systems are $\sqsubseteq^c_{\mathbf{shd}}$-related by directly accessing the denotational characterisation. As an example of a property proved in this way, we formulate a *contraction lemma* stating that under certain circumstances, two states of a transition system can be identified. The proof is rather technical and thus deferred to Appendix B (Page 70). An algebraic counterpart of this lemma is difficult to give.

Lemma 6.2 (contraction lemma) *Let $T = \langle S, \rightarrow, q\rangle$ be a transition system with states $s' \neq q$ and s that satisfy the following conditions (see Figure 6.1):*

a) $s \xrightarrow{\alpha} s'$ iff $\alpha = \tau$;

b) for all $\alpha \neq \tau$ we have $s \xrightarrow{\alpha} s$ iff $s' \xrightarrow{\alpha} s'$;

c) if $s \xrightarrow{\alpha} t$ for some $t \in S \setminus \{s, s'\}$, then $s' \xrightarrow{\alpha} t'$ for some $t' \in S \setminus \{s'\}$;

d) if $t \xrightarrow{\alpha} s'$ for some $t \in S \setminus \{s, s'\}$, then $t \xrightarrow{\alpha} s$, too.

Let $U = \langle S', \rightarrow, q\rangle$ be obtained from T by contracting s and s' to s, i.e., by putting $S' = S \setminus \{s'\}$ and replacing s' in the arcs by s. Then $T \simeq^c_{\mathbf{shd}} U$.

Another denotational argument is presented in the following proposition. We call a transition system *deterministic* if for all reachable states s, $s \xrightarrow{\alpha} s_1$ and $s \xrightarrow{\alpha} s_2$ implies $\alpha \neq \tau$ and $s_1 = s_2$.

Proposition 6.3 *If S is deterministic and $I \sqsubseteq^c_{\mathbf{shd}} S$, then $I \simeq_{\mathbf{bis}} S$.*

Proof. The absence of τ-moves implies stability of S and hence of I; thus, it is sufficient to show that $\mathcal{R} = \{(I', S') \mid \exists w\colon I \stackrel{w}{\Longrightarrow} I' \wedge s \stackrel{w}{\Longrightarrow} S'\}$ is a weak bisimulation. Given $(I', S') \in \mathcal{R}$ and a respective w, we have: $I' \xrightarrow{\tau} I''$ implies $(I'', S') \in \mathcal{R}$; $I' \xrightarrow{a} I''$ implies $w\,a \in \mathcal{L}(I) = \mathcal{L}(S)$, and since S' is unique with $S \stackrel{w}{\Longrightarrow} S'$ by determinism, there is some $S' \xrightarrow{a} S''$ with $(I'', S'') \in \mathcal{R}$. Finally, assume $S' \xrightarrow{a} S''$; if $I' \stackrel{a}{\Longrightarrow}$, we are done, and otherwise $(w, \{a\}) \in \mathcal{F}(I) \subseteq \mathcal{F}(S)$; but the latter is a contradiction, since S' is unique with $S \stackrel{w}{\Longrightarrow} S'$. □

As a final "denotational" proof technique, we mention the result from the preceding section that $\sqsubseteq_{\mathbf{shd}}$ and $\sqsubseteq^c_{\mathbf{shd}}$ are decidable for finite-state systems.

6.4 Compositionality

In conjunction with the methods presented above, it is very important to realise that, due to the fact that our notion of implementation is pre-congruent, proofs of correctness can be done in a *compositional manner*. This means that it is not necessary to consider complete systems; rather, one may take a single module and replace that by a "better" one (more fault-tolerant, more deterministic, etc.); the entire system will thereby be improved, and the result of this replacement is a formal implementation of the original system.

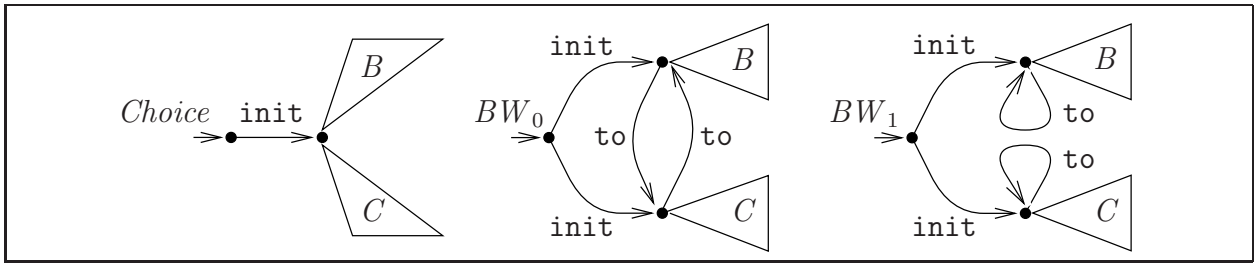


Figure 7.1: Correct and incorrect versions of busy waiting (`to` abbreviates `timeout`).

7 Examples

The purpose of this section is to demonstrate the advantages of **shd**-testing compared to observation congruence, **acc**- and divergence-sensitive must-testing. In this section, we make use of the alternative presentation of recursion mentioned in Section 3, i.e. we give the process environment θ by listing defining equations and use X in place of $\mathrm{rec}_X\,\theta$, leaving θ implicit. In Section 7.3, we also use the **with**-notation.

7.1 External choice as busy-waiting

Our first, simple example concerns the implementation of external choice as busy waiting; see Figure 7.1. In the process $\mathit{Choice} := \mathtt{init}; (B + C)$ we have that, after initialisation, either B or C can be chosen. A *busy-waiting* implementation oscillates between B and C:

$$\begin{aligned} BW_0 &:= \mathtt{init}; \mathit{Wait}_B + \mathtt{init}; \mathit{Wait}_C \\ \mathit{Wait}_B &:= B + \mathtt{timeout}; \mathit{Wait}_C \\ \mathit{Wait}_C &:= C + \mathtt{timeout}; \mathit{Wait}_B \end{aligned}$$

where `timeout` is assumed not to occur in B or C (hence $B/\mathtt{timeout} \simeq_{\mathbf{bis}} B$ and $C/\mathtt{timeout} \simeq_{\mathbf{bis}} C$). From KFAR$^-$ (2), Axiom (12) and idempotence of $+$, it can be derived that $\mathit{Choice} \simeq_{\mathbf{bis}} BW_0/\mathtt{timeout}$; hence this also holds for $\simeq^c_{\mathbf{shd}}$ and $\simeq_{\mathbf{acc}}$ (Proposition 3.6). The systems are not equivalent under De Nicola-Hennessy must-testing, since this is sensitive to the divergence in $BW_0/\mathtt{timeout}$. To show the advantage of $\simeq^c_{\mathbf{shd}}$ over $\simeq_{\mathbf{acc}}$, consider

$$\begin{aligned} BW_1 &:= \mathtt{init}; \mathit{Wait}'_B + \mathtt{init}; \mathit{Wait}'_C \\ \mathit{Wait}'_B &:= B + \mathtt{timeout}; \mathit{Wait}'_B \\ \mathit{Wait}'_C &:= C + \mathtt{timeout}; \mathit{Wait}'_C \ . \end{aligned}$$

BW_1 fails to change between options, so in general $BW_1/\mathtt{timeout}$ is certainly not a correct implementation of *Choice*. Accordingly, $BW_0 \not\simeq^c_{\mathbf{shd}} BW_1$; but acceptance testing suggests that BW_0 and BW_1 can be used interchangeably, since $BW_0 \simeq_{\mathbf{acc}} BW_1$ (any failure of BW_0 can be found in BW_1 and vice versa, by taking the appropriate initial `init`-branch). This, therefore, is an example where the lack of pre-congruence w.r.t. hiding makes acceptance testing unsuitable.

7.2 The alternating bit protocol

As a more extensive example we will use $\simeq^c_{\mathbf{shd}}$ to show the correctness of a version of the alternating-bit protocol. The desired behaviour is that of a one-place buffer:

$$\mathit{Buf} := \mathtt{snd}; \mathtt{rcv}; \mathit{Buf},$$

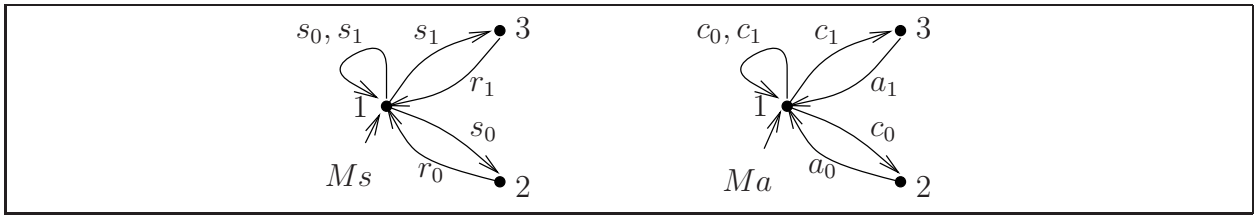


Figure 7.2: Two lossy channels.

where we abstract from the content of the message sent. The implementation is built from the sender Snd_0 and the receiver Rcv, which are connected by two lossy channels Ms and Ma for transmitting messages and acknowledgements. An additional bit is appended to the messages and acknowledgements, so Ms participates in sending with s_i on the one side and in receiving with r_i on the other, while Ma participates in confirming with c_i and in acknowledging with a_i $(i = 0, 1)$. The channels are given by

$$\begin{aligned} Ms &:= s_0; r_0; Ms + s_0; Ms + s_1; r_1; Ms + s_1; Ms \\ Ma &:= c_0; a_0; Ma + c_0; Ma + c_1; a_1; Ma + c_1; Ma \end{aligned}$$

(see Figure 7.2). Since Ms can repeatedly and unboundedly often lose the message, which at best leads to an infinite repetition of this message, it is clear that an implementation on this basis will be able to diverge instead of delivering the message. Hence, such an implementation cannot be correct with respect to a divergence-sensitive relation like De Nicola-Hennessy must-testing. In contrast, $\sqsubseteq_{\mathbf{acc}}$, $\sqsubseteq^c_{\mathbf{shd}}$ and $\simeq_{\mathbf{bis}}$ can ignore divergence due to their built-in fairness assumption; and indeed, our implementation will be correct for each of these.

In this implementation, the sender Snd_0 gets a message with `snd`, appends the bit 0, forwards it with s_0 to the receiver and waits in state Ack_0 for an acknowledgement. In this state, the sender may accept an acknowledgement a_1 with the wrong bit, but will ignore it; the sender may repeat the message; upon getting the correct acknowledgement a_0, it will repeat its behaviour using bit 1. The receiver works analogously, starting in state Rcv where it waits for the first reception. Sender and receiver are defined by:

$$\begin{aligned} Snd_i &:= \texttt{snd}; s_i; Ack_i \\ Ack_i &:= a_i; Snd_{1-i} + a_{1-i}; Ack_i + s_i; Ack_i \\ Rcv &:= r_0; Rcv_0 \\ Rcv_i &:= \texttt{rcv}; c_i; Cnf_i \\ Cnf_i &:= r_{1-i}; Rcv_{1-i} + r_i; Cnf_i + c_i; Cnf_i \end{aligned}$$

(see Figure 7.3). In the implementation, the four components are composed using suitable synchronisation sets, and all actions except `snd` and `rcv` are hidden, i.e.

$$ABP := (\ (Snd_0 \parallel_{s_0,s_1} Ms)/s_0, s_1 \parallel_{r_0,r_1,a_0,a_1} (Rcv \parallel_{c_0,c_1} Ma)/c_0, c_1\ )/r_0, r_1, a_0, a_1\ .$$

We will show $Buf \simeq^c_{\mathbf{shd}} ABP$ in a compositional fashion, viz. by reducing subsystems of ABP while building up its transition system, which in the end will be checked against the specification. First, we compose Snd_0 with Ms synchronising over s_0, s_1. The resulting transition system is shown in Figure 7.4, where a state ij corresponds to Snd_0 being in state i according to Figure 7.3 and Ms being in state j according to Figure 7.2. s_0 and s_1 are subsequently hidden.

For the reduction of the resulting system we can use the contraction lemma (Lemma 6.2) of the previous section, by applying it to $s = 31$ and $s' = 32$ on the one hand and to $s = 61$ and $s' = 63$

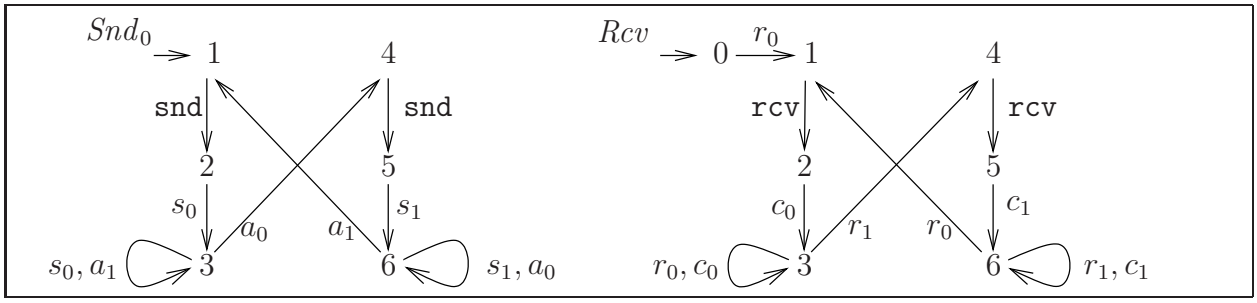


Figure 7.3: Sender and receiver.

