The Linear Time - Branching Time Spectrum I*
The Semantics of Concrete, Sequential Processes

R.J. van Glabbeek
Computer Science Department, Stanford University
Stanford, CA 94305-9045, USA
`rvg@cs.stanford.edu`
`http://theory.stanford.edu/~rvg`

In this paper various semantics in the linear time – branching time spectrum are presented in a uniform, model-independent way. Restricted to the class of finitely branching, concrete, sequential processes, only fifteen of them turn out to be different, and most semantics found in the literature that can be defined uniformly in terms of action relations coincide with one of these fifteen. Several testing scenarios, motivating these semantics, are presented, phrased in terms of 'button pushing experiments' on generative and reactive machines. Finally twelve of these semantics are applied to a simple language for finite, concrete, sequential, nondeterministic processes, and for each of them a complete axiomatization is provided.

Keywords: Concurrency, Labelled transition systems, Nondeterminism, Semantic equivalences, Linear time, Branching time, Generative and reactive systems, Modal logic, Trace semantics, Failures semantics, Failure trace, Ready trace, Simulation, Ready simulation, Bisimulation, Complete axiomatizations.

Table of contents

*This is an extension of [20]. The research reported in this paper has been initiated at CWI in Amsterdam, continued at the Technical University of Munich, and finalized at Stanford University. It has been supported by Sonderforschungsbereich 342 of the TU München and by ONR under grant number N00014-92-J-1974. Part of it was carried out in the preparation of a course Comparative Concurrency Semantics, given at the University of Amsterdam, spring 1988. A coloured version of this paper is available at `http://boole.stanford.edu/pub/spectrum1.ps.gz`.

Introduction

Process theory A *process* is the behaviour of a system. The system can be a machine, an elementary particle, a communication protocol, a network of falling dominoes, a chess player, or any other system. *Process theory* is the study of processes. Two main activities of process theory are *modelling* and *verification*. Modelling is the activity of representing processes, mostly by mathematical structures or by expressions in a system description language. Verification is the activity of proving statements about processes, for instance that the actual behaviour of a system is equal to its intended behaviour. Of course, this is only possible if a criterion has been defined, determining whether or not two processes are equal, i.e. two systems behave similarly. Such a criterion constitutes the *semantics* of a process theory. (To be precise, it constitutes the semantics of the equality concept employed in a process theory.) Which aspects of the behaviour of a system are of importance to a certain user depends on the environment in which the system will be running, and on the interests of the particular user. Therefore it is not a task of process theory to find the 'true' semantics of processes, but rather to determine which process semantics is suitable for which applications.

Comparative concurrency semantics This paper aims at the classification of process semantics.[1] The set of possible process semantics can be partially ordered by the relation 'makes strictly more identifications on processes than', thereby becoming a complete lattice[3]. Now the classification of some useful process semantics can be facilitated by drawing parts of this lattice and locating the positions of some interesting process semantics, found in the literature. Furthermore the ideas involved in the construction of these semantics can be unravelled and combined in new compositions, thereby creating an abundance of new process semantics. These semantics will, by their intermediate positions in the semantic lattice, shed light on the differences and similarities of the established ones. Sometimes they also turn out to be interesting in their own right. Finally the semantic lattice serves as a map on which it can be indicated which semantics satisfy certain desirable properties, and are suited for a particular class of applications.

Most semantic notions encountered in contemporary process theory can be classified along four different lines, corresponding with four different kinds of identifications. First there is the dichotomy of linear time versus branching time: to what extent should one identify processes differing only in the branching structure of their execution paths? Secondly there is the dichotomy of interleaving semantics versus partial order semantics: to what extent should one identify processes differing only in the causal dependencies between their actions (while agreeing on the possible orders of execution)? Thirdly one encounters different treatments of abstraction from internal actions in a process: to what extent should one identify processes differing only in their internal or silent actions? And fourthly there are different approaches to infinity: to what extent should one identify processes differing only in their infinite behaviour? These considerations give rise to a four dimensional representation of the proposed semantic lattice.

[1]This field of research is called *comparative concurrency*[2] *semantics*, a terminology first used by Meyer in [36].

[2]Here *concurrency* is taken to be synonymous with process theory, although strictly speaking it is only the study of *parallel* (as opposed to *sequential*) processes. These are the behaviours of systems capable of performing different actions at the same time. In this paper the term concurrency is considered to include sequential process theory. This may be justified since much work on sequential processes is intended to facilitate later studies involving parallelism.

[3]The supremum of a set of process semantics is the semantics identifying two processes whenever they are identified by every semantics in this set.

However, at least three more dimensions can be distinguished. In this paper, stochastic and real-time aspects of processes are completely neglected. Furthermore it deals with *uniform concurrency*[4] only. This means that processes are studied, performing actions[5] $a, b, c, \ldots$ which are not subject to further investigations. So it remains unspecified if these actions are in fact assignments to variables or the falling of dominoes or other actions. If also the options are considered of modelling (to a certain degree) the stochastic and real-time aspects of processes and the operational behaviour of the elementary actions, three more parameters in the classification emerge.

Process domains In order to be able to reason about processes in a mathematical way, it is common practice to represent processes as elements of a mathematical domain[6]. Such a domain is called a *process domain*. The relation between the domain and the world of real processes is mostly stated informally. The semantics of a process theory can be modelled as an equivalence on a process domain, called a *semantic equivalence*. In the literature one finds among others:

- *graph domains*, in which a process is represented as a *process graph*, or *state transition diagram*,
- *net domains*, in which a process is represented as a (labelled) *Petri net*,
- *event structure domains*, in which a process is represented as a (labelled) *event structure*,
- *explicit domains*, where a process is represented as a mathematically coded set of its properties,
- *projective limit domains*, which are obtained as projective limits of series of finite term domains,
- and *term domains*, in which a process is represented as a term in a system description language.

Action relations Write $p \xrightarrow{a} q$ if the process p can evolve into the process q, while performing the action a. The binary predicates $\xrightarrow{a}$ are called *action relations*. The semantic equivalences which are treated in this paper will be defined entirely in terms of action relations. Hence these definitions apply to any process domain on which action relations are defined. Such a domain is called a *labelled transition system*. Furthermore they will be defined *uniformly* in terms of action relations, meaning that all actions are treated in the same way. For reasons of convenience, even the usual distinction between internal and external actions is dropped in this paper.

Finitely branching, concrete, sequential processes Being a first step, this paper limits itself to a very simple class of processes. First of all only *sequential* processes are investigated: processes capable of performing at most one action at a time. Furthermore, instead of dropping the usual distinction between internal and external actions, one can equivalently maintain to study *concrete* processes: processes in which no internal actions occur. For this simple class of processes the announced semantic lattice collapses in two out of four dimensions and covers only the *infinitary linear time – branching time spectrum*.

Moreover, the main interest is in *finitely branching* processes: processes having in each state only finitely many possible ways to proceed. The material pertaining to infinitely branching processes—coloured brown in the electronic version of this paper—can easily be omitted in first reading.

[4]The term uniform concurrency is employed by DE BAKKER ET AL [8].

[5]Strictly speaking processes do not perform actions, but systems do. However, for reasons of convenience, this paper sometimes uses the word process, when actually referring to a system of which the process is the behaviour.

[6]I use the word *domain* in the sense of *universal algebra*; it can be any class of mathematical objects—typically the first component of an *algebra*; the other component being a collection of operators defined on this domain. Without further adjectives I do not refer to the more restrictive domains employed in *domain theory*.

Literature In the literature on uniform concurrency 12 semantics can be found, which are uniformly definable in terms of action relations and different on the domain of finitely branching, sequential processes (see Figure 1). The coarsest one (i.e. the semantics making the most identifi-

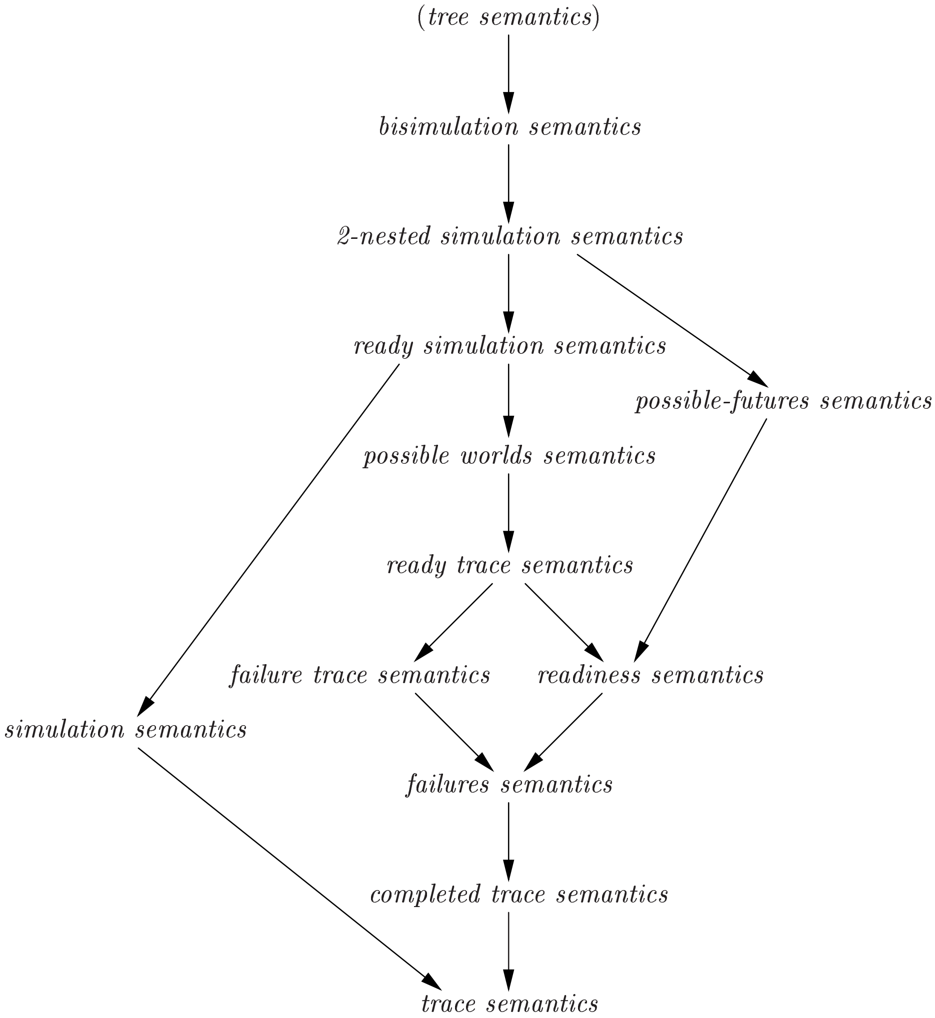


Figure 1: The linear time – branching time spectrum

cations) is *trace semantics*, as presented in Hoare [30]. In trace semantics only *partial traces* are employed. The finest one (making less identifications than any of the others) is *bisimulation semantics*, as presented in Milner [39]. Bisimulation semantics is the standard semantics for the system description language CCS (Milner [37]). The notion of bisimulation was introduced in Park [41]. Bisimulation equivalence is a refinement of *observational equivalence*, as introduced by Hennessy & Milner in [27]. On the domain of finitely branching, concrete, sequential processes, both equivalences coincide. Also the semantics of De Bakker & Zucker, presented in [9], coincides with bisimulation semantics on this domain. Then there are ten semantics in between. First of all a variant of trace semantics can be obtained by using *complete traces* besides partial ones. In this paper it is called *completed trace semantics*. *Failures semantics* is introduced in Brookes, Hoare & Roscoe [13], and used in the construction of a model for the system description language CSP