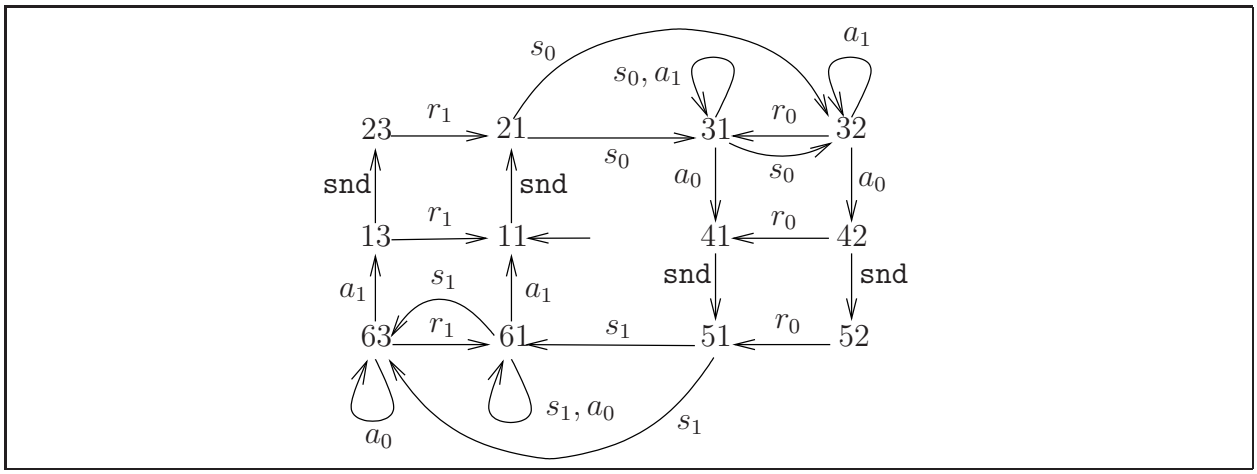


Figure 7.4: Sender composed with message channel.

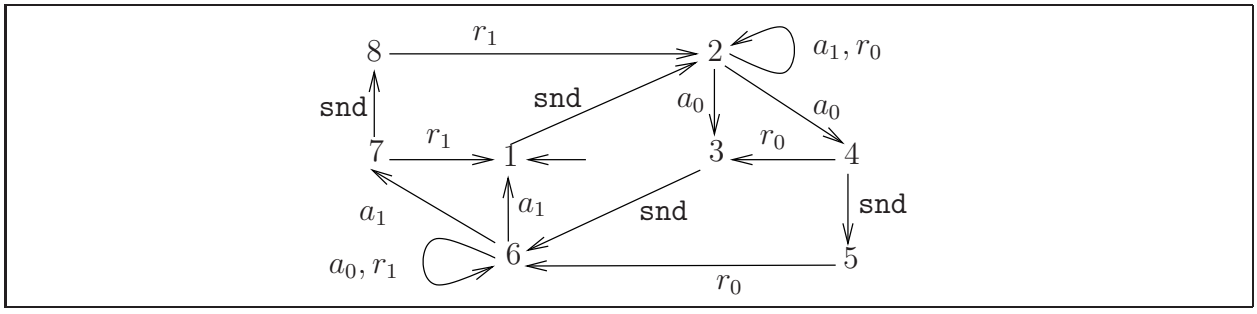


Figure 7.5: Sender and message channel after contraction.

on the other. Afterwards, we can omit the τ-loops at the contracted states according to KFAR$^-$ (2) as argued in Section 6.1. Consequently, there is only one arc leaving 21 and this is a τ-arc; according to Axiom (12), we can contract this arc and similarly for 51. The resulting system is shown in Figure 7.5 with a new enumeration of the states.

Note that this transition system is not observation congruent to the one in Figure 7.4 (with s_0 and s_1 hidden). The latter can perform `snd` to reach 32 from which a_0 can always be followed by r_0. The former necessarily reaches 2 when performing `snd`, and from there a_0 can be performed such that r_0 is impossible. Hence, the reduction discussed above is not valid up to $\simeq_{\mathbf{bis}}$; indeed, Lemma 6.2 fails for $\simeq_{\mathbf{bis}}$.

Similarly as above, we construct the transition system of $Rcv \parallel_{\{c_0,c_1\}} Ma$ (Figure 7.6) and reduce it after hiding c_0 and c_1 (Figure 7.7). As a final step, we compose the systems of Figures 7.5 and 7.7 to get the transition system of $(Snd_0 \parallel_{s_0,s_1} Ms)/s_0, s_1 \parallel_{r_0,r_1,a_0,a_1} (Rcv \parallel_{c_0,c_1} Ma)/c_0, c_1$ reduced up

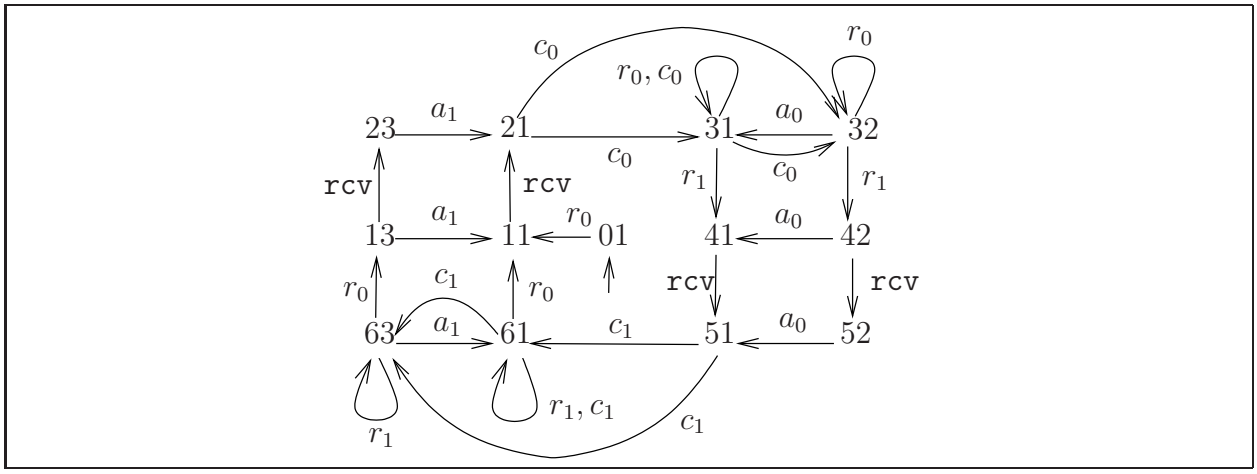


Figure 7.6: Receiver composed with acknowledgement channel.

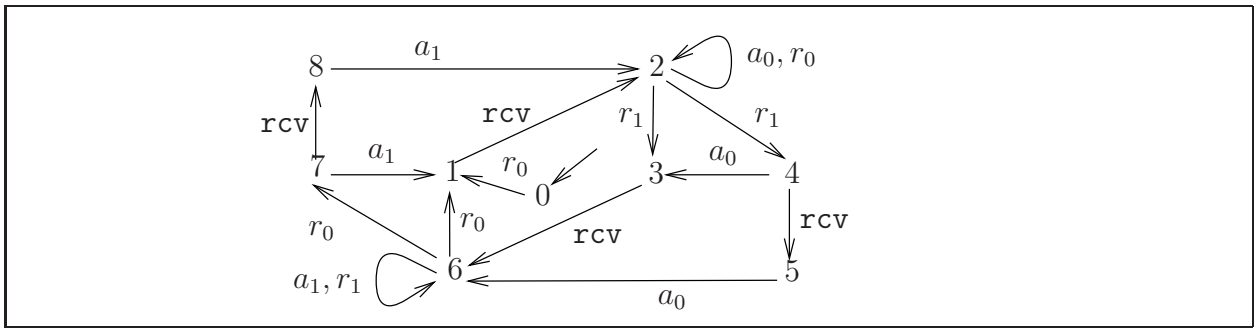


Figure 7.7: Receiver and acknowledgement channel after contraction.

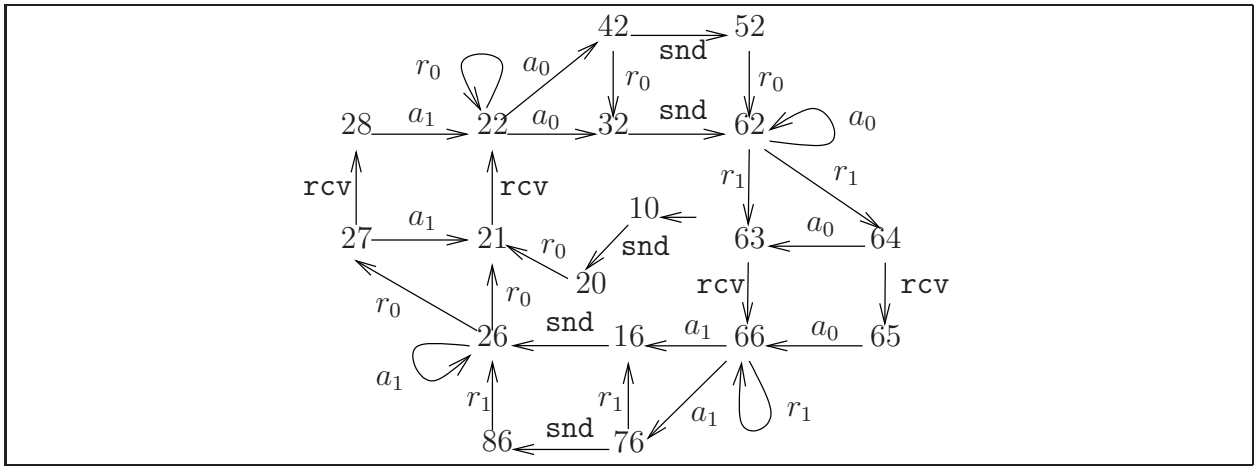


Figure 7.8: The reduced implementation.

to $\simeq^c_{\mathbf{shd}}$; see Figure 7.8.

Now one can show observation congruence of *Buf* with the system in Figure 7.8 after hiding r_0, r_1, a_0, a_1. Informally, from the start state 10 we reach 22 by performing `snd` and `rcv`; from 22, the system moves to 62 with `snd`, from 62 to 66 with `rcv`, and so on. Formally:

$$s \approx \mathit{Buf} \quad \text{for } s = 10, 16, 22, 28, 32, 42, 65, 66, 76$$
$$s \approx \mathtt{rcv}; \mathit{Buf} \quad \text{for } s = 20, 21, 26, 27, 52, 62, 63, 64, 86$$

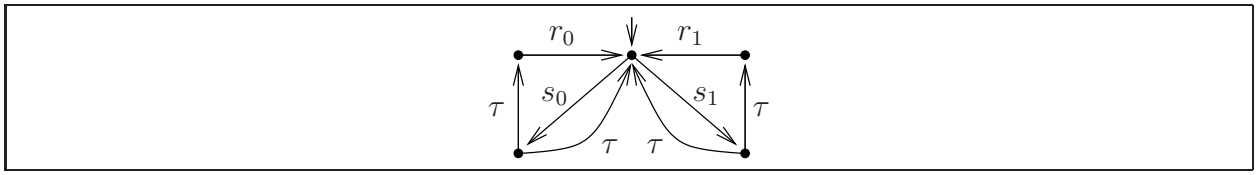


Figure 7.9: A different message channel Ms_1.

and hence $ABP \simeq^c_{\mathbf{shd}} 10$ due to compositionality of $\simeq^c_{\mathbf{shd}}$ for hiding and $10 \simeq_{\mathbf{bis}} Buf$, implying $ABP \simeq^c_{\mathbf{shd}} Buf$. Because Buf is clearly stable and deterministic, from Proposition 6.3 it now even follows that $ABP \simeq_{\mathbf{bis}} Buf$.