(Hoare [29, 31]). It is finer than completed trace semantics. The semantics based on *testing equivalences*, as developed in De Nicola & Hennessy [17], coincides with failures semantics on the domain of finitely branching, concrete, sequential processes, as do the semantics of Kennaway [34] and Darondeau [15]. This has been established in De Nicola [16]. In Olderog & Hoare [40] *readiness semantics* is presented, which is slightly finer than failures semantics. Between readiness and bisimulation semantics one finds *ready trace semantics*, as introduced independently in Pnueli [43] (there called *barbed semantics*), Baeten, Bergstra & Klop [6] and Pomello [44] (under the name *exhibited behaviour semantics*). The natural completion of the square, suggested by failures, readiness and ready trace semantics yields *failure trace semantics*. For finitely branching processes this is the same as *refusal semantics*, introduced in Phillips [42]. *Simulation semantics*, based on the classical notion of *simulation* (see e.g. Park [41]), is independent of the last five semantics. *Ready simulation semantics* was introduced in Bloom, Istrail & Meyer [12] under the name *GSOS trace congruence*. It is finer than ready trace as well as simulation semantics. In Larsen & Skou [35] a more operational characterization of this equivalence was given under the name $\frac{2}{3}$*-bisimulation equivalence*. The (denotational) notion of *possible worlds semantics* of Veglioni & De Nicola [49] fits between ready trace and ready simulation semantics. Finally *2-nested simulation semantics*, introduced in Groote & Vaandrager [25], is located between ready simulation and bisimulation semantics, and *possible-futures semantics*, as proposed in Rounds & Brookes [46], can be positioned between 2-nested simulation and readiness semantics.

Tree semantics, employed in Winskel [50], is even finer than bisimulation semantics. However, a proper treatment requires more than mere action relations.

About the contents The first section of this paper introduces labelled transition systems and process graphs. A labelled transition system is any process domain that is equipped with action relations. The domain of *process graphs* or *state transition diagrams* is one of the most popular labelled transition systems. In Sections 2–14 all semantic equivalences mentioned above are defined on arbitrary labelled transition systems. In particular these definitions apply to the domain of process graphs. Most of the equivalences can be motivated by the observable behaviour of processes, according to some testing scenario. (Two processes are equivalent if they allow the same set of possible observations, possibly in response to certain experiments.) I will try to capture these motivations in terms of *button pushing experiments* (cf. Milner [37], pp. 10-12). Furthermore the semantics will be partially ordered by the relation 'makes at least as many identifications as'. This yields the linear time – branching time spectrum. Counterexamples are provided, showing that on the graph domain this ordering cannot be further expanded. However, for deterministic processes the spectrum collapses, as was first observed by Park [41]. Section 16 describes various other classes of processes on which parts of the spectrum collapse. In Section 17, the semantics are applied to a simple language for finite, concrete, sequential, nondeterministic processes, and for twelve of them a complete axiomatization is provided. Section 18 applies a few criteria indicating which semantics are suitable for which applications. Finally, in Section 19 the work of this paper is extended to labelled transition systems that distinguish between deadlock and successful termination.

With each of the semantic equivalences treated in this paper (except for tree semantics) a preorder is associated that may serve as an implementation relation between processes. The results obtained for the equivalences are extended to the associated preorders as well.

Acknowledgment My thanks to Tony Hoare for suggesting that the axioms of Table 2 could be simplified along the lines of Table 5.

1 Labelled transition systems and process graphs

1.1 Labelled transition systems

In this paper processes will be investigated that are capable of performing actions from a given set Act. By an *action* any activity is understood that is considered as a conceptual entity on a chosen level of abstraction. Actions may be instantaneous or durational and are not required to terminate, but in a finite time only finitely many actions can be carried out. Any activity of an investigated process should be part of some action $a \in Act$ performed by the process. Different activities that are indistinguishable on the chosen level of abstraction are interpreted as occurrences of the same action $a \in Act$.

A process is *sequential* if it can perform at most one action at the same time. In this paper only sequential processes will be considered. A class of sequential processes can often be conveniently represented as a labelled transition system. This is a domain $\mathbb{P}$ on which infix written binary predicates $\xrightarrow{a}$ are defined for each action $a \in Act$. The elements of $\mathbb{P}$ represent processes, and $p \xrightarrow{a} q$ means that p can start performing the action a and after completion of this action reach a state where q is its remaining behaviour. In a labelled transition system it may happen that $p \xrightarrow{a} q$ and $p \xrightarrow{b} r$ for different actions a and b or different processes q and r. This phenomenon is called *branching*. It need not be specified how the choice between the alternatives is made, or whether a probability distribution can be attached to it.

Certain actions may be synchronizations of a process with its environment, or the receipt of a signal sent by the environment. Naturally, these actions can only occur if the environment cooperates. In the labelled transition system representation of processes all these potential actions are included, so $p \xrightarrow{a} q$ merely means that there is an environment in which the action a can occur.

Notation: For any alphabet Σ, let Σ^* be the set of finite sequences and Σ^∞ the set of infinite sequences over Σ. $\Sigma^\omega := \Sigma^* \cup \Sigma^\infty$. Write ε for the empty sequence, $\sigma\rho$ for the concatenation of $\sigma \in \Sigma^*$ and $\rho \in \Sigma^\omega$, and a for the sequence consisting of the single symbol $a \in \Sigma$.

Definition 1.1 A *labelled transition system* is a pair $(\mathbb{P}, \rightarrow)$ with $\mathbb{P}$ a class and $\rightarrow \subseteq \mathbb{P} \times Act \times \mathbb{P}$, such that for $p \in \mathbb{P}$ and $a \in Act$ the class $\{q \in \mathbb{P} \mid (p,a,q) \in \rightarrow\}$ is a set.

Most of this paper should be read in the context of a given labelled transition system $(\mathbb{P}, \rightarrow)$, ranged over by $p, q, r, \ldots$. Write $p \xrightarrow{a} q$ for $(p,a,q) \in \rightarrow$. The binary predicates $\xrightarrow{a}$ are called *action relations*.

Definition 1.2 (Remark that the following concepts are defined in terms of action relations only)

- The *generalized action relations* $\xrightarrow{\sigma}$ for $\sigma \in Act^*$ are defined recursively by:
  1. $p \xrightarrow{\varepsilon} p$, for any process p.
  2. $(p,a,q) \in \rightarrow$ with $a \in Act$ implies $p \xrightarrow{a} q$ with $a \in Act^*$.
  3. $p \xrightarrow{\sigma} q \xrightarrow{\rho} r$ implies $p \xrightarrow{\sigma\rho} r$.

  In words: the generalized action relations $\xrightarrow{\sigma}$ are the reflexive and transitive closure of the ordinary action relations $\xrightarrow{a}$. $p \xrightarrow{\sigma} q$ means that p can evolve into q, while performing the sequence σ of actions. Remark that the overloading of the notion $p \xrightarrow{a} q$ is quite harmless.
- A process $q \in \mathbb{P}$ is *reachable* from $p \in \mathbb{P}$ if $p \xrightarrow{\sigma} q$ for some $\sigma \in Act^*$.

- The set of *initial actions* of a process p is defined by: $I(p) = \{a \in Act \mid \exists q : p \stackrel{a}{\longrightarrow} q\}$.
- A process $p \in \mathbb{P}$ is *finite* if the set $\{(\sigma, q) \in (Act^* \times \mathbb{P}) \mid p \stackrel{\sigma}{\longrightarrow} q\}$ is finite.
- p is *image finite* if for each $\sigma \in Act^*$ the set $\{q \in \mathbb{P} \mid p \stackrel{\sigma}{\longrightarrow} q\}$ is finite.
- p is *deterministic* if $p \stackrel{\sigma}{\longrightarrow} q \wedge p \stackrel{\sigma}{\longrightarrow} r \Rightarrow q = r$.
- p is *well-founded* if there is no infinite sequence $p \stackrel{a_1}{\longrightarrow} p_1 \stackrel{a_2}{\longrightarrow} p_2 \stackrel{a_3}{\longrightarrow} \cdots$.
- p is *finitely branching* if for each q reachable from p, the set $\{(a, r) \in Act \times \mathbb{P} \mid q \stackrel{a}{\longrightarrow} r\}$ is finite.

Note that a process $p \in \mathbb{P}$ is image finite iff for each $q \in \mathbb{P}$ reachable from p and each $a \in Act$, the set $\{r \in \mathbb{P} \mid q \stackrel{a}{\longrightarrow} r\}$ is finite. Hence finitely branching processes are image finite. Moreover, by König's lemma a process is finite iff it is well-founded and finitely branching.

1.2 Process graphs

Definition 1.3 A *process graph* over an alphabet Act is a rooted, directed graph whose edges are labelled by elements of Act. Formally, a process graph g is a triple $(\text{NODES}(g), \text{ROOT}(g), \text{EDGES}(g))$, where

- $\text{NODES}(g)$ is a set, of which the elements are called the *nodes* or *states* of g,
- $\text{ROOT}(g) \in \text{NODES}(g)$ is a special node: the *root* or *initial state* of g,
- and $\text{EDGES}(g) \subseteq \text{NODES}(g) \times Act \times \text{NODES}(g)$ is a set of triples (s, a, t) with $s, t \in \text{NODES}(g)$ and $a \in Act$: the *edges* or *transitions* of g.

If $e = (s, a, t) \in \text{EDGES}(g)$, one says that *$e$ goes from s to t*. A (finite) *path* π in a process graph is an alternating sequence of nodes and edges, starting and ending with a node, such that each edge goes from the node before it to the node after it. If $\pi = s_0(s_0, a_1, s_1)s_1(s_1, a_2, s_2) \cdots (s_{n-1}, a_n, s_n)s_n$, also denoted as $\pi : s_0 \stackrel{a_1}{\longrightarrow} s_1 \stackrel{a_2}{\longrightarrow} \cdots \stackrel{a_n}{\longrightarrow} s_n$, one says that *$\pi$ goes from s_0 to s_n*; it *starts* in s_0 and *ends* in $end(\pi) = s_n$. Let $\text{PATHS}(g)$ be the set of paths in g starting from the root. If s and t are nodes in a process graph then *t can be reached from s* if there is a path going from s to t. A process graph is said to be *connected* if all its nodes can be reached from the root; it is a *phrase-tree* if each node can be reached from the root by exactly one path. Let $\mathbb{G}$ be the domain of connected process graphs over a given alphabet Act.

Definition 1.4 Let $g, h \in \mathbb{G}$. A *graph isomorphism* between g and h is a bijective function $f : \text{NODES}(g) \rightarrow \text{NODES}(h)$ satisfying

- $f(\text{ROOT}(g)) = \text{ROOT}(g)$ and
- $(s, a, t) \in \text{EDGES}(g) \quad \Leftrightarrow \quad (f(s), a, f(t)) \in \text{EDGES}(h)$.

Graphs g and h are *isomorphic*, notation $g \cong h$, if there exists a graph isomorphism between them.

In this case g and h differ only in the identity of their nodes. Remark that graph isomorphism is an equivalence relation on $\mathbb{G}$.

Connected process graphs can be pictured by using open dots ($\circ$) to denote nodes, and labelled arrows to denote edges, as can be seen further on. There is no need to mark the root of such a process graph if it can be recognized as the unique node without incoming edges, as is the case in all my examples. These pictures determine process graphs only up to graph isomorphism, but usually this suffices since it is virtually never needed to distinguish between isomorphic graphs.

Definition 1.5 For $g \in \mathbb{G}$ and $s \in \text{NODES}(g)$, let g_s be the process graph defined by

- $\text{NODES}(g_s) = \{t \in \text{NODES}(g) \mid \text{there is a path going from } s \text{ to } t\}$,
- $\text{ROOT}(g_s) = s \in \text{NODES}(g_s)$,
- and $(t, a, u) \in \text{EDGES}(g_s)$ iff $t, u \in \text{NODES}(g_s)$ and $(t, a, u) \in \text{EDGES}(g)$.

Of course $g_s \in \mathbb{G}$. Note that $g_{\text{ROOT}(g)} = g$. Now on $\mathbb{G}$ action relations $\xrightarrow{a}$ for $a \in Act$ are defined by $g \xrightarrow{a} h$ iff $(\text{ROOT}(g), a, s) \in \text{EDGES}(g)$ and $h = g_s$. This makes $\mathbb{G}$ into a labelled transition system.

1.3 Embedding labelled transition systems in $\mathbb{G}$

Let $(\mathbb{P}, \rightarrow)$ be an arbitrary labelled transition system and let $p \in \mathbb{P}$. The *canonical graph* $G(p)$ of p is defined as follows:

- $\text{NODES}(G(p)) = \{q \in \mathbb{P} \mid \exists \sigma \in Act^* : \ p \xrightarrow{\sigma} q\}$,
- $\text{ROOT}(G(p)) = p \in \text{NODES}(G(p))$,
- and $(q, a, r) \in \text{EDGES}(G(p))$ iff $q, r \in \text{NODES}(G(p))$ and $q \xrightarrow{a} r$.

Of course $G(p) \in \mathbb{G}$. This means G is a function from $\mathbb{P}$ to $\mathbb{G}$.

Proposition 1.1 $G : \mathbb{P} \rightarrow \mathbb{G}$ is injective and satisfies, for $a \in Act$: $G(p) \xrightarrow{a} G(q) \Leftrightarrow p \xrightarrow{a} q$. Moreover, $G(p) \xrightarrow{a} h$ only if h has the form $G(q)$ for some $q \in \mathbb{P}$ (with $p \xrightarrow{a} q$).
Proof: Trivial. □

Proposition 1.1 says that G is an *embedding* of $\mathbb{P}$ in $\mathbb{G}$. It implies that any labelled transition system over Act can be represented as a subclass $G(\mathbb{P}) = \{G(p) \in \mathbb{G} \mid p \in \mathbb{P}\}$ of $\mathbb{G}$.

Since $\mathbb{G}$ is also a labelled transition system, G can be applied to $\mathbb{G}$ itself. The following proposition says that the function $G : \mathbb{G} \rightarrow \mathbb{G}$ leaves its arguments intact up to graph isomorphism.

Proposition 1.2 For $g \in \mathbb{G}$, $G(g) \cong g$.
Proof: Remark that $\text{NODES}(G(g)) = \{g_s \mid s \in \text{NODES}(g)\}$.
Now the function $f : \text{NODES}(G(g)) \rightarrow \text{NODES}(g)$ defined by $f(g_s) = s$ is a graph isomorphism. □

1.4 Equivalences relations and preorders on labelled transition systems

This paper studies semantics on labelled transition systems. Each of the semantics examined here (except for tree semantics) is defined or characterized in terms of a function $\mathcal{O}$ that associates with every process $p \in \mathbb{P}$ a set $\mathcal{O}(p)$. In most cases the elements of $\mathcal{O}(p)$ can be regarded as the possible observations one could make while interacting with the process p in the context of a particular testing scenario. The set $\mathcal{O}(p)$ then constitutes the observable behaviour of p. For every such $\mathcal{O}$, the equivalence relation $=_{\mathcal{O}} \in \mathbb{P} \times \mathbb{P}$ is given by $p =_{\mathcal{O}} q \Leftrightarrow \mathcal{O}(p) = \mathcal{O}(q)$, and the preorder $\sqsubseteq_{\mathcal{O}} \in \mathbb{P} \times \mathbb{P}$ by $p \sqsubseteq_{\mathcal{O}} q \Leftrightarrow \mathcal{O}(p) \subseteq \mathcal{O}(q)$. Obviously $p =_{\mathcal{O}} q \Leftrightarrow p \sqsubseteq_{\mathcal{O}} q \wedge q \sqsubseteq_{\mathcal{O}} p$. The semantic equivalence $=_{\mathcal{O}}$ partitions $\mathbb{P}$ into equivalence classes of processes that are indistinguishable by observation (using observations of type $\mathcal{O}$). The preorder $\sqsubseteq_{\mathcal{O}}$ moreover provides a partial order between these equivalence classes; one that could be taken to constitute an "implementation" relation. The associated *semantics*, also called $\mathcal{O}$, is the criterion that identifies two processes whenever they are

$\mathcal{O}$-equivalent. Two semantics are considered the same if the associated equivalence relations are the same.

As the definitions of $\mathcal{O}$ are given entirely in terms of action relations, they apply to any labelled transition system $\mathbb{P}$. Moreover, the definitions of $\mathcal{O}(p)$ involve only action relations between processes reachable from p. Thus Proposition 1.1 implies that $\mathcal{O}(G(p)) = \mathcal{O}(p)$. This in turn yields

Corollary 1.1 $p \sqsubseteq_{\mathcal{O}} q$ iff $G(p) \sqsubseteq_{\mathcal{O}} G(q)$ and $p =_{\mathcal{O}} q$ iff $G(p) =_{\mathcal{O}} G(q)$. □

Write $\mathcal{O} \preceq_{\mathbb{P}} \mathcal{N}$ if semantics $\mathcal{O}$ makes at least as much identifications as semantics $\mathcal{N}$. This is the case if the equivalence corresponding with $\mathcal{O}$ is equal to or coarser than the one corresponding with $\mathcal{N}$, i.e. if $p =_{\mathcal{N}} q \Rightarrow p =_{\mathcal{O}} q$ for all $p, q \in \mathbb{P}$. Let $\preceq$ abbreviate $\preceq_{\mathbb{G}}$. The following is then immediate by Corollary 1.1.

Corollary 1.2 $\mathcal{O} \preceq \mathcal{N}$ iff $\mathcal{O} \preceq_{\mathbb{P}} \mathcal{N}$ for *each* labelled transition system $\mathbb{P}$.
On the other hand, $\mathcal{O} \not\preceq \mathcal{N}$ iff $\mathcal{O} \not\preceq_{\mathbb{P}} \mathcal{N}$ for *some* labelled transition system $\mathbb{P}$. □

Write $\mathcal{O} \preceq^*_{\mathbb{P}} \mathcal{N}$ if $p \sqsubseteq_{\mathcal{N}} q \Rightarrow p \sqsubseteq_{\mathcal{O}} q$ for all $p, q \in \mathbb{P}$, and let $\preceq^*$ abbreviate $\preceq^*_{\mathbb{G}}$. By definition $\mathcal{O} \preceq^* \mathcal{N} \Rightarrow \mathcal{O} \preceq \mathcal{N}$ for all semantics $\mathcal{O}$ and $\mathcal{N}$. The reverse does not hold by definition, but it will be shown to hold for all semantics discussed in this paper (cf. Section 15).

1.5 Initial nondeterminism

In a process graph it need not be determined in which state one ends after performing a nonempty sequence of actions. This phenomenon is called *nondeterminism*. However, process graphs as defined above are not capable of modelling *initial nondeterminism*, as there is only one initial state. This can be rectified by considering *process graphs with multiple roots*, in which $\text{ROOTS}(g)$ may be any nonempty subset of $\text{NODES}(g)$—let $\mathbb{G}^{mr}$ be the class of such connected process graphs. A process graph with multiple roots can also be regarded as a nonempty set of process graphs with single roots. More generally, initial nondeterminism can be modelled in any labelled transition system $\mathbb{P}$ by regarding the nonempty subsets of $\mathbb{P}$ (rather than merely its elements) to be processes. The elements of a process $P \subseteq \mathbb{P}$ then represent the possible initial states of P.

Now any notion of observability $\mathcal{O}$ on $\mathbb{P}$ extends to processes with initial nondeterminism by defining $\mathcal{O}(P) = \bigcup_{p \in P} \mathcal{O}(p)$ for $P \subseteq \mathbb{P}$. Thus also the equivalences $=_{\mathcal{O}}$ and preorders $\sqsubseteq_{\mathcal{O}}$ are defined on such processes. Write $\mathcal{O} \preceq'_{\mathbb{P}} \mathcal{N}$ if $P =_{\mathcal{N}} Q \Rightarrow P =_{\mathcal{O}} Q$ for all nonempty $P, Q \subseteq \mathbb{P}$, and let $\preceq'$ abbreviate $\preceq'_{\mathbb{G}}$. Clearly, one has $\mathcal{O} \preceq' \mathcal{N} \Rightarrow \mathcal{O} \preceq \mathcal{N}$ for all semantics $\mathcal{O}$ and $\mathcal{N}$.