7.3 Alternative channels

To further stress the advantages of $\simeq^c_{\mathbf{shd}}$, we will discuss the effect of changing the behaviour of the channel Ms. For this purpose, we introduce a notation for changing environments: let θ^B_X map X to B and coincide with θ otherwise. Furthermore, we define a number of contexts.

$$\begin{aligned}
C_0[-] &= s_0; r_0; - + s_0; - + s_1; r_1; - + s_1; - \\
C_1[-] &= s_0; (\tau; r_0; - + \tau; -) + s_1; (\tau; r_1; - + \tau; -) \\
C_2[-] &= s_0; (r_0; - + -) + s_1; (r_1; - + -) \\
C_3[-] &= s_0; (r_0; - + \tau; -) + s_1; (r_1; - + \tau; -)
\end{aligned}$$

Above, we defined the message channel by $Ms := C_0[Ms]$. An alternative definition is $Ms_1 := C_1[Ms_1]$; the resulting behaviour is shown in Figure 7.9. This channel is perhaps more realistic than the original one, since the decision to lose or to deliver a message is taken after (and not while) accepting it with s_0 or s_1. The above implementation remains correct if we change to Ms_1: since $C_0[Ms] \simeq^c_{\mathbf{shd}} C_1[Ms]$ in arbitrary environments due to Axiom (15) and $\sqsubseteq^c_{\mathbf{shd}}$ is a recursive pre-congruence (Theorem 3.15), for θ as implicitly introduced in Section 7.2 we have

$$ABP \textbf{ with } \theta^{Ms_1}_{Ms} \simeq^c_{\mathbf{shd}} ABP \textbf{ with } \theta^{C_1[Ms]}_{Ms} \simeq^c_{\mathbf{shd}} ABP \textbf{ with } \theta \simeq^c_{\mathbf{shd}} Buf$$

and hence (again due to Proposition 6.3) $ABP \textbf{ with } \theta^{Ms_1}_{Ms} \simeq_{\mathbf{bis}} Buf$. Hence, in proving correctness with respect to should-testing, we can reuse the existing proof to a large degree.

This argument, which is an example of the compositionality principle discussed in Section 6.4, fails for observational congruence, since $Ms \simeq_{\mathbf{bis}} Ms_1$ is false. The compositionality argument also fails for $\simeq_{\mathbf{acc}}$: we do have $Ms \simeq_{\mathbf{acc}} Ms_1$, but since $\simeq_{\mathbf{acc}}$ is not a congruence for hiding, we cannot exchange Ms and Ms_1 in the context of ABP.

As a further variation, consider the channel $Ms_2 := C_2[Ms_2]$. This channel cannot lose a message autonomously, but if the next message arrives before the previous one was read, then it overwrites this previous message. Since $C_2[Ms] \sqsubseteq^c_{\mathbf{shd}} C_1[Ms]$ for arbitrary process environments by Axiom (18), it follows by recursive pre-congruence that

$$ABP \textbf{ with } \theta^{Ms_2}_{Ms} \sqsubseteq^c_{\mathbf{shd}} ABP \textbf{ with } \theta^{C_2[Ms]}_{Ms} \sqsubseteq^c_{\mathbf{shd}} ABP \textbf{ with } \theta^{C_1[Ms]}_{Ms} \simeq^c_{\mathbf{shd}} Buf \ .$$

Again, Proposition 6.3 then implies $ABP \textbf{ with } \theta^{Ms_2}_{Ms} \simeq_{\mathbf{bis}} Buf$ and, thus, also $ABP \textbf{ with } \theta^{Ms_2}_{Ms} \simeq^c_{\mathbf{shd}} Buf$. The same argument works for the channel $Ms_3 := C_3[Ms_3]$ used by Natarajan and Cleaveland [31], where the part $\tau; Ms_3$ describes the autonomous decision to lose the message, which can also serve for freeing the channel for the next message.

8 Concluding remarks

We briefly summarise the achievements of this paper, after which we review related work. We end the section with a discussion of open questions.

- We have defined a testing scenario, along the lines of the De Nicola-Hennessy framework, which we called *should-testing*. In this scenario, a test is satisfied if success always remains *within reach* in the system under test. Clearly, this is less demanding than requiring that success is always *reached*; in fact, there is an implicit fairness assumption in should-testing. (Section 3.2)

- We have made this fairness assumption explicit in several manners: we have shown that (for finite state systems) the should-satisfaction of a test corresponds to the certainty of reaching success under a strong fairness assumption, and we have pointed out that should-testing is strictly weaker than observation congruence, and hence satisfies (a weakened version of) Kooman's Fair Abstraction Rule. (Section 3.3)

- We have shwon that should-testing gives rise to the coarsest liveness-preserving pre-congruence for a fragment of process algebra consisting of prefixing, synchronisation, renaming and hiding — where the notion of liveness that is being preserved is that a system can't unobservably get to a state where it can never perform any visible action any more. In this respect, should-testing improves upon pure failure inclusion, which is the coarsest liveness-preserving pre-congruence for the same fragment with the exception of hiding. (Section 3.2)

- We have shown that the combination of should-testing and the preservation of stability gives rise to the coarsest liveness-preserving pre-congruence for the aforementioned fragment of process algebra plus choice. (This is in fact the standard way to obtain (pre-)congruence with respect to choice.) (Section 3.2)

- We have shown that the combination of should-testing, the preservation of stability and language equality gives rise to the coarsest liveness-preserving pre-congruence for the aforementioned fragment of process algebra plus choice and recursion — thus, for a full-fledged process algebra containing all the standard features, such as CSP, CCS, ACP or LOTOS. (Section 3.2)

- We have characterised a space of denotational models for should-testing, with constructions modelling the aforementioned process algebraic operators. At the heart of the denotational model lies an extension of standard failures which we call *tree failures*, previously studied by Vogler. (Section 4.3)

- We have defined a denotational semantics, based on the first-order constructions mentioned above and the usual least-fixpoint construction for recursion, and shown it to be equivalent to the combination of tree failures and stability from the operational semantics (Section 4.4).

- We have proved decidability of should-testing for finite state systems, and shown the decision procedure to be linear exponential. (Section 5)

- We have provided several axioms, laws and proof principles for should-testing (albeit no full axiomatisation). (Section 6)

- We have demonstrated the practical applicability of should-testing on several examples, including a busy-waiting scheme and the Alternating Bit-protocol. It is crucial here that should-testing is a pre-congruence in order for the proof to be carried out in the given, modular fashion. (Section 7)

8.1 Related work

In the introduction and in Section 3 we have already discussed the position of should-testing in the lattice of behavioural equivalences and pre-orders; see especially Proposition 3.6. Rather than repeating ourselves, here we limit ourselves to pointing out avenues of research similar to ours, and compare the results achieved.

This paper is a continuation and extension of our own work with Brinksma [10, 11], where we first presented the should-testing scenario and stated some of the congruence results, resp. presented some applications. The main achievement over these preliminary versions (apart from the fact that the present paper contains full proofs) are the denotational characterisations (which provide independent proofs for the pre-congruence properties of should-testing, which are at least in some cases much easier than the operational proofs we had in mind previously), the recursion pre-congruence result, and the proof of decidability of should-testing.

Natarajan and Cleaveland [31] concurrently and independently developed the same testing scenario that we call should-testing in this paper. They also present a — different — denotational characterisation (but no denotational constructions), and moreover give a topological argument that the difference with must-testing is small. However, they do not address congruence issues.

Boreale et al. [6] discuss and compare our should-testing to some other testing scenarios in "a general approach to define behavioural pre-orders by considering the pre-congruences induced by three basic observables". That is, they start with "basic observables" and investigate the coarsest pre-congruences generated by those, in the barbed bisimulation style (see Milner and Sangiorgi [29]). One of these basic observables corresponds to our liveness predicate, and indeed our coarsest liveness-preserving pre-congruence result corresponds to the characterisation of should-testing in that setting. Pre-congruence with respect to recursion, however, is not considered there.

Our notion of test satisfaction corresponds to the notion of liveness in Petri net theory. Therefore, the study of liveness in arbitrary contexts performed in a Petri net setting by Vogler [41] is very close to our should-testing; the impact on hiding and divergence was not considered in [41], but our denotational semantics is derived from this study.

The issue of fair testing has also been investigated in the context of the *join calculus* by Fournet and Gonthier [20]; it turns out that in that setting, part of the hierarchy of equivalences collapses so that fair testing comes to coincide with coupled simulation.

A preorder closely related to our full pre-congruence is studied by Voorhoeve and Mauw [42] for a process algebra with a very restricted form of recursion: this preorder combines $\mathcal{F}^+$-inclusion with language equality and, to cater for choice, preservation of initial stability. The authors argue that this incorporates fairness in a sensible way, but has nicer algebraic properties than our should-testing. Their claim is substantiated by giving characteristic formulae in Hennessy-Milner logic for their preorder as well as an axiomatisation. Voorhoeve and Van Glabbeek have announced that this preorder is the coarsest full pre-congruence refining acceptance testing and satisfying RSP; this is currently work in progress [37].

There is a natural link between fairness and probability theory. Roughly, one would expect that a process satisfies a test under a fairness assumption if and only if it satisfies that test under a uniform distribution of probabilities over outgoing transitions, where satisfaction becomes "even-

tually reaching success with probability 1". Núñez and Rupérez [32] show that this correspondence indeed holds, under certain restrictions stating essentially that (i) the system under test may not be infinitely branching and (ii) there is an upper bound to the 'distance' (measured in number of transitions, including internal ones) from an arbitrary state of the system under test to the nearest success state. On the other hand, the testing pre-orders studied by, e.g., Jonsson et al. [23] and Cleaveland et al. [15] are different in that they compare the likelihood of success for arbitrary tests (and hence do not restrict themselves to tests for which success is certain, as in [32]). This leads to greater distinguishing power, as shown in [30].

With respect to the examples we treat in Section 7, especially the Alternating Bit Protocol, comparable proofs in the literature have been carried out for observation congruence; see, e.g., Larsen and Milner [24]. Since our examples are indeed also correct up to observation congruence, and observation congruence is less costly to check than should-testing, one may wonder where the added value of should-testing lies. As an answer to this, we have pointed out that, up to should-testing, state space reductions are possible on individual modules of the system that are invalid up to observation congruence; since the sub-systems that are being reduced are much smaller than the total system, there is a real space and time benefit. See also Valmari [34] for an extensive discussion of this point. Another advantage of should-testing is that it is a pre-order rather than an equivalence. Again in Section 7, several communication channel implementations are considered, each modelling a subtly different kind of data loss. We could immediately conclude that the protocol is correct for each of these kinds of media. Observation congruence does not support this particular proof strategy, because the different channels are not observation congruent — even if the protocol as a whole is still correct modulo observation congruence.

We close with a short review of earlier attempts to solve the problem to find a fair testing pre-congruence. In fact, we can here rely very much on Leduc [25], who compares three approaches and presents a new one. To ease the discussion, we limit this review to equalities, although there are related pre-orders in all cases.

The first approach, called FAUD-semantics (Failures with Abstraction from Unfair Divergences) in [25], was developed by Bergstra et al. [2]. Here, processes are considered equivalent if they are language equivalent and have the same *stable failures*; a stable failure is a failure pair (w, A) of a process B, if $B \stackrel{w}{\Longrightarrow} C$ such that $\mathcal{L}(C) \cap A = \emptyset$ and C is stable. Clearly, this semantics ignores runs that lead to a diverging process.

The second and third approaches are defined by Valmari and Tienari [35, 36]. The CFFD-semantics (Chaos-Free Failures Divergences, see [35]) refines the FAUD-semantics by additionally considering the set of traces that lead to a diverging process and the set of infinite traces. This semantics has been studied in a number of later papers. The NDFD-semantics (Non-Divergent Failures Divergences, see [36]) abstracts to some degree from the CFFD-semantics, by considering only convergent failures instead of stable failures. A failure pair (w, A) of B is convergent, if B' is not divergent for any B' with $B \stackrel{w}{\Longrightarrow} B'$.

Leduc [25] shows that CFFD-equivalence is the intersection of FAUD- and NDFD-equivalence, and that none of the three is comparable to simple failure equivalence. Since in the present paper we have looked at the coarsest congruence within simple failure equivalence, and since FAUD-, CFFD- and NDFD-equivalence are congruences as well, it follows that should-testing cannot be weaker than these three. A significant example taken from [25] are the processes $B = \mathrm{rec}\, X.\, a; \mathbf{0} + \tau; X$ and $C = a; \mathbf{0} + \tau; \mathrm{rec}\, X.\, \tau; X$. These processes are FAUD-, CFFD- and NDFD-equivalent; note that the empty sequence leads to divergence, and that all stable/convergent failures have the form (a, A). We would argue that this is intuitively wrong, and B and C are indeed distinguished by should-testing: B has a τ-loop initially, which under the assumption of fairness will eventually be

abandoned in favour of a; in contrast, C can choose internally to behave like a livelock that will never perform a. On the other hand, the processes $B = a; \mathbf{0}$ and $C = a; \mathrm{rec}\, X.\tau; X$ are distinguished by FAUD and NDFD (and hence also by CFFD) but not by should-testing. Hence should-testing is incomparable to these three.

The new suggestions in [25] are two semantics supported by intuitive arguments. The respective equivalences are not congruences, and the coarsest congruence refining these equivalences turns out to be NDFD-equivalence. Thus, this contribution actually supports NDFD-equivalence.

Finally, we want to mention Erdogmus et al. [19]. They introduce a variant of labelled transition systems where internal transitions are avoided; instead, the system has a kind of macro states corresponding to processes, where each macro state corresponding to B has a set of internal states which correspond more or less to the processes that can be reached from B internally. Furthermore, divergence predicates are added to the macro states which allow to distinguish divergences that a process can depart from under a fairness assumption from other divergences. While this is a point of contact to our approach, a clear difference is that an operational semantics defined according to [19] is divergence-sensitive. They do not present congruence results.

8.2 Open questions

The largest remaining open question is that of the axiomatic theory of should-testing. Section 6 provides some axioms and laws, but they are not complete. We have the impression (supported by the complexity of the denotational characterisation) that if it exists at all, a complete axiomatisation will be very difficult to construct.