Let g be a process graph over Act with multiple roots. Let i be an action (*initialize*) which is not in Act. Define $\rho(g)$ as the process graph over $Act \cup \{i\}$ obtained from g by adding a new state $*$, which will be the root of $\rho(g)$, and adding a transition $(*, i, r)$ for every $r \in \text{ROOTS}(g)$. Now for every semantics $\mathcal{O}$ to be discussed in this paper it will be the case that $g \sqsubseteq_{\mathcal{O}} h \Leftrightarrow \rho(g) \sqsubseteq_{\mathcal{O}} \rho(h)$, as the reader may easily verify for each such $\mathcal{O}$. From this it follows that we have in fact $\mathcal{O} \preceq' \mathcal{N} \Leftrightarrow \mathcal{O} \preceq \mathcal{N}$ for all semantics $\mathcal{O}$ and $\mathcal{N}$ treated in this paper. This justifies focusing henceforth on process graphs with single roots and processes as mere elements of labelled transition systems.

2 Trace semantics

Definition 2 $\sigma \in Act^*$ is a *trace* of a process p if there is a process q such that $p \xrightarrow{\sigma} q$. Let $T(p)$ denote the set of traces of p. Two processes p and q are *trace equivalent*, notation $p =_T q$, if $T(p) = T(q)$. In *trace semantics* (T) two processes are identified iff they are trace equivalent.

Testing scenario Trace semantics is based on the idea that two processes are to be identified if they allow the same set of observations, where an observation simply consists of a sequence of actions performed by the process in succession.

Modal characterization

Definition 2.1 The set $\mathcal{L}_T$ of *trace formulas* over *Act* is defined recursively by:

- $\top \in \mathcal{L}_T$.
- If $\varphi \in \mathcal{L}_T$ and $a \in Act$ then $a\varphi \in \mathcal{L}_T$.

The *satisfaction relation* $\models \subseteq \mathbb{P} \times \mathcal{L}_T$ is defined recursively by:

- $p \models \top$ for all $p \in \mathbb{P}$.
- $p \models a\varphi$ if for some $q \in \mathbb{P}$: $p \xrightarrow{a} q$ and $q \models \varphi$.

Note that a trace formula satisfied by a process p represents nothing more or less than a trace of p. Hence one has

Proposition 2.1 $p =_T q \;\Leftrightarrow\; \forall \varphi \in \mathcal{L}_T (p \models \varphi \Leftrightarrow q \models \varphi)$. □

Process graph characterization Let $g \in \mathbb{G}^{mr}$ and $\pi : \; s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s_n \in \text{PATHS}(g)$. Then $T(\pi) := a_1 a_2 \cdots a_n \in Act^*$ is the *trace* of π. As $\mathbb{G}$ is a labelled transition system, $T(g)$ is defined above. Alternatively, it could be defined as the set of traces of paths of g. It is easy to see that these definitions are equivalent:

Proposition 2.2 $T(g) = \{T(\pi) \mid \pi \in \text{PATHS}(g)\}$. □

Explicit model In trace semantics a process can be represented by a trace equivalence class of process graphs, or equivalently by the set of its traces. Such a *trace set* is always nonempty and prefix-closed. The next proposition shows that the domain $\mathbb{T}$ of trace sets is in bijective correspondence with the domain $\mathbb{G}/_{=_T}$ of process graphs modulo trace equivalence, as well as with the domain $\mathbb{G}^{mr}/_{=_T}$ of process graphs with multiple roots modulo trace equivalence. Models of concurrency like $\mathbb{T}$, in which a process is not represented as an equivalence class but rather as a mathematically coded set of its properties, are sometimes referred to as *explicit models*.

Definition 2.2 The *trace domain* $\mathbb{T}$ is the set of subsets T of Act^* satisfying

$$\begin{array}{ll} \mathrm{T1} & \varepsilon \in \mathrm{T}, \\ \mathrm{T2} & \sigma\rho \in \mathrm{T} \;\Rightarrow\; \sigma \in \mathrm{T}. \end{array}$$

Proposition 2.3 $\mathrm{T} \in \mathbb{T} \Leftrightarrow \exists g \in \mathbb{G} : T(g) = \mathrm{T} \Leftrightarrow \exists g \in \mathbb{G}^{mr} : T(g) = \mathrm{T}$.
Proof: Let $\mathrm{T} \in \mathbb{T}$. Define the *canonical graph* $G(\mathrm{T})$ of T by $\text{NODES}(G(\mathrm{T})) = \mathrm{T}$, $\text{ROOT}(G(\mathrm{T})) = \varepsilon$ and $(\sigma, a, \rho) \in \text{EDGES}(G(\mathrm{T}))$ iff $\rho = \sigma a$. As T satisfies T2, $G(\mathrm{T})$ is connected, i.e. $G(\mathrm{T}) \in \mathbb{G}$. In fact, $G(\mathrm{T})$ is a tree. Moreover, for every path $\pi \in \text{PATHS}(G(\mathrm{T}))$ one has $T(\pi) = end(\pi)$. Hence, using Proposition 2.2, $T(G(\mathrm{T})) = \mathrm{T}$.

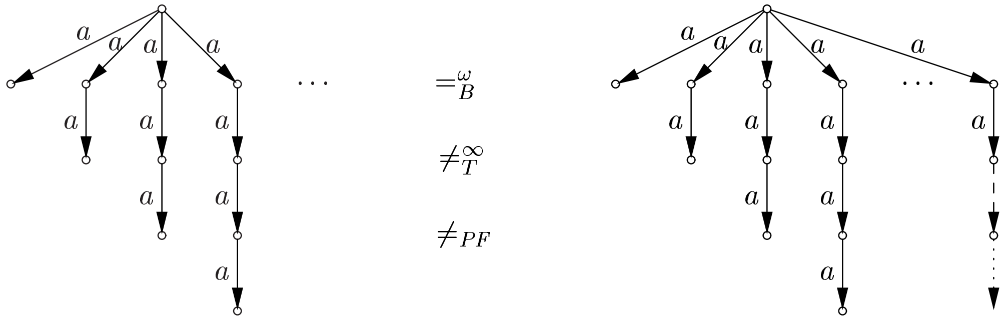
For the remaining two implication, note that $\mathbb{G} \subseteq \mathbb{G}^{mr}$, and the trace set $T(g)$ of any graph $g \in \mathbb{G}^{mr}$ satisfies T1 and T2. □

$\mathbb{T}$ was used as a model of concurrency in HOARE [30].

Infinite processes For infinite processes one distinguishes two variants of trace semantics: *(finitary) trace semantics* as defined above, and *infinitary trace semantics* (T^∞), obtained by taking infinite runs into account.

Definition 2.3 $a_1a_2\cdots \in Act^\infty$ is an *infinite trace* of a process $p \in \mathbb{P}$ if there are processes $p_1, p_2, \ldots$ such that $p \xrightarrow{a_1} p_1 \xrightarrow{a_2} \cdots$. Let $T^\infty(p)$ denote the set of infinite traces of p. Two processes p and q are *infinitary trace equivalent*, notation $p =_T^\infty q$, if $T(p) = T(q)$ and $T^\infty(p) = T^\infty(q)$.

Clearly $p =_T^\infty q \Rightarrow p =_T q$. That on $\mathbb{G}$ the reverse does not hold follows from Counterexample 1:



Counterexample 1: Finitary equivalent but not infinitary equivalent

one has $T(\mathit{left}) = T(\mathit{right}) = \{a^n \mid n \in \mathbb{N}\}$, but $T^\infty(\mathit{left}) \neq T^\infty(\mathit{right})$, as only the graph at the right has an infinite trace.

However, with König's lemma one easily proves that for image finite processes finitary and infinitary trace equivalence coincide:

Proposition 2.4 Let p and q be image finite processes with $p =_T q$. Then $p =_T^\infty q$.
Proof: It is sufficient to show that $T^\infty(p)$ can be expressed in terms of $T(p)$ for any image finite process p. In fact, $T^\infty(p)$ consists of all those infinite traces for which all finite prefixes are in $T(p)$. One direction of this statement is trivial: if $\sigma \in T^\infty(p)$, all finite prefixes of σ must be in $T(p)$. For the other direction suppose that, for $i \in \mathbb{N}$, $a_i \in Act$ and $a_1a_2\cdots a_i \in T(p)$. With induction on $i \in \mathbb{N}$ one can show that there exists processes p_i such that $i = 0$ and $p_0 = p$, or $p_{i-1} \xrightarrow{a_i} p_i$, and for every $j \geq i$ one has $a_{i+1}a_{i+2}\cdots a_j \in T(p_i)$. The existence of these p_i's immediately entails that $a_1a_2a_3\cdots \in T^\infty(p)$. The base case ($i = 0$) is trivial. Suppose the claim holds for certain i. For every $j \geq i+1$ there must be a process q with $p_i \xrightarrow{a_{i+1}} q$ and $a_{i+2}a_{i+3}\cdots a_j \in T(q)$. As there are only finitely many processes q with $p_i \xrightarrow{a_{i+1}} q$, there must be one choice of q for which $a_{i+2}a_{i+3}\cdots a_j \in T(q)$ for infinitely many values of j. Take this q to be p_{i+1}. As $T(p_{i+1})$ is prefix-closed, one has $a_{i+2}a_{i+3}\cdots a_j \in T(p_{i+1})$ for *all* $j \geq i+1$. $\square$

An explicit representation of infinitary trace semantics is obtained by taking the subsets T of Act^ω satisfying T1 and T2.

3 Completed trace semantics

Definition 3 $\sigma \in Act^*$ is a *complete trace* of a process p, if there is a process q such that $p \xrightarrow{\sigma} q$ and $I(q) = \emptyset$. Let $CT(p)$ denote the set of complete traces of p. Two processes p and q are *completed trace equivalent*, notation $p =_{CT} q$, if $T(p) = T(q)$ and $CT(p) = CT(q)$. In *completed trace semantics* (CT) two processes are identified iff they are completed trace equivalent.

Testing scenario Completed trace semantics can be explained with the following (rather trivial) *completed trace machine*. The process is modelled as a black box that contains as its interface to

Figure 2: The completed trace machine

the outside world a display on which the name of the action is shown that is currently carried out by the process. The process autonomously chooses an execution path that is consistent with its position in the labelled transition system $(\mathbb{P}, \rightarrow)$. During this execution always an action name is visible on the display. As soon as no further action can be carried out, the process reaches a state of deadlock and the display becomes empty. Now the existence of an observer is assumed that watches the display and records the sequence of actions displayed during a run of the process, possibly followed by deadlock. It is assumed that an observation takes only a finite amount of time and may be terminated before the process stagnates. Hence the observer records either a sequence of actions performed in succession—a trace of the process—or such a sequence followed by deadlock—a completed trace. Two processes are identified if they allow the same set of observations in this sense.

The *trace machine* can be regarded as a simpler version of the completed trace machine, were the last action name remains visible in the display if deadlock occurs (unless deadlock occurs in the beginning already). On this machine traces can be recorded, but stagnation can not be detected, since in case of deadlock the observer may think that the last action is still continuing.

Modal characterization

Definition 3.1 The set $\mathcal{L}_{CT}$ of *completed trace formulas* over *Act* is defined recursively by:

- $\top \in \mathcal{L}_{CT}$.
- $0 \in \mathcal{L}_{CT}$.
- If $\varphi \in \mathcal{L}_{CT}$ and $a \in Act$ then $a\varphi \in \mathcal{L}_{CT}$.

The *satisfaction relation* $\models \subseteq \mathbb{P} \times \mathcal{L}_{CT}$ is defined recursively by:

- $p \models \top$ for all $p \in \mathbb{P}$.
- $p \models 0$ if $I(p) = \emptyset$.
- $p \models a\varphi$ if for some $q \in \mathbb{P}$: $p \xrightarrow{a} q$ and $q \models \varphi$.

Note that a completed trace formula satisfied by a process p represents either a trace (if it has the form $a_1 a_2 \cdots a_n \top$) or a completed trace (if it has the form $a_1 a_2 \cdots a_n 0$). Hence one has

Proposition 3.1 $p =_{CT} q \;\Leftrightarrow\; \forall \varphi \in \mathcal{L}_{CT}(p \models \varphi \Leftrightarrow q \models \varphi)$. □

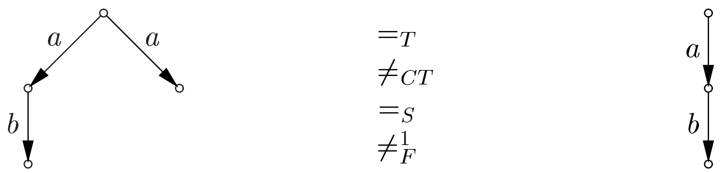
Also note the close link between the constructors of the modal formulas (corresponding to the three clauses in Definition 3.1) and the types of observations according to the testing scenario: $\top$

represents the act of the observer of terminating the observation, regardless of whether the observed process has terminated, 0 represents the observation of deadlock (the display becomes empty), and $a\varphi$ represents the observation of a being displayed, followed by the observation φ.

Process graph characterization Let $g \in \mathbb{G}^{mr}$ and $s \in \textsc{nodes}(g)$. Then $I(s) := \{a \in Act \mid \exists t : (s,a,t) \in \textsc{edges}(g)\}$ is the *menu* of s. $CT(g)$ can now be characterized as follows.

Proposition 3.2 $CT(g) = \{T(\pi) \mid \pi \in \textsc{paths}(g) \wedge I(end(\pi)) = \emptyset\}$. □

Classification Trivially $T \preceq CT$ (as in Figure 1). Counterexample 2 shows that the reverse



$ab + a$ ab

Counterexample 2: Trace and simulation equivalent, but not completed trace equivalent

does not hold: one has $T(\mathit{left}) = T(\mathit{right}) = \{\varepsilon,\ a,\ ab\}$, whereas $CT(\mathit{left}) \neq CT(\mathit{right})$ (since $a \in CT(\mathit{left}) - CT(\mathit{right})$). Hence the two process graphs are identified in trace semantics but distinguished in completed trace semantics. Thus $T \prec CT$: on $\mathbb{G}$ completed trace semantics makes strictly less identifications than trace semantics.

Explicit model In completed trace semantics a process can be represented by a completed trace equivalence class of process graphs, or equivalently by the pair $(\mathrm{T}, \mathrm{CT})$ of its sets of traces and complete traces. The next proposition gives an explicit characterization of the domain $\mathbb{CT}$ of pairs of sets of traces and complete traces of process graphs with multiple roots.

Definition 3.2 The *completed trace domain* $\mathbb{CT}$ is the set of pairs $(\mathrm{T}, \mathrm{CT}) \in Act^* \times Act^*$ satisfying

$$\mathrm{T} \in \mathbb{T} \text{ and } \mathrm{CT} \subseteq \mathrm{T},$$
$$\sigma \in \mathrm{T} - \mathrm{CT} \ \Rightarrow\ \exists a \in Act : \sigma a \in \mathrm{T}.$$

Proposition 3.3 $(\mathrm{T}, \mathrm{CT}) \in \mathbb{CT} \Leftrightarrow \exists g \in \mathbb{G}^{mr} : T(g) = \mathrm{T} \wedge CT(g) = \mathrm{T}$.
Proof: Let $(\mathrm{T}, \mathrm{CT}) \in \mathbb{CT}$. Define the *canonical graph* $G(\mathrm{T}, \mathrm{CT})$ of $(\mathrm{T}, \mathrm{CT})$ by

- $\textsc{nodes}(G(\mathrm{T}, \mathrm{CT})) = \mathrm{T} \cup \{\sigma\delta \mid \sigma \in \mathrm{CT}\}$,
- $\textsc{roots}(G(\mathrm{T}, \mathrm{CT})) = \{\varepsilon\} \cup \{\delta \mid \varepsilon \in \mathrm{CT}\}$ and
- $(\sigma, a, \rho) \in \textsc{edges}(G(\mathrm{T}))$ iff $\rho = \sigma a \vee \rho = \sigma a \delta$.

As T satisfies T2, $G(\mathrm{T}, \mathrm{CT})$ is connected, i.e. $G(\mathrm{T}, \mathrm{CT}) \in \mathbb{G}^{mr}$. In fact, $G(\mathrm{T}, \mathrm{CT})$ is a tree, except that it may have two roots. Using Propositions 2.2 and 3.2 it is easy to see that $T(G(\mathrm{T}, \mathrm{CT})) = \mathrm{T}$ and $CT(G(\mathrm{T}, \mathrm{CT})) = \mathrm{CT}$. □

The pairs obtained from process graphs with single roots are the ones moreover satisfying

$$\varepsilon \in CT \Leftrightarrow T = \{\varepsilon\}.$$

Infinite processes Also for completed trace semantics one can distinguish a finitary and an infinitary variant. In terms of the testing scenario, the latter (CT^∞) postulates that observations may take an infinite amount of time.

Definition 3.3 Two processes p and q are *infinitary completed trace equivalent*, notation $p =_{CT}^{\infty} q$, if $CT(p) = CT(q)$ and $T^\infty(p) = T^\infty(q)$. Note that in this case also $T(p) = T(q)$.

Proposition 2.4 implies that for image finite processes CT and CT^∞ coincide, whereas Counterexample 1 shows that in general the two are different. In fact, $T \prec T^\infty \prec CT^\infty$ and $T \prec CT \prec CT^\infty$, and the two preceding counterexamples show that there are no further inclusions.

4 Failures semantics

Testing scenario The *failures machine* contains as its interface to the outside world not only the display of the completed trace machine, but also a switch for each action $a \in Act$ (as in Figure 3). By means of these switches the observer may determine which actions are *free* and which are

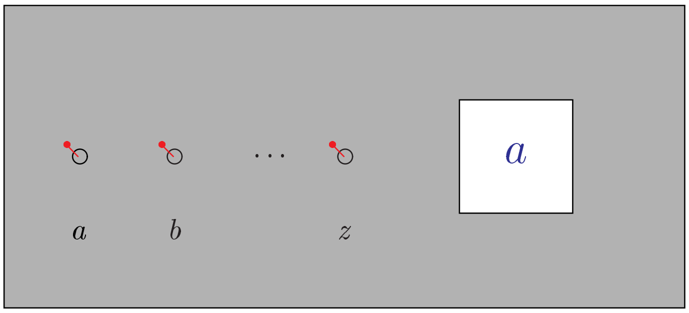


Figure 3: The failure trace machine

blocked. This situation may be changed any time during a run of the process. As before, the process autonomously chooses an execution path that fits with its position in $(\mathbb{P}, \rightarrow)$, but this time the process may only start the execution of free actions. If the process reaches a state where all initial actions of its remaining behaviour are blocked, it can not proceed and the machine stagnates, which can be recognized from the empty display. In this case the observer may record that after a certain sequence of actions σ, the set X of free actions is refused by the process. X is therefore called a *refusal set* and $\langle \sigma, X \rangle$ a *failure pair*. The set of all failure pairs of a process is called its *failure set*, and constitutes its observable behaviour.

Definition 4 $\langle \sigma, X \rangle \in Act^* \times \mathcal{P}(Act)$ is a *failure pair* of a process p if there is a process q such that $p \xrightarrow{\sigma} q$ and $I(q) \cap X = \emptyset$. Let $F(p)$ denote the set of failure pairs of p. Two processes p and q are *failures equivalent*, notation $p =_F q$, if $F(p) = F(q)$. In *failures semantics* (F) two processes are identified iff they are failures equivalent.

Note that $T(p)$ can be expressed in terms of $F(p)$: $T(p) = \{\sigma \in Act^* \mid \langle \sigma, \emptyset \rangle \in F(p)\}$; hence $p =_F q$ implies $T(p) = T(q)$.

Definition 4.1 For $p \in \mathbb{P}$ and $\sigma \in T(p)$, let $Cont_p(\sigma) = \{a \in Act \mid \sigma a \in T(p)\}$, the set of possible *continuations* of σ.

The following proposition says that the failure set $F(p)$ of a process p is completely determined by the set of failure pairs $\langle \sigma, X\rangle$ with $X \subseteq Cont_p(\sigma)$.

Proposition 4.1 Let $p\in\mathbb{P}$, $\sigma\in T(p)$ and $X\subseteq Act$. Then $\langle\sigma, X\rangle\in F(p) \Leftrightarrow \langle\sigma, X \cap Cont_p(\sigma)\rangle\in F(p)$.
Proof: If $p \xrightarrow{\sigma} q$ then $I(q) \subseteq Cont_p(\sigma)$. □

Modal characterization

Definition 4.2 The set $\mathcal{L}_F$ of *failure formulas* over Act is defined recursively by:

- $\top \in \mathcal{L}_F$.
- $\widetilde{X} \in \mathcal{L}_F$ for $X \subseteq Act$.
- If $\varphi \in \mathcal{L}_F$ and $a \in Act$ then $a\varphi \in \mathcal{L}_F$.

The *satisfaction relation* $\models \subseteq \mathbb{P} \times \mathcal{L}_F$ is defined recursively by:

- $p \models \top$ for all $p \in \mathbb{P}$.
- $p \models \widetilde{X}$ if $I(p) \cap X = \emptyset$.
- $p \models a\varphi$ if for some $q \in \mathbb{P}$: $p \xrightarrow{a} q$ and $q \models \varphi$.