To give one particular example where the axiomatic theory is lacking: it is not yet completely understood whether the contraction lemma (Lemma 6.2), which is currently given in operational terms, has a proper algebraic formulation. So far, we have only found algebraic statements corresponding to special cases.

Acknowledgements

We thank Ed Brinksma for his contribution to the precursors of this paper, and for numerous discussions on the current one. The first author also gladly acknowledges fruitful discussions on the topic of this paper with Rom Langerak (who provided the counter-example against recursion pre-congruence in the absence of language equivalence) and Rob van Glabbeek (who put us on the right track towards the coarsest pre-congruence property for synchronisation and hiding).

References

[1] J. C. M. Baeten and W. P. Weijland. *Process Algebra*. Cambridge University Press, 1990.

[2] J. A. Bergstra, J. W. Klop, and E.-R. Olderog. Failures without chaos: A new process semantics for fair abstraction. In M. Wirsing, editor, *Formal Description of Programming Concepts — III*, pages 77–103. IFIP, Elsevier Science Publishers B.V., 1987.

[3] B. Bloom. Structural operational semantics for weak bisimulation. *Theoretical Comput. Sci.*, 146:25–68, 1995. Report version: TR 93–1373, Dept. of Computer Science, Cornell University, Aug. 1993.

[4] B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced. *J. ACM*, 42(1):232–268, Jan. 1995.

[5] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks and ISDN Systems*, 14:25–59, 1987.

[6] M. Boreale, R. De Nicola, and R. Pugliese. Basic observables for processes. *Information and Computation*, 149:77–98, 1999.

[7] E. Brinksma. On the existence of canonical testers. Memorandum INF-87-5, University of Twente, 1987.

[8] E. Brinksma. A theory for the derivation of tests. In S. Aggarwal and K. Sabnani, editors, *Protocol Specification, Testing, and Verification VIII*, pages 63–74. Elsevier Science Publishers B.V., 1988. Report version: Memoranda Informatica 88–19, University of Twente.

[9] E. Brinksma. Cache consistency by design. *Distributed Computing*, 2–3(12):552–565, 1999. Conference version in PSTV '94.

[10] E. Brinksma, A. Rensink, and W. Vogler. Fair testing. In I. Lee and S. A. Smolka, editors, *Concur '95: Concurrency Theory*, volume 962 of *Lecture Notes in Computer Science*, pages 313–327. Springer-Verlag, 1995.

[11] E. Brinksma, A. Rensink, and W. Vogler. Applications of fair testing. In *Protocol Specification, Testing, and Verification, XVI*. IFIP, Chapman & Hall, 1996.

[12] E. Brinksma and G. Scollo. The characterization of implementations of LOTOS specifications. In *NGI-SION Symposium 'Stimulerende Informatica'*, pages 485–496. Stichting Informatica Congressen, Amsterdam, 1986. Full report version: Memorandum INF–86–13, University of Twente.

[13] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31(3):560–599, July 1984.

[14] S. D. Brookes and A. W. Roscoe. An improved failures model for communicating processes. In S. D. Brookes, A. W. Roscoe, and G. Winskel, editors, *Seminar on Concurrency*, volume 197 of *Lecture Notes in Computer Science*, pages 281–305. Springer-Verlag, 1985.

[15] R. Cleaveland, Z. Dayar, S. A. Smolka, and S. Yuen. Testing preorders for probabilistic processes. *Information and Computation*, 154(2):93–148, 1999.

[16] R. Cleaveland and M. C. B. Hennessy. Testing equivalence as a bisimulation equivalence. *Formal Aspects of Computing*, 5:1–20, 1993.

[17] R. De Nicola and M. C. B. Hennessy. Testing equivalences for processes. *Theoretical Comput. Sci.*, 34:83–133, 1984.

[18] R. De Simone. Higher-level synchronising devices in Meije-SCCS. *Theoretical Comput. Sci.*, 37:245–267, 1985.

[19] M. H. Erdogmus, R. Johnston, and M. Ferguson. On the operational semantics of nondeterminism and divergence. *Theoretical Comput. Sci.*, 159:271–317, 1996.

[20] C. Fournet and G. Gonthier. A hierarchy of equivalences for asynchronous calculi. *J. Logic and Algebraic Programming*, 63:131–173, 2005.

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

[22] ISO. Information processing systems — open systems interconnection — LOTOS — a formal description technique based on the temporal ordering of observational behaviour. International Standard 8807, ISO, Geneva, Feb. 1989. 1st Edition.

[23] B. Jonsson, W. Yi, and K. G. Larsen. Probabilistic extensions of process algebras. In J. A. Bergstra, A. Ponse, and S. A. Smolka, editors, *Handbook of Process Algebra*, chapter 11, pages 685–710. Elsevier, 2001.

[24] K. G. Larsen and R. Milner. Verifying a protocol using relativized bisimulation. In T. Ottman, editor, *Automata, Languages and Programming*, volume 267 of *Lecture Notes in Computer Science*, pages 126–135. Springer-Verlag, 1987.

[25] G. Leduc. Failure-based congruences, unfair divergences, and new testing theory. In S. Vuong and S. Chanson, editors, *Protocol Specification, Testing, and Verification, XIV*, IFIP Series. Chapman & Hall, 1995.

[26] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[27] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Comput. Sci.*, 25:267–310, 1983.

[28] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[29] R. Milner and D. Sangiorgi. Barbed bisimulation. In W. Kuich, editor, *Automata, Languages and Programming*, volume 623 of *Lecture Notes in Computer Science*, pages 685–695. Springer-Verlag, 1992.

[30] K. Narayan Kumar, R. Cleaveland, and S. A. Smolka. Infinite probabilistic and nonprobabilistic testing. In V. Arvind and R. Ramunajam, editors, *Foundations of Software Technology and Theoretical Computer Science*, volume 1530 of *Lecture Notes in Computer Science*, pages 209–220. Springer-Verlag, 1998.

[31] V. Natarajan and R. Cleaveland. Divergence and fair testing. In Z. Fülöp and F. Gécseg, editors, *Automata, Languages and Programming*, volume 944 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[32] M. Núñez and D. Rupérez. Fair testing through probabilistic testing. In J. Wu, S. T. Chanson, and Q. Gao, editors, *Formal Methods for Protocol Engineering and Distributed Systems*, volume 156 of *IFIP Conference Proceedings*, pages 135–150. Kluwer, 1999.

[33] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proceedings 5th GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.

[34] A. Valmari. Failure-based equivalences are faster than many believe. In J. Desel, editor, *Structures in Concurrency Theory*, Workshops in Computing, pages 326–340. Springer-Verlag, 1995.

[35] A. Valmari and M. Tienari. An improved failures equivalence for finite-state systems with a reduction algorithm. In B. Jonsson, J. Parrow, and B. Pehrson, editors, *Protocol Specification, Testing and Verification XI*, pages 3–18. IFIP WG 6.1, North-Holland Publishing Company, 1991.

[36] A. Valmari and M. Tienari. Compositional failure-based semantic models for basic LOTOS. *Formal Aspects of Computing*, 7:440–468, 1995. Report version: Tampere University of Technology, Software Systems Laboratory, Report 16, Tampere, Finland, July 1993.

[37] R. van Glabbeek. A congruence result for impossible futures. Private communication, 2005.

[38] R. J. van Glabbeek. The linear time – branching time spectrum II: The semantics of sequential systems with silent moves. In E. Best, editor, *Concur '93*, volume 715 of *Lecture Notes in Computer Science*, pages 66–81. Springer-Verlag, 1993.

[39] R. J. van Glabbeek. The linear time – branching time spectrum I: The semantics of concrete, sequential processes. In J. A. Bergstra, A. Ponse, and S. A. Smolka, editors, *Handbook of Process Algebra*, chapter 1, pages 3–100. Elsevier, 2001.

[40] W. Vogler. Failures semantics and deadlocking of modular Petri nets. *Acta Inf.*, 26:333–348, 1989.

[41] W. Vogler. *Modular Construction and Partial Order Semantics of Petri Nets*, volume 625 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[42] M. Voorhoeve and S. Mauw. Impossible futures and determinism. *Information Processing Letters*, 80(1):51–58, 2001.

A Completeness

Apart from the *correctness* property in Proposition 4.11, which confirms that the tree failures of any term of $\mathbf{L}$ form an element of $\mathbf{T}$ and thus tells us that the saturation properties of $\mathbf{T}$ are not too strong, we would like to know the (dual) *completeness property* that any element of $\mathbf{T}$ constitutes the set of tree failures of a term, or in other words, that $\mathbf{T}$ contains 'no junk', which tells us that we have not 'forgotten' any saturation properties.

Unfortunately, in its simplest form the completeness property does *not* hold, due to the cardinalities of tree failure sets (which may be uncountable) on the one hand and of terms of $\mathbf{L}$ (which have countable sums and sorts) on the other. However, the reason why this fails is beside the main point of this paper: it has nothing to do with the tree failures or with fair testing in itself, merely with the very technical issue of choosing the right cardinalities. Let us elaborate on this.

Calling a tree failure set $\mathcal{F} \in \mathbf{T}$ *countably representable* if there is a term $B \in \mathbf{L}$ such that $\mathcal{F} = \mathcal{F}^+(B)$, $\mathcal{F}$ may fail to be countably representable for two distinct reasons:

- The alphabet of actions that can be performed may be uncountable: for instance, $(a, \emptyset) \in \mathcal{F}$ for all $a \in \mathbf{A}$. A term representing this behaviour would need to have an uncountable sort, which is ruled out in $\mathbf{L}$.
- The degree of nondeterminism may be uncountable: for instance, if there is a countably infinite set of actions $A \subseteq \mathbf{A}$ such that for all $A' \subseteq \mathbf{A}$, $(\varepsilon, A') \in \mathcal{F}$ iff $A \setminus A'$ is infinite.

It is not difficult to formulate a restriction ruling out the first of these cases. For this purpose, we successively define the *language* and the *alphabet* of a tree failure set, $\mathcal{F} \in \mathbf{T}$:

$$\begin{aligned} \mathcal{L}(\mathcal{F}) &:= \{w \in \mathbf{A}^* \mid (w, \emptyset) \in \mathcal{F}\} \\ \mathcal{A}(\mathcal{F}) &:= \{a \in \mathbf{A} \mid \exists w \in \mathbf{A}^* : w\,a \in \mathcal{L}(\mathcal{F})\} \end{aligned}$$

It should be clear that $\mathcal{L}(B) = \mathcal{L}(\mathcal{F}^+(B))$ for all closed $B \in \mathbf{L}$; moreover, for arbitrary $\mathcal{F}, \mathcal{G} \in \mathbf{T}$, if $\mathcal{F} \sqsubseteq_{\mathbf{T}} \mathcal{G}$ then $\mathcal{L}(\mathcal{F}) \subseteq \mathcal{L}(\mathcal{G})$. An alternative characterisation for the language is $\mathcal{L}(\mathcal{F}) = \{w \in \mathbf{A}^* \mid \exists (w\,v, V) \in \mathcal{F}\}$.

Note the difference between the concepts of alphabet and *sort*: the latter is a syntactic notion, which is defined for terms only, whereas the alphabet is the set of actions actually occurring. Since the latter is a subset of the former, and the former is countable, for the completeness of $\mathbf{T}$ we have to restrict to $\mathcal{F} \in \mathbf{T}$ for which $\mathcal{A}(\mathcal{F})$ is a countable set.

On the other hand, we have not found a good characterisation for countable non-determinism, due to the aforementioned discrepancies in cardinalities between syntax and semantics. In fact, we can recover the 'no junk' property by (temporarily) allowing uncountable sums and process environments. To remain consistent (for instance, to guarantee the existence of fresh variables in the definition of syntactic substitution, see Section 2.2) we should then also choose a higher cardinality for $\mathbf{X}$, the set of all process variables.

In the following theorem, we use $\sum_{i \in I}^{\infty} B_i$ to denote summation with a possibly uncountable index set I, with $\bigcup_{i \in I} \mathcal{S}(B_i)$ countable, and $\mathrm{rec}_X^{\infty}\,\theta$ to denote recursion with a possibly uncountable process environment θ, with $\mathcal{S}(\theta(X))$ countable for all $X \in \mathit{dom}(\theta)$. The extended language is denoted $\mathbf{L}^{\infty}$. Note that the semantics of the extended operators is the same as for the original ones.

Theorem A.1 *$\mathcal{F} \in \mathbf{T}$ with $\mathcal{A}(\mathcal{F})$ countable if and only if $\mathcal{F} = \mathcal{F}^+(B)$ for some $B \in \mathbf{L}^{\infty}$.*

The proof is quite involved, and since this is an isolated result not used in the main body of the paper, we advise the casual reader to move on to Section 4.3.

Proof of Theorem A.1. First, we show a property we will need in this proof: for $w \notin {\uparrow}W$ we have $w^{-1}{\uparrow}W = {\uparrow}(w^{-1}W)$. For the inclusion, consider $x \in w^{-1}{\uparrow}W$, i.e. $w\,x \in {\uparrow}W$; by assumption on w, we get $x = x_1x_2$ with $w\,x_1 \in W$, hence $x_1 \in w^{-1}W$ and $x = x_1x_2 \in {\uparrow}(w^{-1}W)$. The reverse inclusion is easy.