$\widetilde{X}$ represents the observation that the process refuses the set of actions X, i.e. that stagnation occurs in a situation where X is the set of actions allowed by the environment. Note that a failure formula satisfied by a process p represents either a trace (if it has the form $a_1a_2\cdots a_n\top$) or a failure pair (if it has the form $a_1a_2\cdots a_n\widetilde{X}$). Hence one has

Proposition 4.2 $p =_F q \;\Leftrightarrow\; \forall\varphi \in \mathcal{L}_F(p \models \varphi \Leftrightarrow q \models \varphi)$. □

Process graph characterization Let $g \in \mathbb{G}^{mr}$ and $\pi \in \textsc{paths}(g)$. Then

$$F(\pi) := \{\langle T(\pi), X\rangle \mid I(end(\pi)) \cap X = \emptyset\}$$

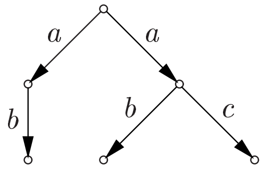
is the *failure set* of π. $F(g)$ can now be characterized as follows.

Proposition 4.3 $F(g) = \bigcup_{\pi\in\textsc{paths}(g)} F(\pi)$. □

Classification $CT \prec F$.
Proof: For "$CT \preceq F$" it suffices to show that also $CT(p)$ can be expressed in terms of $F(p)$:

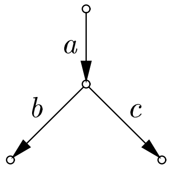
$$CT(p) = \{\sigma \in Act^* \mid \langle\sigma, Act\rangle \in F(p)\}.$$

It also suffices to show that the modal language $\mathcal{L}_{CT}$ is a sublanguage of $\mathcal{L}_F$: $p \models 0 \Leftrightarrow p \models \widetilde{Act}$.
"$CT \not\succeq F$" follows from Counterexample 3: one has $CT(left) = CT(right) = \{ab,\ ac\}$, whereas $F(left) \neq F(right)$ (since $\langle a, \{c\}\rangle \in F(left) - F(right)$). □



$ab + a(b + c)$

$=_{CT}$
$\neq_F$
$=_{CS}$
$\neq_F^1$



$a(b + c)$

Counterexample 3: Completed trace and completed simulation equivalent, but not failures equivalent or even singleton-failures equivalent

Explicit model In failures semantics a process can be represented by a failures equivalence class of process graphs, or equivalently by its failure set. The next proposition gives an explicit characterization of the domain $\mathbb{F}$ of failure sets of process graphs with multiple roots.

Definition 4.3 The *failures domain* $\mathbb{F}$ is the set of subsets F of $Act^* \times \mathcal{P}(Act)$ satisfying

F1 $\langle\varepsilon, \emptyset\rangle \in \mathrm{F}$,
F2 $\langle\sigma\rho, \emptyset\rangle \in \mathrm{F} \;\Rightarrow\; \langle\sigma, \emptyset\rangle \in \mathrm{F}$,
F3 $\langle\sigma, Y\rangle \in \mathrm{F} \wedge X \subseteq Y \;\Rightarrow\; \langle\sigma, X\rangle \in \mathrm{F}$,
F4 $\langle\sigma, X\rangle \in \mathrm{F} \wedge \forall a \in Y(\langle\sigma a, \emptyset\rangle \notin \mathrm{F}) \;\Rightarrow\; \langle\sigma, X \cup Y\rangle \in \mathrm{F}$.

Proposition 4.4 $\mathrm{F} \in \mathbb{F} \Leftrightarrow \exists g \in \mathbb{G}^{mr} : F(g) = \mathrm{F}$.
Proof: "$\Leftarrow$": F1 and F2 follow from T1 and T2 in Section 2, as one has $\langle\sigma, \emptyset\rangle \in F(g) \Leftrightarrow \sigma \in T(g)$.
F3 follows immediately from the definitions, as $I(q) \cap Y = \emptyset \wedge X \subseteq Y \Rightarrow I(q) \cap X = \emptyset$.
F4 follows immediately from Proposition 4.1, as $\forall a \in Y(\langle\sigma a, \emptyset\rangle \notin F(g))$ iff $Y \cap Cont_g(\sigma) = \emptyset$.

For "$\Rightarrow$" let $\mathrm{F} \in \mathbb{F}$. For $\sigma \in Act^*$ write $Cont_{\mathrm{F}}(\sigma)$ for $\{a \in Act \mid \langle\sigma a, \emptyset\rangle \in \mathrm{F}\}$.
Define the *canonical graph* $G(\mathrm{F})$ of F by

- $\textsc{nodes}(G(\mathrm{F})) = \{\langle\sigma, X\rangle \in \mathrm{F} \mid X \subseteq Cont_{\mathrm{F}}(\sigma)\}$,
- $\textsc{roots}(G(\mathrm{F})) = \{\langle\varepsilon, X\rangle \mid \langle\varepsilon, X\rangle \in \mathrm{F}\}$,
- $\textsc{edges}(G(\mathrm{F})) = \{(\langle\sigma, X\rangle, a, \langle\sigma a, Y\rangle) \mid \langle\sigma, X\rangle, \langle\sigma a, Y\rangle \in \textsc{nodes}(G(\mathrm{F})) \wedge a \notin X\}$.

By F1, $\textsc{roots}(G(\mathrm{F})) \neq \emptyset$. Using F3 and F2, any node $s = \langle a_1 \cdots a_n, X\rangle$ of $G(\mathrm{F})$ is reachable from a root by the path $\pi_s : \langle\varepsilon, \emptyset\rangle \xrightarrow{a_1} \langle a_1, \emptyset\rangle \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} \langle a_1 \cdots a_{n-1}, \emptyset\rangle \xrightarrow{a_n} \langle a_1 \cdots a_n, X\rangle$; hence $G(F)$ is connected. So $G(\mathrm{F}) \in \mathbb{G}^{mr}$. I have to show that $F(G(\mathrm{F})) = \mathrm{F}$.

"$\supseteq$": Suppose $\langle\sigma, X\rangle \in \mathrm{F}$. Then, by F3, $s := \langle\sigma, X \cap Cont_{\mathrm{F}}(\sigma)\rangle \in \textsc{nodes}(G(\mathrm{F}))$. By construction one has $T(\pi_s) = \sigma$ and $I(s) \cap X = \emptyset$. Hence $\langle\sigma, X\rangle \in F(\pi_s) \subseteq F(G(\mathrm{F}))$.

"$\subseteq$": With induction on the length of paths, it follows immediately from the definition of $G(\mathrm{F})$ that for $\pi \in \textsc{paths}(G(\mathrm{F}))$, if $end(\pi) = \langle\rho, Y\rangle$ then $\rho = T(\pi)$ and $I(end(\pi)) = Cont_{\mathrm{F}}(\rho) - Y$. (*)
Suppose $\langle\sigma, X\rangle \in F(G(\mathrm{F}))$. Then, by Proposition 4.3, there must be a path $\pi \in \textsc{paths}(G(\mathrm{F}))$ with $\langle\sigma, X\rangle \in F(\pi)$. So $T(\pi) = \sigma$ and $I(end(\pi)) \cap X = \emptyset$. Let $end(\pi) := \langle\rho, Y\rangle \in \mathrm{F}$. By (*), $\rho = \sigma$ and $X \cap Cont_{\mathrm{F}}(\sigma) \subseteq Y$. By F3 it follows that $\langle\sigma, X \cap Cont_{\mathrm{F}}(\sigma)\rangle \in \mathrm{F}$, and F4 yields $\langle\sigma, X\rangle \in \mathrm{F}$. □

A variant of $\mathbb{F}$ was used as a model of concurrency in Hoare [31].[7]

[7] There a process is given as a triple (A, F, D) with $A \subseteq Act$ a set of actions that may occur in the process, $F \in \mathbb{F}$ and D a set of so-called *divergencies*, traces that can lead along a state where an infinite sequence of internal actions is possible. As this paper considers only concrete, and hence divergence-free, processes, D is always empty here.

If $\text{ROOTS}(g)$ would be allowed to be empty, a characterization is obtained by dropping requirement F1. A characterization of the domain of failure sets of process graphs with single roots is given by adding to F1–4 the requirement

$$\text{F5} \quad \langle \varepsilon, X\rangle \in \text{F} \;\Rightarrow\; \forall a \in X : \langle a, \emptyset\rangle \notin \text{F}.$$

That F5 holds follows from the observation that $I(\text{ROOT}(g)) = \{a \in \mathit{Act} \mid \langle a, \emptyset\rangle \in F(g)\}$ for $g \in \mathbb{G}$.

Alternative characterizations In De Nicola [16] several equivalences, that were proposed in Kennaway [34], Darondeau [15] and De Nicola & Hennessy [17], are shown to coincide with failures semantics on the domain of finitely branching transition systems without internal moves. For this purpose he uses the following alternative characterization of failures equivalence.

Definition 4.4 Write p *after* σ *MUST* X if for each $q \in \mathbb{P}$ with $p \xrightarrow{\sigma} q$ there is an $a \in I(q)$ with $a \in X$. Put $p \simeq q$ if for all $\sigma \in \mathit{Act}^*$ and $X \subseteq \mathit{Act}$: p *after* σ *MUST* $X \;\Leftrightarrow\; q$ after σ *MUST* X.

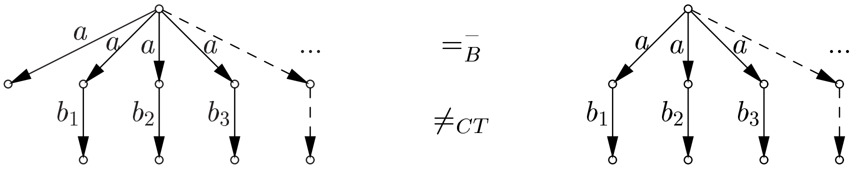
Proposition 4.5 Let $p, q \in \mathbb{P}$. Then $p \simeq q \;\Leftrightarrow\; p =_F q$.
Proof: p *after* σ *MUST* $X \;\Leftrightarrow\; \langle \sigma, X\rangle \notin F(p)$ [16]. □

Instead of the complement of the failure set of a process p, one can also take the complement $\mathit{Cont}_p(\sigma) - X$ of every refusal set X within a failure pair $\langle \sigma, X\rangle$ of p. In view of Proposition 4.1, the same information stored in $F(p)$ is given by the set of all pairs $\langle \sigma, X\rangle \in \mathit{Act}^* \times \mathcal{P}(\mathit{Act})$ for which there is a process q such that $p \xrightarrow{\sigma} q$ and $I(q) \subseteq X \subseteq \mathit{Cont}_p(\sigma)$. In Hennessy [26], a model for nondeterministic behaviours is proposed in which a process is represented as an *acceptance tree*. An acceptance tree of a finitely branching process without internal moves is essentially the set of pairs described above, conveniently represented as a finitely branching, deterministic process tree, of which the nodes are labelled by collections of sets of actions. Thus acceptance trees constitute an explicit model of failures semantics.

Infinite processes For infinite processes, three versions of failures semantics can be distinguished.

Definition 4.5 Two processes p and q are *(finitary) failures equivalent* if $F(p) = F(q)$. p and q are *infinitary failures equivalent*, notation $p =_F^\infty q$, if $F(p) = F(q)$ and $T^\infty(p) = T^\infty(q)$. They are *finite-failures equivalent*, notation $p =_F^- q$, if $F^-(p) = F^-(q)$, where $F^-(p)$ denotes the set of failure pairs $\langle \sigma, X\rangle$ of p with X finite.

The original failures semantics of Brookes, Hoare & Roscoe [13] is F^-, i.e. what I call *finite-failures semantics*. They “adopt this view of distinguishability because [they] consider a *realistic* environment to be one that is at any time capable of performing only a finite number of events.” In terms of the failures machine this means that at any time only finitely many switches can be set on free. Finitary failures semantics is the default version introduced at the beginning of this section. This can be regarded to be the semantics employed in Brookes & Roscoe [14] and Hoare [31]. Infinitary failures semantics was first discussed in Bergstra, Klop & Olderog [10]; it was proposed as a semantics for CSP in Roscoe [45]. The difference between the testing scenarios for F and F^∞ is that only the latter allows observations of infinite duration. Obviously, $F^- \preceq F \preceq F^\infty$. That the latter inclusion is strict follows from Counterexample 1; Counterexample 4 shows that also the former is strict: one has $F^-(\mathit{left}) = F^-(\mathit{right})$, whereas $F(\mathit{left}) \neq F(\mathit{right})$. In fact even



Counterexample 4: HML- and finite-failures equivalent, but not completed trace equivalent

$CT(\mathit{left}) \neq CT(\mathit{right})$, as $a \in CT(\mathit{left}) - CT(\mathit{right})$. Thus, although $T \prec F^-$, $CT \prec F$ and $CT^\infty \prec F^\infty$, CT and F^- are independent, as are CT^∞ and F.

In addition to the three variants of Definition 4.5 one could also define a version of failures semantics based on infinite traces and finite refusal sets. Such a semantics would distinguish the two graphs of Counterexample 1, but identify the ones of Counterexample 4. As this semantics does not occur in the literature, and has no clear advantages over the other variants, I will not further consider it here.

Proposition 4.6 Let p en q be image finite processes. Then $p =^-_F q \Leftrightarrow p =_F q \Leftrightarrow p =^\infty_F q$.
Proof: "$\Leftarrow$" has been established for all processes, and the second "$\Rightarrow$" follows immediately from Proposition 2.4 (as $p =_F q \Rightarrow p =_T q \Rightarrow p =^\infty_T q$). So it remains to show that $p \neq_F q \Rightarrow p \neq^-_F q$. Suppose $F(p) \neq F(q)$, say there is a failure pair $\langle \sigma, X \rangle \in F(p) - F(q)$. By the image finiteness of q there are only finitely many processes r_i with $q \xrightarrow{\sigma} r_i$, and for each of them there is an action $a_i \in I(r_i) \cap X$ (as otherwise $\langle \sigma, X \rangle$ would be a failure pair of q). Let Y be the set of all those a_i's. Then Y is a finite subset of X, so $\langle \sigma, Y \rangle \in F^-(p)$. On the other hand, $a_i \in I(r_i) \cap Y$ for all r_i, so $\langle \sigma, Y \rangle \notin F^-(q)$. □

It is not hard to change the leftmost process in Counterexample 4 to an image finite one with the same failure pairs. Thus, in the first statement of Proposition 4.6 it is necessary that both processes are image finite. For the subclass of finitely branching processes a stronger result can be obtained.

Proposition 4.7 Let $p, q \in \mathbb{P}$ and p is finitely branching. Then $p =^-_F q \Leftrightarrow p =_F q$.
Proof: Suppose $p =^-_F q$. As p is finitely branching, $\mathit{Cont}_p(\sigma)$ is finite for all $\sigma \in T(p)$. And as $T(q) = T(p)$, $\mathit{Cont}_q(\sigma) = \mathit{Cont}_p(\sigma)$, which is finite, for all $\sigma \in T(q)$. Now for processes p with this property, $F(p)$ is completely determined by $F^-(p)$, as follows from Proposition 4.1. □

The second statement of Proposition 4.6 does not allow such a strengthening, as will follow from Counterexample 12.

5 Failure trace semantics

Testing scenario The *failure trace machine* has the same layout as the failures machine, but is does not stagnate permanently if the process cannot proceed due to the circumstance that all actions it is prepared to continue with are blocked by the observer. Instead it idles—recognizable from the empty display—until the observer changes its mind and allows one of the actions the process is ready to perform. What can be observed are traces with idle periods in between, and for each such period the set of actions that are not blocked by the observer. Such observations can be coded as sequences of members and subsets of *Act*.

Example: The sequence $\{a,b\}cdb\{b,c\}\{b,c,d\}a(Act)$ is the account of the following observation: At the beginning of the execution of the process p, only the actions a and b were allowed by the observer. Apparently, these actions were not on the menu of p, for p started with an idle period. Suddenly the observer canceled its veto on c, and this resulted in the execution of c, followed by d and b. Then again an idle period occurred, this time when b and c were the actions not being blocked by the observer. After a while the observer decided to allow d as well, but the process ignored this gesture and remained idle. Only when the observer gave the green light for the action a, it happened immediately. Finally, the process became idle once more, but this time not even one action was blocked. This made the observer realize that a state of eternal stagnation had been reached, and disappointed he terminated the observation.

A set $X \subseteq Act$, occurring in such a sequence, can be regarded as an offer from the environment, that is refused by the process. Therefore such a set is called a *refusal set*. The occurrence of a refusal set may be interpreted as a 'failure' of the environment to create a situation in which the process can proceed without being disturbed. Hence a sequence over $Act \cup \mathcal{P}(Act)$, resulting from an observation of a process p may be called a *failure trace* of p. The observable behaviour of a process, according to this testing scenario, is given by the set of its failure traces, its *failure trace set*. The semantics in which processes are identified iff their failure trace sets coincide, is called *failure trace semantics* (FT).

For image finite processes failure trace semantics is exactly the equivalence that originates from Phillips notion of *refusal testing* [42]. (Image infinite processes are not considered in [42].) There it is called *refusal equivalence*.

Definition 5

- The *refusal relations* $\xrightarrow{X}$ for $X \subseteq Act$ are defined by: $p \xrightarrow{X} q$ iff $p = q$ and $I(p) \cap X = \emptyset$. $p \xrightarrow{X} q$ means that p can evolve into q, while being idle during a period in which X is the set of actions allowed by the environment.
- The *failure trace relations* $\xrightarrow{\sigma}$ for $\sigma \in (Act \cup \mathcal{P}(Act))^*$ are defined as the reflexive and transitive closure of both the action and the refusal relations. Again the overloading of notation is harmless.
- $\sigma \in (Act \cup \mathcal{P}(Act))^*$ is a *failure trace* of a process p if there is a process q such that $p \xrightarrow{\sigma} q$. Let $FT(p)$ denote the set of failure traces of p. Two processes p and q are *failure trace equivalent*, notation $p =_{FT} q$, if $FT(p) = FT(q)$.

Modal characterization

Definition 5.1 The set $\mathcal{L}_{FT}$ of *failure trace formulas* over Act is defined recursively by:

- $\top \in \mathcal{L}_{FT}$.
- If $\varphi \in \mathcal{L}_{FT}$ and $X \subseteq Act$ then $\widetilde{X}\varphi \in \mathcal{L}_{FT}$.
- If $\varphi \in \mathcal{L}_{FT}$ and $a \in Act$ then $a\varphi \in \mathcal{L}_{FT}$.

The *satisfaction relation* $\models \subseteq \mathbb{P} \times \mathcal{L}_{FT}$ is defined recursively by:

- $p \models \top$ for all $p \in \mathbb{P}$.
- $p \models \widetilde{X}\varphi$ if $I(p) \cap X = \emptyset$ and $p \models \varphi$.
- $p \models a\varphi$ if for some $q \in \mathbb{P}$: $p \xrightarrow{a} q$ and $q \models \varphi$.

$\widetilde{X}\varphi$ represents the observation that the process refuses the set of actions X, followed by the observation φ. A modal failure trace formula satisfied by a process p represents exactly a failure trace as defined above. Hence one has

Proposition 5.1 $p =_{FT} q \;\Leftrightarrow\; \forall\varphi \in \mathcal{L}_{FT}(p \models \varphi \Leftrightarrow q \models \varphi)$. □

Process graph characterization Let $g \in \mathbb{G}^{mr}$ and $\pi : s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s_n \in \text{PATHS}(g)$. Then the *failure trace set* of π, $FT(\pi)$, is the smallest subset of $(Act \cup \mathcal{P}(Act))^*$ satisfying

- $(Act - I(s_0))a_1(Act - I(s_1))a_2 \cdots a_n(Act - I(s_n)) \in FT(\pi)$,
- $\sigma X \rho \in FT(\pi) \Rightarrow \sigma\rho \in FT(\pi)$,
- $\sigma X \rho \in FT(\pi) \Rightarrow \sigma X X \rho \in FT(\pi)$,
- $\sigma X \rho \in FT(\pi) \wedge Y \subset X \Rightarrow \sigma Y \rho \in FT(\pi)$.

$FT(g)$ can now be characterized as follows.

Proposition 5.2 $FT(g) = \bigcup_{\pi\in\text{PATHS}(g)} FT(\pi)$. □

Proposition 5.2 yields a technique for deciding that two process graphs are failure trace equivalent, without calculating their entire failure trace set.
Let $g, h \in \mathbb{G}^{mr}$, $\pi : s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s_n \in \text{PATHS}(g)$ and π': $t_0 \xrightarrow{b_1} t_1 \xrightarrow{b_2} \cdots \xrightarrow{b_m} t_m \in \text{PATHS}(h)$. Path π' is a *failure trace augmentation* of π, notation $\pi \leq_{FT} \pi'$, if $FT(\pi) \subseteq FT(\pi')$. This is the case exactly when $n = m$, $a_i = b_i$ and $I(t_i) \subseteq I(s_i)$ for $i = 1, ..., n$. From this the following can be concluded.

Corollary 5.1 Two process graphs $g, h \in \mathbb{G}^{mr}$ are failure trace equivalent iff

- for any path $\pi \in \text{PATHS}(g)$ in g there is a $\pi' \in \text{PATHS}(h)$ such that $\pi \leq_{FT} \pi'$
- and for any path $\pi \in \text{PATHS}(g)$ in h there is a $\pi' \in \text{PATHS}(g)$ such that $\pi \leq_{FT} \pi'$.