The "if" direction of the theorem consists of proving that $\mathcal{F}^+(B)$ meets the constraints in the definition of $\mathbf{T}$, which is a straightforward generalisation of Proposition 4.11 and omitted here, and proving that $\mathcal{A}(\mathcal{F}^+(B))$ is countable, which follows from the fact that $\mathcal{A}(\mathcal{F}^+(B)) \subseteq \mathcal{S}(B)$ and $\mathcal{S}(B)$ is countable.

For the "only if", we have to construct a term $B \in \mathbf{L}^\infty$ for an arbitrary $\mathcal{F} \in \mathbf{T}$ with $\mathcal{A}(\mathcal{F})$ countable, such that $\mathcal{F}^+(B) = \mathcal{F}$. For this purpose, first we concentrate on the *saturated* tree failures of $\mathcal{F}$, which are the pairs $(v, V) \in \mathcal{F}$ that satisfy

$$V = \uparrow V \wedge \forall w \notin V : (v\,w, w^{-1}V) \in \mathcal{F} \ .$$

Let us denote the saturated tree failures of $\mathcal{F}$ by $sat(\mathcal{F})$. The saturated failures satisfy a number of special properties, of which we list some that we will use later on.

(*i*) $(\varepsilon, \mathbf{A}^* \setminus \mathcal{L}(\mathcal{F})) \in sat(\mathcal{F})$ for all $\mathcal{F} \in \mathbf{T}$.

To see this, take $w \in \mathcal{L}(\mathcal{F})$ and $W = w^{-1}(\mathbf{A}^* \setminus \mathcal{L}(\mathcal{F}))$; then $u \in W$ implies $wu \notin \mathcal{L}(\mathcal{F})$, i.e. $(wu, \emptyset) \notin \mathcal{F}$; with this we can apply extension saturation to $(w, \emptyset) \in \mathcal{F}$ to get $(w, W) \in sat(\mathcal{F})$. In particular, for $w = \varepsilon$ this gives $(\emptyset, \mathbf{A}^* \setminus \mathcal{L}(\mathcal{F})) \in sat(\mathcal{F})$.

(*ii*) $(v, V) \in sat(\mathcal{F})$ implies $\downarrow V = \mathbf{A}^*$ for all $\mathcal{F} \in \mathbf{T}$.

To see this, take $a \in \mathbf{A} \setminus \mathcal{A}(\mathcal{F})$ (which exists due to the difference in cardinalities between $\mathcal{A}(\mathcal{F})$ and $\mathbf{A}$); then for all $w \in \mathbf{A}^*$ we have $v\,w\,a \notin \mathcal{L}(\mathcal{F})$, implying $(v\,w\,a, w\,a^{-1}V) \notin \mathcal{F}$ and hence (by saturation) $w\,a \in V$; it follows that $w \in \downarrow V$.

(*iii*) If $(v, V) \in sat(\mathcal{F})$ and $w \notin V$, then $(v\,w, w^{-1}V) \in sat(\mathcal{F})$ as well.

To see this, note that $w^{-1}V = \uparrow(w^{-1}V)$; moreover, $u \notin w^{-1}V$ implies $w\,u \notin V$ and $(v\,w\,u, u^{-1}w^{-1}V) \in \mathcal{F}$.

(*iv*) $sat(\mathcal{F}) \subseteq \mathcal{G}$ implies $\mathcal{F} \subseteq \mathcal{G}$ for arbitrary $\mathcal{F}, \mathcal{G} \in \mathbf{T}$.

To see this, consider arbitrary $\mathcal{F} \in \mathbf{T}$ and $(v, V) \in \mathcal{F}$ and let

$$\overline{(v, V)} = (v, \uparrow V \cup \uparrow\{w \mid (v\,w, w^{-1}V) \notin \mathcal{F}\}) \ .$$

We will show that $\overline{(v, V)} \in sat(\mathcal{F})$; then $\overline{(v, V)} \in \mathcal{G}$ implies $(v, V) \in \mathcal{G}$ for arbitrary $\mathcal{G} \in \mathbf{T}$.

Take $W = \{w \mid (v\,w, w^{-1}V) \notin \mathcal{F}\}$; then $\overline{(v, V)} = (v, \uparrow(V \cup W))$. By saturation under extension, $(v, V \cup W) \in \mathcal{F}$ and thus $(v, \uparrow(V \cup W)) \in \mathcal{F}$. It remains to show that $(v, \uparrow(V \cup W))$ is saturated. Obviously, $\uparrow\uparrow(V \cup W) = \uparrow(V \cup W)$. Now take some $u \notin \uparrow(V \cup W)$; we have to show $(v\,u, u^{-1}\uparrow(V \cup W)) \in \mathcal{F}$. Since $u \notin \uparrow(V \cup W)$, we have by the property shown at the beginning of this proof that $u^{-1}\uparrow(V \cup W) = \uparrow u^{-1}(V \cup W)$; thus it suffices to show $(v\,u, u^{-1}(V \cup W)) \in \mathcal{F}$. Since $u \notin W$, we have $(v\,u, u^{-1}V) \in \mathcal{F}$; now we are done by saturation under extension since $w \in u^{-1}(V \cup W) \setminus u^{-1}V$ implies $u\,w \in W$, i.e. $(v\,u\,w, w^{-1}u^{-1}V) \notin \mathcal{F}$.

Now we assume the existence of a set of variables $Z_{v,V}$ for all $(v, V) \in sat(\mathcal{F})$, and we define a process environment $\theta_{\mathcal{F}}$ with

$$Z_{v,V} := \textstyle\sum^\infty \{\tau; Z_{v,W} \mid V \subset W, (v, W) \in sat(\mathcal{F})\} + \sum \{a; Z_{v\,a, a^{-1}V} \mid a \notin V\} \ .$$

for all $(v, V) \in sat(\mathcal{F})$.[9] Note that $dom(\theta_{\mathcal{F}})$ is, in general, uncountable, and so is the first summation in the above definition; here is where we need the corresponding extensions to $\mathbf{L}$. Note that $(v\,a, a^{-1}V) \in sat(\mathcal{F})$ for all $(v, V) \in sat(\mathcal{F})$ and $a \notin V$, due to property (*iii*) above; hence $\mathrm{rec}_Z^\infty\,\theta_{\mathcal{F}}$ is closed for all $Z \in dom(\theta_{\mathcal{F}})$.

We have $Z_{\varepsilon, \mathbf{A}^* \setminus \mathcal{L}(\mathcal{F})} \in dom(\theta_{\mathcal{F}})$ due to property (*i*) above. We will denote $B_{\mathcal{F}} = Z_{\varepsilon, \mathbf{A}^* \setminus \mathcal{L}(\mathcal{F})}$ (where all variables $Z \in dom(\theta_{\mathcal{F}})$ are implicitly interpreted as $\mathrm{rec}_Z^\infty\,\theta_{\mathcal{F}}$); this is the required term whose tree failure set equals $\mathcal{F}$. The heart of the proof is the following property:

(∗) For all $(v, V) \in sat(\mathcal{F})$, $Z_{v,V} \stackrel{w}{\Longrightarrow} B$ if and only if $B = Z_{v\,w,W}$ for some $(v\,w, W) \in sat(\mathcal{F})$ with $W \supseteq w^{-1}V$.

Among other things, from this it follows that $\mathcal{L}(Z_{v,V}) \cap V = \emptyset$ (since $\varepsilon \notin W$); hence (in combination with (*iii*) above) $\mathcal{L}(Z_{v,V}) = \mathbf{A}^* \setminus V$ for all $(v, V) \in sat(\mathcal{F})$. The required property $\mathcal{F}^+(B_{\mathcal{F}}) = \mathcal{F}$ is then proved by the combination of the following two items, together with property (*iv*) above:

[9] This process definition is the counterpart, for the case of tree failures, of the *canonical testers* in [7] or the *acceptance graphs* in [16].

- $\mathcal{F}^+(B_\mathcal{F}) \subseteq \mathcal{F}$. To see this, let $(v,V) \in \mathcal{F}^+(B_\mathcal{F})$ be arbitrary; thus $\exists (B_\mathcal{F} \stackrel{v}{\Longrightarrow} B)$ such that $\mathcal{L}(B) \cap V = \emptyset$. According to $(*)$, this implies $B = Z_{v,W}$ for some $(v,W) \in sat(\mathcal{F})$. As observed above, it follows that $\mathcal{L}(B) = \mathbf{A}^* \setminus W$, and thus $V \subseteq W$; hence $(v,V) \in \mathcal{F}$ due to the saturation properties of $\mathcal{F}$.
- $sat(\mathcal{F}) \subseteq \mathcal{F}^+(B_\mathcal{F})$. To see this, let $(v,V) \in sat(\mathcal{F})$ be arbitrary. $w \in v^{-1}(\mathbf{A}^* \setminus \mathcal{L}(\mathcal{F}))$ implies $(v\,w, w^{-1}V) \notin \mathcal{F}$ and hence (by definition of $sat(\mathcal{F})$) $w \in V$; thus we have $V \supseteq v^{-1}(\mathbf{A}^* \setminus \mathcal{L}(\mathcal{F}))$. Due to $(*)$, it follows that $B_\mathcal{F} \stackrel{v}{\Longrightarrow} Z_{v,V}$ and (as noted above) $\mathcal{L}(Z_{v,V}) \cap V = \emptyset$; hence $(v,V) \in \mathcal{F}^+(B_\mathcal{F})$.

It only remains to prove $(*)$.

If. By induction on the length of w.

Base case. Assume $w = \varepsilon$. It follows that $V \subseteq W$, implying either $Z_{v,V} = Z_{v,W}$ or $Z_{v,V} \stackrel{\tau}{\longrightarrow} Z_{v,W}$; in either case, $Z_{v,V} \stackrel{w}{\longrightarrow} Z_{v\,w,W}$.

Induction step. Assume $w = a\,u$, and assume the property has been proved for u. If $a \in V$ then by $V = {\uparrow}V$ we would have $w \in V$, which contradicts $(v\,w, W) \in \mathcal{F}$ with $W \supseteq w^{-1}V$. Thus, we have $a \notin V$. We can therefore derive

$$Z_{v,V} \stackrel{a}{\longrightarrow} Z_{v\,a,a^{-1}V} \stackrel{u}{\Longrightarrow} Z_{v\,w,W}$$

where the first step is by the definition of $\theta_\mathcal{F}$ and the second by the induction hypothesis (noting that $w^{-1}V = u^{-1}a^{-1}V$).

Only if. By induction on the length n of the transition sequence $Z_{v,V} \stackrel{\alpha_1}{\longrightarrow} \cdots \stackrel{\alpha_n}{\longrightarrow} B$ underlying $Z_{v,V} \stackrel{w}{\Longrightarrow} B$ (hence w equals $\alpha_1 \cdots \alpha_n$ with τ's removed). If $n = 0$, then the property trivially holds (with $W = V$). Now assume the property holds up to $n-1$, and regard the transition $Z_{v,V} \stackrel{\alpha_1}{\longrightarrow} B' \stackrel{\alpha_2}{\longrightarrow} \cdots \stackrel{\alpha_n}{\longrightarrow} B$. Let u equal $\alpha_2 \cdots \alpha_n$ with τ's removed. By construction of $\theta_\mathcal{F}$, we distinguish two cases.

- $\alpha_1 = \tau$, in which case $u = w$ and $B' = Z_{v,W}$ for some $W \supset V$. By the induction hypothesis, then, $B = Z_{v\,w,W'}$ for some $W' \supseteq w^{-1}W$. Since $w^{-1}W \supseteq w^{-1}V$, we are done.
- $\alpha_1 = a \in \mathbf{A}$, in which case $w = a\,u$, $a \notin V$ and $B' = Z_{v\,a,a^{-1}V}$. By the induction hypothesis, then, $B = Z_{v\,a\,u,W}$ for some $(v\,a\,u, W) \in sat(\mathcal{F})$ where $W \supseteq u^{-1}a^{-1}V$; since the latter set equals $w^{-1}V$, we are done.

□

B Additional proofs

This appendix contains some of the more technical and uninteresting proofs.

Lemma 3.8 $\sqsubseteq^+_{\mathbf{shd}}$ *is a pre-congruence for choice.*

In order to prove this, we first show that *stable* tests are sufficient to check $\sqsubseteq_{\mathbf{shd}}$.

Lemma B.1 *If S **shd** t implies I **shd** t whenever t **stb**, then $I \sqsubseteq_{\mathbf{shd}} S$.*

Proof. For arbitrary $t \in \mathbf{L}_{\checkmark}$, let $\hat{t} = \sum\{\alpha; t' \mid t \stackrel{\alpha}{\Longrightarrow} t', \alpha \neq \tau\}$. Note that $t \stackrel{w}{\Longrightarrow}$ iff $\hat{t} \stackrel{w}{\Longrightarrow}$ for all $w \in \mathbf{A}^*_{\checkmark}$, and moreover, if $|w| > 0$ then $t \stackrel{w}{\Longrightarrow} t'$ iff $\hat{t} \stackrel{w}{\Longrightarrow} t'$.

We show that for arbitrary $B \in \mathbf{L}$ and $t \in \mathbf{L}_{\checkmark}$, B **shd** t iff B **shd** $\hat{t}'$ whenever $t \stackrel{\varepsilon}{\Longrightarrow} t'$. Since $\hat{t}'$ **stb** for all $t' \in \mathbf{L}_{\checkmark}$, this establishes the lemma.

if. Assume $B \parallel t \stackrel{w}{\Longrightarrow} B'$ for some $w \in \mathbf{A}^*$. If $|w| > 0$ then $B \parallel \hat{t} \stackrel{w}{\Longrightarrow} B'$, and hence $\exists v \in \mathbf{A}^* : B' \stackrel{v\checkmark}{\Longrightarrow}$ due to B **shd** $\hat{t}$. Otherwise $B' = B'' \parallel t''$ where $B \stackrel{\varepsilon}{\Longrightarrow} B''$ and $t \stackrel{\varepsilon}{\Longrightarrow} t''$. Due to B **shd** $\hat{t}''$ it follows that $\exists v \in \mathbf{A}^* : B'' \parallel \hat{t}'' \stackrel{v\checkmark}{\Longrightarrow}$; but then also $B' \stackrel{v\checkmark}{\Longrightarrow}$. We may conclude B **shd** t.

only if. Assume $B \parallel \hat{t}' \stackrel{w}{\Longrightarrow} B'$ for some $t \stackrel{\varepsilon}{\Longrightarrow} t'$ and $w \in \mathbf{A}^*$. If $|w| > 0$ then $B \parallel t \stackrel{w}{\Longrightarrow} B'$, and hence $\exists v \in \mathbf{A}^* : B' \stackrel{v\checkmark}{\Longrightarrow}$ due to B **shd** t. Otherwise $B' = B'' \parallel t''$ where $B \stackrel{\varepsilon}{\Longrightarrow} B''$ and $\hat{t}' \stackrel{\varepsilon}{\Longrightarrow} t''$; hence (due to $\hat{t}'$ **stb**) $t'' = \hat{t}'$. Due to B **shd** t it follows that $\exists v \in \mathbf{A}^* : B'' \parallel t' \stackrel{v\checkmark}{\Longrightarrow}$; but then also $B' \stackrel{v\checkmark}{\Longrightarrow}$. We may conclude B **shd** $\hat{t}'$.

□

Moreover, we need another auxiliary relation. We define a sub-predicate $\mathbf{shd}^1$ of **shd** that tests just for states reachable through at least one visible action.

$$B \ \mathbf{shd}^1 \ t \quad :\Leftrightarrow \quad \forall w \in \mathbf{A}^+ : \forall B' : (B \parallel t) \stackrel{w}{\Longrightarrow} B' \text{ implies } \exists v \in \mathbf{A}^* : B' \stackrel{v\checkmark}{\Longrightarrow} \quad .$$

The following lemma states that the resulting testing pre-order, $\sqsubseteq_{\mathbf{shd}^1}$, is weaker than $\sqsubseteq_{\mathbf{shd}}$.

Lemma B.2

$$\sqsubseteq_{\mathbf{shd}} \subseteq \sqsubseteq_{\mathbf{shd}^1} \quad .$$

Proof. Assume $I \sqsubseteq_{\mathbf{shd}} S$ and S $\mathbf{shd}^1$ t, and assume $I \parallel t \stackrel{w}{\Longrightarrow} I' \parallel t'$ for some $w \in \mathbf{A}^+$. Let

$$\begin{aligned} \hat{t} &= t \parallel (\checkmark; \mathbf{0} + \mathrm{rec}\, X. \sum\{a; X \mid a \in \mathcal{S}(t)\}) \\ \hat{t}' &= t' \parallel \mathrm{rec}\, X. \sum\{a; X \mid a \in \mathcal{S}(t)\} \quad . \end{aligned}$$

Clearly S **shd** $\hat{t}$ and hence I **shd** $\hat{t}$. Since $\hat{t} \stackrel{w}{\Longrightarrow} \hat{t}'$ it follows that $I \| \hat{t} \stackrel{w}{\Longrightarrow} I' \| \hat{t}'$ and hence $\exists v \in \mathbf{A}^* : I' \| \hat{t}' \stackrel{v\checkmark}{\Longrightarrow}$. But then also $I \parallel t' \stackrel{v\checkmark}{\Longrightarrow}$. We may conclude I $\mathbf{shd}^1$ t, and hence $I \sqsubseteq_{\mathbf{shd}^1} S$. □

Proof of Lemma 3.8. Assume $I_k \sqsubseteq^+_{\mathbf{shd}} S_k$ for $k \in K$; we have to show that $I \sqsubseteq^+_{\mathbf{shd}} S$ for $I = \sum_{k\in K} I_k$ and $S = \sum_{k\in K} S_k$. The case for $\sqsubseteq_{\mathbf{stb}}$ is easily shown, so we concentrate on demonstrating $I \sqsubseteq_{\mathbf{shd}} S$.

We show that the following characterisation for the *stable* should-tests of a sum-term $B = \sum_{k\in K} B_k$ holds, i.e., assuming t **stb**:

$$B \ \mathbf{shd} \ t \quad \text{iff} \quad (\forall k \in K : B_k \ \mathbf{shd}^1 \ t) \wedge (\forall k \in K : B_k \ \mathbf{shd} \ t \vee B_k \ \mathbf{stb}) \wedge (\exists k \in K : B_k \ \mathbf{shd} \ t) \quad .$$

if. Assume $B \parallel t \stackrel{w}{\Longrightarrow} B'$. If $w \in \mathbf{A}^+$ then $B_k \| \stackrel{w}{\Longrightarrow} B'$ for some $k \in K$, hence $\exists v \in \mathbf{A}^* : B' \stackrel{v\checkmark}{\Longrightarrow}$ by B_k $\mathbf{shd}^1$ t. Otherwise one of the following cases holds:

- $B' = B \parallel t$. Since B_k **shd** t for some $k \in K$ it follows that $\exists v \in \mathbf{A}^* : B_k \parallel t \stackrel{v\checkmark}{\Longrightarrow}$ and hence $B' \stackrel{v\checkmark}{\Longrightarrow}$;

- $B' = B'' \parallel t$ where $B \stackrel{\tau}{\Longrightarrow} B''$. In that case $B_k \stackrel{\tau}{\Longrightarrow} B''$ for some $k \in K$, implying $\neg(B_k\ \mathbf{stb})$ and hence $B_k\ \mathbf{shd}\ t$. It follows that $B' \stackrel{v\surd}{\Longrightarrow}$.

only if. The proof obligation consists of three parts, which we prove separately.

- Assume $B_k \parallel t \stackrel{w}{\Longrightarrow} B'$ for some $k \in K$ and $w \in \mathbf{A}^+$. It follows that $B \parallel t \stackrel{w}{\Longrightarrow} B'$ and hence $B' \stackrel{v\surd}{\longrightarrow}$. We may conclude $B_k\ \mathbf{shd}^1\ t$.
- Assume $B_k \parallel t \stackrel{w}{\Longrightarrow} B'$ for some $k \in K$ and $w \in \mathbf{A}^*$, where $\neg(B_k\ \mathbf{stb})$. If $B' = B_k \parallel t$ then there is a transition $B_k \stackrel{\tau}{\longrightarrow} B'_k$; hence $B \parallel t \stackrel{\varepsilon}{\Longrightarrow} B'_k \parallel t$ and (due to $B\ \mathbf{shd}\ t$) $\exists v \in \mathbf{A}^* : B'_k \parallel t \stackrel{v\surd}{\Longrightarrow}$; thus also $B' \stackrel{v\surd}{\Longrightarrow}$. Otherwise it follows (due to $B\ \mathbf{shd}\ t$) that $B \parallel t \stackrel{w}{\Longrightarrow} B'$ and hence $\exists v \in \mathbf{A}^* : B' \stackrel{v\surd}{\longrightarrow}$. We may conclude $B_k\ \mathbf{shd}\ t$.
- Due to $B\ \mathbf{shd}\ t$ and $B \parallel t \stackrel{\varepsilon}{\Longrightarrow} B \parallel t$ we have $\exists v \in \mathbf{A}^* : B \parallel t \stackrel{v\surd}{\Longrightarrow}$. It follows that $B_k \parallel t \stackrel{v\surd}{\Longrightarrow}$ for some $k \in K$. If $\neg(B_k\ \mathbf{stb})$ then $B_k\ \mathbf{shd}\ t$ by the above. Otherwise let $B_k \parallel t \stackrel{\varepsilon}{\Longrightarrow} B'$; it follows that $B' = B_k \parallel t$, and hence $B' \stackrel{v\surd}{\Longrightarrow}$. Since in addition we have $B_k\ \mathbf{shd}^1\ t$, we may conclude $B_k\ \mathbf{shd}\ t$.

Using this characterisation, it is not difficult to show that $I \sqsubseteq_{\mathbf{shd}} S$. Assume $t\ \mathbf{stb}$ and $S\ \mathbf{shd}\ t$.

- For all $k \in K$, $S_k\ \mathbf{shd}^1\ t$, implying (since $I_k \sqsubseteq_{\mathbf{shd}^1} S_k$ by Lemma B.2) $I_k\ \mathbf{shd}^1\ t$;
- For all $k \in K$, $S_k\ \mathbf{shd}\ t$ or $S_k\ \mathbf{stb}$, implying (by $I_k \sqsubseteq^+_{\mathbf{shd}} S_k$) $I_k\ \mathbf{shd}\ t$ or $I_k\ \mathbf{stb}$;
- There exists $k \in K$ such that $S_k\ \mathbf{shd}\ t$, implying (by $I_k \sqsubseteq_{\mathbf{shd}} S_k$) $I_k\ \mathbf{shd}\ t$.

We may conclude $I\ \mathbf{shd}\ t$. Due to Lemma B.1, then, $I \sqsubseteq_{\mathbf{shd}} S$. □

Lemma 3.9 *$\sqsubseteq_{\mathbf{shd}}$ and $\sqsubseteq^+_{\mathbf{shd}}$ are pre-congruences for renaming.*

Proof. We show that $\sqsubseteq_{\mathbf{shd}}$ is a pre-congruence for renaming; since $\sqsubseteq_{\mathbf{stb}}$ is also clearly a pre-congruence (due to the fact that $\varphi(\alpha) = \tau$ iff $\alpha = \tau$) the result for $\sqsubseteq^+_{\mathbf{shd}}$ follows immediately.

The proof proceeds by constructing, from a given test t and renaming function φ, a test $t{\uparrow}\varphi^{-1}$ such that for an arbitrary term B

$$B[\varphi]\ \mathbf{shd}\ t \quad \text{iff} \quad B\ \mathbf{shd}\ t{\uparrow}\varphi^{-1}\ .$$

Essentially, the behaviour of $t{\uparrow}\varphi^{-1}$ is described by the operational rule

$$\frac{B \stackrel{\alpha}{\longrightarrow} B' \quad \alpha = \varphi(\beta)}{B{\uparrow}\varphi^{-1} \stackrel{\beta}{\longrightarrow} B'{\uparrow}\varphi^{-1}}$$

That is, "$-{\uparrow}\varphi^{-1}$" is very much like renaming according to "$-[\varphi^{-1}]$," except that φ^{-1} is in general not a function. If we assume for the moment that $\mathbf{L}\surd$ is extended with an operator with precisely this behaviour, then it is not difficult to show that indeed

$$B[\varphi]\ \mathbf{shd}\ t \quad \text{iff} \quad B\ \mathbf{shd}\ t{\uparrow}\varphi^{-1}$$

if Assume $B[\varphi] \parallel t \stackrel{w}{\Longrightarrow} B'$ for some $w \in \mathbf{A}^*$. It follows that $B' = B''[\varphi] \parallel t''$ such that $B \stackrel{w'}{\Longrightarrow} B''$ for some $w' \in \mathbf{A}^*$ such that $w = \varphi(w')$, and $t \stackrel{w}{\Longrightarrow} t''$. But then $t{\uparrow}\varphi^{-1} \stackrel{w'}{\Longrightarrow} t''{\uparrow}\varphi^{-1}$, and hence $B \parallel (t{\uparrow}\varphi^{-1}) \stackrel{w'}{\Longrightarrow} B'' \parallel (t''{\uparrow}\varphi^{-1})$. Due to $B\ \mathbf{shd}\ t{\uparrow}\varphi^{-1}$, it follows that $B'' \parallel (t''{\uparrow}\varphi^{-1}) \stackrel{v\surd}{\Longrightarrow}$ for some $v \in \mathbf{A}^*$, implying $B'' \stackrel{v}{\Longrightarrow}$ and $t''{\uparrow}\varphi^{-1} \stackrel{v\surd}{\Longrightarrow}$. But then $B''[\varphi] \stackrel{v'}{\Longrightarrow}$ and $t'' \stackrel{v'\surd}{\Longrightarrow}$ for $v' = \varphi(v)$, implying $B' \stackrel{v'\surd}{\Longrightarrow}$.

only if Analogous to the above.

However, we do not want to extend the language just in order to facilitate this test construction. Instead, we will define $t{\uparrow}\varphi^{-1}$ syntactically.

For the purpose of this construction, we need some auxiliary constructs. First, we assume (essentially) that $A_{\checkmark}$ is closed under Cartesian product. To be precise, we assume a product operation $\otimes$ over the universe of actions, which is an injective partial function $\otimes\colon \mathbf{A}_{\checkmark} \times \mathbf{A}_{\checkmark} \rightharpoonup \mathbf{A}_{\checkmark}$ such that $\alpha \otimes \beta$ is defined iff $\{\alpha, \beta\} \cap \{\tau, \checkmark\} = \emptyset$ or $\alpha = \beta$, and $\tau \otimes \tau = \tau$ and $\checkmark \otimes \checkmark = \checkmark$. Moreover, for $i = 1, 2$ we assume projection operations

$$\pi_i\colon \alpha \mapsto \begin{cases} \alpha_i & \text{if } \alpha = \alpha_1 \otimes \alpha_2 \\ \alpha & \text{otherwise.} \end{cases}$$

The inverse φ^{-1} is to be treated as a function from actions to sets of actions (since φ is in general non-injective); it is an inverse renaming in the following sense. In general, an *inverse renaming* is a function $\Phi\colon \mathbf{A}_{\checkmark} \to \mathcal{P}(\mathbf{A}_{\checkmark})$ such that $\Phi(\tau) = \{\tau\}$, $\Phi(\checkmark) = \{\checkmark\}$ and $\Phi(\alpha) \cap \Phi(\beta) = \emptyset$ if $\alpha \neq \beta$. Inverse renamings induce a mapping over $\mathbf{X}$, associating with each $X \in \mathbf{X}$ a variable $X_\Phi \in \mathbf{X}$ such that $\mathcal{S}_{X_\Phi} = \bigcup \Phi(\mathcal{S}(X))$.

To combine an inverse renaming and an (ordinary) renaming into a new inverse renaming, we define

$$\Phi;\psi\colon \alpha \mapsto \{\alpha \otimes \beta \mid \beta \in \Phi(\psi(\alpha))\}$$

Note that $\Phi;\psi$ is indeed an inverse renaming. If we had simply defined $\Psi;\psi$ as $\alpha \mapsto \Phi(\psi(\alpha))$ then this would not have been the case: one can have $\psi(\alpha) = \psi(\beta)$ and thus $\Phi(\psi(\alpha)) = \Phi(\psi(\beta))$ even though $\alpha \neq \beta$. As it is, we can construct $\Phi(\psi(\alpha))$ from $\Phi;\psi(\alpha)$ by applying π_2.

For arbitrary inverse renamings Φ and (test) terms $B \in \mathbf{L}_{\checkmark}$, let $B{\uparrow}\Phi$ be defined inductively by

$$\begin{array}{rcl}
(\alpha; B){\uparrow}\Phi & := & \sum_{\beta \in \Phi(\alpha)} \beta; (B{\uparrow}\Phi) \\
(\sum_{i \in I} B_i){\uparrow}\Phi & := & \sum_{i \in I} (B_i{\uparrow}\Phi) \\
(B_1 \parallel_A B_2){\uparrow}\Phi & := & (B_1{\uparrow}\Phi) \parallel_{\bigcup \Phi(A)} (B_2{\uparrow}\Phi) \\
B[\psi]{\uparrow}\Phi & := & (B{\uparrow}\Phi;\psi)[\pi_2] \\
(B/A){\uparrow}\Phi & := & (B{\uparrow}\Phi)/(\bigcup \Phi(A)) \\
X{\uparrow}\Phi & := & X_\Phi \\
(\mathrm{rec}_X\, \theta){\uparrow}\Phi & := & \mathrm{rec}_{X_\Phi}(\theta{\uparrow}\Phi) \quad \text{where } (\theta{\uparrow}\Phi) = \{Y_\Phi := \theta(Y){\uparrow}\Phi\}_{Y \in dom(\theta)} \ .
\end{array}$$

The following properties can then be shown to hold, essentially reflecting the operational rule given above:

1. $B \xrightarrow{\alpha} B'$ implies $B{\uparrow}\Phi \xrightarrow{\beta} B'{\uparrow}\Phi$ for all $\beta \in \Phi(\alpha)$;
2. $B{\uparrow}\Phi \xrightarrow{\beta} B'$ implies $B \xrightarrow{\alpha} B''$ such that $\beta \in \Phi(\alpha)$ and $B' = B''{\uparrow}\Phi$.

The proof of these properties is (naturally) by induction on the structure of B. We show the more interesting cases of the proof: synchronisation and renaming.

- Assume $B = C_1 \parallel_A C_2$. It follows that $B{\uparrow}\Phi = (C_1{\uparrow}\Phi) \parallel_{\bigcup \Phi(A)} (C_2{\uparrow}\Phi)$.

  1. $B \xrightarrow{\alpha} B'$ can be generated in either of three possible ways:
     - $\alpha \notin A$, $C_1 \xrightarrow{\alpha} C_1'$ and $B' = C_1' \parallel_A C_2$. By the induction hypothesis, it follows that $C_1{\uparrow}\Phi \xrightarrow{\beta} C_1'{\uparrow}\Phi$ for all $\beta \in \Phi(\alpha)$. Due to the properties of inverse renamings, $\beta \notin \bigcup \Phi(A)$; hence $B{\uparrow}\Phi \xrightarrow{\beta} C_1'{\uparrow}\Phi \parallel_{\bigcup \Phi(A)} C_2{\uparrow}\Phi$ $(= B'{\uparrow}\Phi)$.
     - $\alpha \notin A$, $C_2 \xrightarrow{\alpha} C_2'$ and $B' = C_1 \parallel_A C_2'$. Analogous to the above.
     - $\alpha \in A$, $C_i \xrightarrow{\alpha} C_i'$ for $i = 1, 2$ and $B' = C_1' \parallel_A C_2'$. By the induction hypothesis, it follows that $C_i{\uparrow}\Phi \xRightarrow{\beta} C_i'{\uparrow}\Phi$ for all $\beta \in \Phi(\alpha)$ and $i = 1, 2$. Since $\beta \in \bigcup \Phi(A)$, we then have $B{\uparrow}\Phi \xrightarrow{\beta} C_1'{\uparrow}\Phi \parallel_{\bigcup \Phi(A)} C_2'{\uparrow}\Phi = B'{\uparrow}\Phi$.
  2. $B{\uparrow}\Phi \xrightarrow{\beta} B'$ can be generated in either of three possible ways:
     - $\beta \notin \bigcup \Phi(A)$, $C_1{\uparrow}\Phi \xrightarrow{\beta} C_1'$ and $B' = C_1' \parallel_{\bigcup \Phi(A)} C_2{\uparrow}\Phi$. By the induction hypothesis, it follows that $C_1 \xrightarrow{\alpha} C_1''$ such that $\beta \in \Phi(\alpha)$ and $C_1' = C_1''{\uparrow}\Phi$. It follows that $\alpha \notin A$, hence $B \xrightarrow{\alpha} C_1'' \parallel_A C_2$; moreover, $B' = (C_1'' \parallel_A C_2){\uparrow}\Phi$.

– $\beta \notin \bigcup \Phi(A)$, $C_2{\uparrow}\Phi \xrightarrow{\beta} C_2'$ and $B' = C_1{\uparrow}\Phi \parallel_{\bigcup \Phi(A)} C_2'$. Analogous to the above.

– $\beta \in \bigcup \Phi(A)$, $C_i{\uparrow}\Phi \xrightarrow{\beta} C_i'$ for $i = 1, 2$ and $B' = C_1' \parallel_{\bigcup \Phi(A)} C_2'$. By the induction hypothesis, it follows that for $i = 1, 2$, $C_i \xrightarrow{\alpha_i} C_i''$ such that $\beta \in \Phi(\alpha_i)$ and $C_i' = C_i''{\uparrow}\Phi$. Due to the properties of inverse renamings, it follows that $\alpha_1 = \alpha_2 \in A$, hence $B \xrightarrow{\alpha_1} C_1'' \parallel_A C_2''$; moreover, $B' = (C_1'' \parallel_A C_2''){\uparrow}\Phi$.

- Assume $B = C[\psi]$. It follows that $B{\uparrow}\Phi = (C{\uparrow}\Phi;\psi)[\pi_2]$.

  1. $B \xrightarrow{\alpha} B'$ implies $C \xrightarrow{\delta} C'$ where $\alpha = \psi(\delta)$ and $B' = C'[\psi]$. By induction, we have that $C{\uparrow}\Phi;\psi \xrightarrow{\gamma} C'{\uparrow}\Phi;\psi$ for all $\gamma \in \Phi;\psi(\delta)$; i.e., for all $\gamma = \delta \otimes \beta$ with $\beta \in \Phi(\alpha)$. But then $B{\uparrow}\Phi \xrightarrow{\beta} (C'{\uparrow}\Phi;\psi)[\pi_2]$ for all $\beta \in \Phi(\alpha)$; moreover, $B'{\uparrow}\Phi = (C'{\uparrow}\Phi;\psi)[\pi_2]$.
  2. $B{\uparrow}\Phi \xrightarrow{\beta} B'$ implies $C{\uparrow}\Phi;\psi \xrightarrow{\gamma} C'$ where $\beta = \pi_2(\gamma)$ and $B' = C'[\pi_2]$. By induction, we have that $C \xrightarrow{\delta} C''$ such that $\gamma \in \Phi;\psi(\delta)$ and $C' = C''{\uparrow}\Phi;\psi$. Let $\alpha = \psi(\delta)$. It follows that $\beta \in \Phi(\alpha)$; furthermore, if $B'' = C''[\psi]$ then $B \xrightarrow{\alpha} B''$ and $B' = (C''{\uparrow}\Phi;\psi)[\pi_2] = B''{\uparrow}\Phi$.

□

Lemma 4.18 *If $\mathcal{F}_i \sqsubseteq_{\mathbf{T}} \mathcal{G}_i$ for $i = 1, 2$ with $\mathcal{A}(\mathcal{G}_1) \cap \mathcal{A}(\mathcal{G}_2) \subseteq A$, then $\mathcal{F}_1 \parallel_A \mathcal{F}_2 \sqsubseteq_{\mathbf{T}} \mathcal{G}_1 \parallel_A \mathcal{G}_2$.*

Proof. Note that $\mathcal{F}_i \sqsubseteq_{\mathbf{T}} \mathcal{G}_i$ implies $\mathcal{L}(\mathcal{F}_i) \subseteq \mathcal{L}(\mathcal{G}_i)$ and hence $\mathcal{A}(\mathcal{F}_i) \subseteq \mathcal{A}(\mathcal{G}_i)$; therefore $\mathcal{F}_1 \parallel_A \mathcal{F}_2$ is defined. By commutativity of $\parallel_A$, we may assume that $\mathcal{F}_2 = \mathcal{G}_2$. For understandability, we show the full proof for the case where $A = \mathbf{A}$, since there the restriction functions in the construction of the synchronised tree failures become the identity over $\mathbf{A}$ and hence disappear.

- First note that $\uparrow(V \cup W) = \uparrow V \cup \uparrow W$ and $u^{-1}(V \cup W) = u^{-1}V \cup u^{-1}W$. Furthermore, for $w \notin \uparrow W$ we have $w^{-1}{\uparrow}W = \uparrow(w^{-1}W)$.
- These properties imply the following law, which we will need below: $w \notin \uparrow W$ implies $w^{-1}(V \cup \uparrow W) \subseteq \uparrow(w^{-1}(V \cup W))$, due to

$$w^{-1}(V \cup \uparrow W) = w^{-1}V \cup w^{-1}{\uparrow}W \subseteq \uparrow(w^{-1}V) \cup \uparrow(w^{-1}W) = \uparrow(w^{-1}(V \cup W)) \ .$$

- Let $(v, V_1 \cup V_2) \in \mathcal{F}_1 \parallel \mathcal{F}_2$ be arbitrary, such that $(v, V_i) \in \mathcal{F}_i$ for $i = 1, 2$.
- We will shift some of the traces in the refusal set V_1 (which is refused by $\mathcal{F}_1$) to V_2 (which is refused by $\mathcal{F}_2$). For this purpose we define the following sets:

$$\begin{aligned} W &= \{w \mid (v\,w, w^{-1}V_2) \notin \mathcal{F}_2\} \\ V_1' &= V_1 \setminus \uparrow W \\ V_2' &= V_2 \cup \uparrow W \ . \end{aligned}$$

  It should be clear that $V_1' \cup V_2' \supseteq V_1 \cup V_2$. Furthermore, by the saturation properties of $\mathbf{M}$, clearly $(v, V_1') \in \mathcal{F}_1$. (Actually, also $(v, V_2') \in \mathcal{F}_2$, and hence $(v, V_1' \cup V_2') \in \mathcal{F}_1 \parallel \mathcal{F}_2$. In effect, this constitutes a new "explanation" of the tree failure $(v, V_1 \cup V_2) \in \mathcal{F}_1 \parallel \mathcal{F}_2$.)
- Due to $\mathcal{F}_1 \sqsubseteq_{\mathbf{T}} \mathcal{G}_1$, there is a $w \in \{\varepsilon\} \cup \downarrow V_1'$ such that $(v\,w, w^{-1}V_1') \in \mathcal{G}_1$. Note that $w \notin \uparrow W$ (in particular, since $\varepsilon \notin W$).
- From $w \notin W$ it follows that $(v\,w, w^{-1}V_2) \in \mathcal{F}_2$. By construction, $(v\,w\,u, u^{-1}w^{-1}V_2) \notin \mathcal{F}_2$ for all $u \in w^{-1}W$; by saturation, it follows that $(v\,w, w^{-1}(V_2 \cup W)) \in \mathcal{F}_2$.
- By $\mathcal{F}_2 = \mathcal{G}_2$, the saturation properties of $\mathbf{M}$ and the above law we have $(v\,w, w^{-1}V_2') \in \mathcal{G}_2$.
- By definition (Table 4.3) we have $(v\,w, w^{-1}(V_1' \cup V_2')) \in \mathcal{G}_1 \parallel \mathcal{G}_2$.
- By $V_1' \subseteq V_1$ we have found a $w \in \{\varepsilon\} \cup \downarrow(V_1 \cup V_2)$ with (due to $V_1' \cup V_2' \supseteq V_1 \cup V_2$ and saturation) $(v\,w, w^{-1}(V_1 \cup V_2)) \in \mathcal{G}_1 \parallel \mathcal{G}_2$.

Now we show the (entirely analogous) proof for the general case, without comment. Let $\varphi_1 = \pi_{\overline{\mathcal{A}(\mathcal{G}_2)\setminus A}}$ and $\varphi_2 = \pi_{\overline{\mathcal{A}(\mathcal{G}_1)\setminus A}}$. We implicitly use some algebraic properties of our string homomorphisms, for instance $\varphi(V \cup W) = \varphi(V) \cup \varphi(W)$, $\varphi(\downarrow V) = \downarrow\varphi(V)$, $\varphi(\uparrow V) \subseteq \uparrow\varphi(V)$ and $\varphi(w^{-1}V) \subseteq \varphi(w)^{-1}\varphi(V)$.

- Let $(v, V_1 \cup V_2) \in \mathcal{F}_1 \parallel_A \mathcal{F}_2$ be arbitrary, such that $(\varphi_i(v), \varphi_i(V_i)) \in \mathcal{F}_i$ for $i = 1, 2$.
- Let $W = \{w \mid (\varphi_2(v\,w), \varphi_2(w^{-1}V_2)) \notin \mathcal{F}_2\}$, and let $V_1' = V_1 \setminus \uparrow W$ and $V_2' = V_2 \cup \uparrow W$. It follows that $(\varphi_1(v), \varphi_1(V_1')) \in \mathcal{F}_1$.
- Due to $\mathcal{F}_1 \sqsubseteq_{\mathbf{T}} \mathcal{G}_1$, there is a $w \in \{\varepsilon\} \cup \downarrow V_1'$ such that $(\varphi_1(v\,w), \varphi_1(w)^{-1}\varphi_1(V_1')) \in \mathcal{G}_1$; hence (by $\varphi_1(w^{-1}V_1') \subseteq \varphi_1(w)^{-1}\varphi_1(V_1')$ and saturation) $(\varphi_1(v\,w), \varphi_1(w^{-1}V_1')) \in \mathcal{G}_1$. Again, $w \notin \uparrow W$.
- From $w \notin W$ it follows that $(\varphi_2(v\,w), \varphi_2(w^{-1}V_2)) \in \mathcal{F}_2$. For all $u \in w^{-1}W$, we have by construction that $(\varphi_2(v\,w\,u), \varphi_2((wu)^{-1}V_2)) \notin \mathcal{F}_2$, hence $(\varphi_2(v\,w\,u), \varphi_2(u)^{-1}\varphi_2(w^{-1}V_2)) \notin \mathcal{F}_2$. By saturation, this implies $(\varphi_2(v\,w), \varphi_2(w^{-1}(V_2 \cup W))) \in \mathcal{F}_2$.
- By $\mathcal{F}_2 = \mathcal{G}_2$ and the saturation properties of $\mathbf{M}$, we get $(\varphi_2(v\,w), \varphi_2(\uparrow(w^{-1}(V_2 \cup W)))) \in \mathcal{G}_2$. With the above law, we have $(\varphi_2(v\,w), \varphi_2(w^{-1}V_2')) \in \mathcal{G}_2$.
- The rest of the proof is exactly as in the special case above.

□

Theorem 5.5 *For a fixed finite alphabet size, $T \sqsubseteq_{\mathbf{shd}} U$ and $T \sqsubseteq^c_{\mathbf{shd}} U$ can be decided in time linear exponential in the numbers of states of T and U.*

Proof. First note that language inclusion (or equality) has to be checked first, which can be done within the given time bound. We have to determine the time complexity of our improved algorithm. Let T have n and U have m states. We assume that T and U are given by adjacency matrices for each $\alpha \in \mathbf{A}_\tau$, i.e., T and U have sizes $O(n^2)$ and $O(m^2)$.

Making U deterministic as described above costs $O(m^2 2^m)$, in fact we can construct the complete powerset automaton in this time: first, we determine the transitive closure of the τ-edges in $O(m^3)$, which is subsumed by the complexity of the next step. Given a state Q among the 2^m many subsets of the state set of U and $a \in \mathbf{A}$, we build Q' with $Q \xrightarrow{a}_{BB} Q'$ by constructing $Q'' = \{q' \mid \exists q \in Q\colon q \xrightarrow{a}_U q'\}$ in $O(m^2)$ and then constructing Q' from Q'' in $O(m^2)$ using the transitive closure. We can store the complete powerset automaton as a $2^m \times |\mathbf{A}|$-matrix where Q'' is the entry for Q and a.

BB consists of the nonempty states reachable from Q_0 as above. The complete deterministic powerset automaton can also serve for describing all associated automata; in this role, the empty state $\emptyset$ is the only final state and the automata in BB_Q are obtained by taking the $\{q\}$ with $q \in Q$ as initial states. Compare our running example and Figure 5.2.

Similarly, we construct the complete powerset automaton for T to have a description of the associated automata for AA. Thus, building AA and BB costs $O(n^2 2^n + m^2 2^m)$. From S, we only need the states; building these takes time $O(n^2 2^m)$, since AA has n states having $O(n)$ edges each, while BB has $O(2^m)$ states having $O(1)$ edges each. S has $O(n2^m)$ states.

For each state (p, Q) of S we do the following: We construct P from two deterministic automata with $O(2^n)$ and $O(2^m)$ states; this takes time $O(2^{n+m})$ and results in as many states. Searching for a suitable (p', Q') in P or any productive sub-automaton R, we check for all of the $O(2^{n+m})$ possible (p', Q') and all of the $O(m)$ associated $B \in BB_{Q'}$ whether $\mathcal{L}_R(p', Q') \subseteq \mathcal{L}(B)$. R and B are deterministic of sizes $O(2^{n+m})$ and $O(2^m)$; hence, this inclusion is checked by building yet another product, which takes time $O(2^{n+2m})$. Hence, treating R takes time $O(2^{n+m})O(m)O(2^{n+2m}) = O(m2^{2n+3m})$. (Building the new productive sub-automaton is a simple graph-theoretic reachability analysis and takes time in the order of the number of edges, i.e. $O(2^{n+m})$.)

Thus, treating (p, Q), we build P with $O(2^{n+m})$ states and check as many productive subautomata; this takes time $O(m2^{3n+4m})$. Since S has $O(n2^m)$ states, for deciding $\sqsubseteq_{\mathbf{shd}}$ we need altogether time $O(n^2 2^n + m^2 2^m) + O(n^2 2^m) + O(n2^m)O(m2^{3n+4m}) = O(nm2^{3n+5m})$. Deciding $\sqsubseteq^c_{\mathbf{shd}}$ additionally involves deciding inverse stability implication and inverse language inclusion; both can be done within the given time bound. □

Lemma 6.2 *Let $T = \langle S, \to, q\rangle$ be a transition system with states $s' \neq q$ and s that satisfy the following conditions (see Figure 6.1):*

a) $s \xrightarrow{\alpha} s'$ *iff* $\alpha = \tau$;

b) for all $\alpha \neq \tau$ *we have* $s \xrightarrow{\alpha} s$ *iff* $s' \xrightarrow{\alpha} s'$;

c) if $s \xrightarrow{\alpha} t$ *for some* $t \in S \setminus \{s, s'\}$, *then* $s' \xrightarrow{\alpha} t'$ *for some* $t' \in S \setminus \{s'\}$;

d) if $t \xrightarrow{\alpha} s'$ *for some* $t \in S \setminus \{s, s'\}$, *then* $t \xrightarrow{\alpha} s$, *too.*

Let $U = \langle S', \to, q\rangle$ be obtained from T by contracting s and s' to s, i.e., by putting $S' = S \setminus \{s'\}$ and replacing s' in the arcs by s. Then $T \simeq^{c}_{\mathbf{shd}} U$.

Proof. Initial stability is not modified by the contraction, hence it suffices to check $T \simeq_{\mathbf{shd}} U$ (which includes language equality) using our denotational characterization. We proceed in several steps.

i) For all $t, t' \in S - \{s'\}$, we have $t \overset{w}{\Longrightarrow} t'$ in T iff $t \overset{w}{\Longrightarrow} t'$ in U.

Proof. Paths in T can be translated to paths in U easily, replacing s' by s. To translate a path in U to one in T, we consider several cases. On such a path, an arc that does not start in s exists in T as well. A loop $s \xrightarrow{\tau} s$ can be ignored. A loop $s \xrightarrow{\alpha} s$ with $\alpha \neq \tau$ either exists in T as well; or it arises from a loop $s' \xrightarrow{\alpha} s'$, but then it exists in T as well by *b)*; or by *a)* it arises from an arc $s' \xrightarrow{\alpha} s$, in which case it can be translated to $s \xrightarrow{\tau} s' \xrightarrow{\alpha} s$. Finally, an arc $s \xrightarrow{\alpha} t''$ with $t'' \neq s$ either exists in T or can be translated to $s \xrightarrow{\tau} s' \xrightarrow{\alpha} t''$.

ii) For all $t \in S - \{s'\}$, we have that $t \overset{w}{\Longrightarrow} s'$ in T implies $t \overset{w}{\Longrightarrow} s$ in T.

Proof. Very similar to the second part of the above proof, also using *d)*.

iii) Let $V \subseteq \mathbf{A}^*$ and $u \in \mathbf{A}^*$ be given such that in U $q \overset{u}{\Longrightarrow} t$ and $\nexists v \in V.\, t \overset{v}{\Longrightarrow}$. Then the same holds in T.

Proof. The proof follows directly from *i)* and *ii)*, since $t \neq s'$.

iv) Let $V \subseteq \mathbf{A}^*$ and $u \in \mathbf{A}^*$ be given such that in T $q \overset{u}{\Longrightarrow} t$ with $t \neq s'$ and $\nexists v \in V.\, t \overset{v}{\Longrightarrow}$. Then the same holds in U.

Proof. The proof follows directly from *i)*.

v) Let $V \subseteq \mathbf{A}^*$ and $u \in \mathbf{A}^*$ be given such that in T $q \overset{u}{\Longrightarrow} s'$ and $\nexists v \in V.\, s' \overset{v}{\Longrightarrow}$. Then there exists some $t \neq s'$ and some $w \in {\downarrow}V$ such that in T $q \overset{uw}{\Longrightarrow} t$ and $\nexists v \in w^{-1}V.\, t \overset{v}{\Longrightarrow}$.

Proof. This is shown in two steps:

α) If for some $w \in {\downarrow}V$ we have $s' \overset{w}{\Longrightarrow} t$ with some $t \neq s'$, then we can choose this t and w.

β) Otherwise we choose $t = s$ and $w = \varepsilon$. By *ii)* we have $q \overset{u}{\Longrightarrow} s$. If we had $s \overset{v}{\Longrightarrow}$ for some $v \in w^{-1}V = V$, then consider the corresponding path. If this path consists of loops at s only, we find the same loops at s' by *b)*, which gives the contradiction $s' \overset{v}{\Longrightarrow}$. If the first state $\neq s$ on the path is s', then the path starts with loops at s and the arc $s \xrightarrow{\tau} s'$ by *a)*, we find the same loops at s' by *b)*, and again get the contradiction $s' \overset{v}{\Longrightarrow}$. Finally, if the first state $\neq s$ on the path is some $t'' \neq s'$ reached with the arc $s \xrightarrow{\alpha} t''$, then we find the same loops at s' by *b)* and an arc $s' \xrightarrow{\alpha} t'$ by *d)*, hence we are in case *α)*, which is a contradiction.

We now conclude from *i)* and *ii)* that T and U have the same language; *iii)* shows $\mathcal{F}^+(U) \subseteq \mathcal{F}^+(T)$, and *iv)* and *v)* show $T \sqsubseteq_{\mathcal{F}^+} U$. □