If g and h are moreover without infinite paths, then it suffices to check the requirements above for maximal paths. □

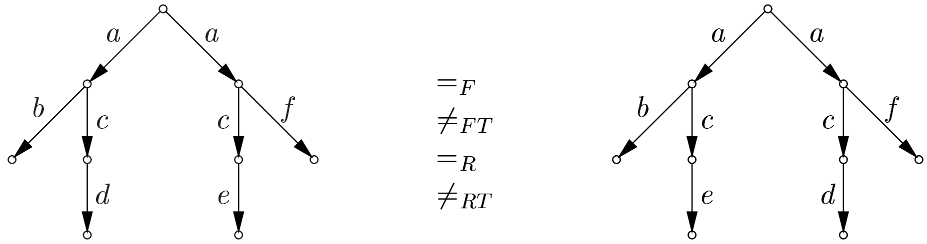
Classification $F \prec FT$.
Proof: For "$F \preceq FT$" it suffices to show that $F(p)$ can be expressed in terms of $FT(p)$:

$$\langle\sigma, X\rangle \in F(p) \quad\Leftrightarrow\quad \sigma X \in FT(p).$$

"$F \not\succeq FT$" follows from Counterexample 5; see Section 7 for details. □

Infinite processes As for failures semantics, three variants of failure trace semantics for infinite processes can be defined. Besides the default version (FT) there is an infinitary version (FT^∞), motivated by observations that may last forever, and a finite version (FT^-), motivated by an observer that may only set finitely many switches on free at any time.

Definition 5.2 $\sigma_1\sigma_2 \cdots \in (Act \cup \mathcal{P}(Act))^\infty$ is an *infinite failure trace* of a process $p \in \mathbb{P}$ if there are processes $p_1, p_2, ...$ such that $p \xrightarrow{\sigma_1} p_1 \xrightarrow{\sigma_2} \cdots$. Let $FT^\infty(p)$ denote the set of infinite failure traces of p. Two processes p and q are *infinitary failure trace equivalent*, notation $p =^\infty_{FT} q$, if



$a(b + cd) + a(f + ce)$ $\qquad$ $a(b + ce) + a(f + cd)$

Counterexample 5: Failures and ready equivalent, but not failure trace or ready trace equivalent

$FT^\infty(p) = FT^\infty(q)$ and $FT(p) = FT(q)$. They are *finite-failure trace equivalent*, notation $p =^-_{FT} q$, if $FT^-(p) = FT^-(q)$, where $FT^-(p)$ denotes the set of failure traces of p in which all refusal sets are finite.

Clearly, $FT^- \prec FT \prec FT^\infty$; Counterexamples 1 and 4 show that the inclusions and strict. One also has $F^- \prec FT^-$, $F \prec FT$ and $F^\infty \prec FT^\infty$; here strictness follows from Counterexample 5.

Proposition 5.3 Let p en q be image finite processes. Then $p =^-_{FT} q \Leftrightarrow p =_{FT} q \Leftrightarrow p =^\infty_{FT} q$.
Proof: "$p =^-_{FT} q \Leftarrow p =_{FT} q \Leftarrow p =^\infty_{FT} q$" holds for all processes.
Note that the definition of $FT(p)$ is exactly like the definition of $T(p)$, except that the failure trace relations are used instead of the generalized action relations; the same relation exists between $FT^\infty(p)$ and $T^\infty(p)$. Moreover, a process $p \in \mathbb{P}$ is image finite in terms of the failure trace relations on $\mathbb{P}$ iff it is image finite in terms of terms of the (generalized) action relations on $\mathbb{P}$, as defined in Definition 1.2. Hence "$p =_{FT} q \Rightarrow p =^\infty_{FT} q$" follows immediately from Proposition 2.4.
"$p =^-_{FT} q \Rightarrow p =_{FT} q$": Suppose $FT(p) \neq FT(q)$, say $FT(p) - FT(q) \neq \emptyset$. Let σ be a failure trace in $FT(p) - FT(q)$ with at least one infinite refusal set. I will show that there must be a failure trace in $FT(p) - FT(q)$ with strictly fewer infinite refusal sets than σ. By applying this result a finite number of times, a failure trace $\rho \in FT(p) - FT(q)$ is found without infinite refusal sets, showing that $FT^-(p) \neq FT^-(q)$.

So let $\sigma = \sigma_1 X \sigma_2 \in FT(p) - FT(q)$ with X an infinite refusal set. Clearly $\sigma_1\sigma_2 \in FT(p)$. By the image finiteness of q there are only finitely many pairs of processes r_i, s_i with $q \xrightarrow{\sigma_1} r_i \xrightarrow{\sigma_2} s_i$, and for each of them there is an action $a_i \in I(r_i) \cap X$ (as otherwise $\sigma_1 X \sigma_2$ would be a failure trace of q). Let Y be the set of all those a_i's. Then Y is finite. As Y is a subset of X, one has $\sigma_1 Y \sigma_2 \in FT(p)$. On the other hand, $a_i \in I(r_i) \cap Y$ for all r_i, so $\sigma_1 Y \sigma_2 \notin FT(q)$. $\square$

Unlike the situation for failures semantics, in the first statement of Proposition 5.3 it is not necessary that *both* processes are image finite.

Proposition 5.4 Let $p, q \in \mathbb{P}$ and p is image finite. Then $p =^-_{FT} q \Leftrightarrow p =_{FT} q$.
Proof: More difficult, and omitted here. $\square$

The second statement of Proposition 5.3 does not allow such a strengthening, as will follow from Counterexample 12.

6 Ready trace semantics

Testing scenario The *ready trace machine* is a variant of the failure trace machine that is equipped with a lamp for each action $a \in Act$. Each time the process idles, the lamps of all actions

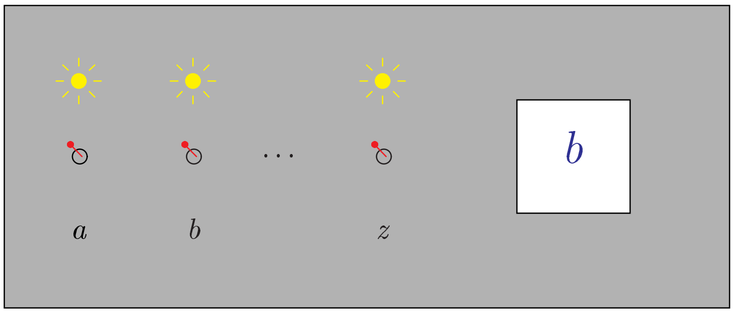


Figure 4: The ready trace machine

the process is ready to engage in are lit. Of course all these actions are blocked by the observer, otherwise the process wouldn't idle. Now the observer can see which actions could be released in order to let the process proceed. During the execution of an action no lamps are lit. An observation now consists of a sequence of members and subsets of Act, the actions representing information obtained from the display, and the sets of actions representing information obtained from the lights. Such a sequence is called a *ready trace* of the process, and the subsets occurring in a ready trace are referred to as *menus*. The information about the free and blocked actions is now redundant. The set of all ready traces of a process is called its *ready trace set*, and constitutes its observable behaviour.

Definition 6

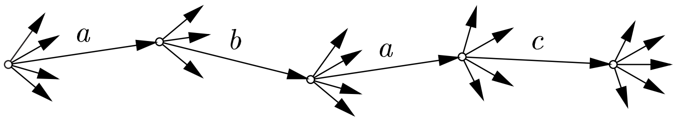
- The *ready trace relations* $\overset{\sigma}{\ast\!\!\longrightarrow}$ for $\sigma \in (Act \cup \mathcal{P}(Act))^*$ are defined recursively by:

  1. $p \overset{\varepsilon}{\ast\!\!\longrightarrow} p$, for any process p.
  2. $p \xrightarrow{a} q$ implies $p \overset{a}{\ast\!\!\longrightarrow} q$.
  3. $p \overset{X}{\ast\!\!\longrightarrow} q$ with $X \subseteq Act$ whenever $p = q$ and $I(p) = X$.
  4. $p \overset{\sigma}{\ast\!\!\longrightarrow} q \overset{\rho}{\ast\!\!\longrightarrow} r$ implies $p \overset{\sigma\rho}{\ast\!\!\longrightarrow} r$.

  The special arrow $\overset{\sigma}{\ast\!\!\longrightarrow}$ had to be used, since further overloading of $\xrightarrow{\sigma}$ would cause confusion with the failure trace relations.

- $\sigma \in (Act \cup \mathcal{P}(Act))^*$ is a *ready trace* of a process p if there is a process q such that $p \overset{\sigma}{\ast\!\!\longrightarrow} q$. Let $RT(p)$ denote the set of ready traces of p. Two processes p and q are *ready trace equivalent*, notation $p =_{RT} q$, if $RT(p) = RT(q)$. In *ready trace semantics* (RT) two processes are identified iff they are ready trace equivalent.

In BAETEN, BERGSTRA & KLOP [6], PNUELI [43] and POMELLO [44] ready trace semantics was defined slightly differently. By Proposition 6.1 below, their definition yields the same equivalence as mine.



Definition 6.1 $X_0 a_1 X_1 a_2 \cdots a_n X_n \in \mathcal{P}(Act) \times (Act \times \mathcal{P}(Act))^*$ is a *normal ready trace* of a process p if there are processes $p_1, ..., p_n$ such that $p \xrightarrow{a_1} p_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} p_n$ and $I(p_i) = X_i$ for $i = 1, ..., n$. Let $RT_N(p)$ denote the set of normal ready traces of p. Two processes p and q are ready trace equivalent in the sense of [6, 43, 44] if $RT_N(p) = RT_N(q)$.

Proposition 6.1 Let $p, q \in \mathbb{P}$. Then $RT_N(p) = RT_N(q) \;\Leftrightarrow\; RT(p) = RT(q)$.
Proof: The normal ready traces of a process are just the ready traces which are an alternating sequence of sets and actions, and vice versa the set of all ready traces can be constructed form the set of normal ready traces by means of doubling and leaving out menus. □

Modal characterization

Definition 6.2 The set $\mathcal{L}_{RT}$ of *ready trace formulas* over *Act* is defined recursively by:

- $\top \in \mathcal{L}_{RT}$.
- If $\varphi \in \mathcal{L}_{RT}$ and $X \subseteq Act$ then $X\varphi \in \mathcal{L}_{RT}$.
- If $\varphi \in \mathcal{L}_{RT}$ and $a \in Act$ then $a\varphi \in \mathcal{L}_{RT}$.

The *satisfaction relation* $\models \subseteq \mathbb{P} \times \mathcal{L}_{RT}$ is defined recursively by:

- $p \models \top$ for all $p \in \mathbb{P}$.
- $p \models X\varphi$ if $I(p) = X$ and $p \models \varphi$.
- $p \models a\varphi$ if for some $q \in \mathbb{P}$: $p \xrightarrow{a} q$ and $q \models \varphi$.

$X\varphi$ represents the observation of a menu, followed by the observation φ. A ready trace formula satisfied by a process p represents exactly a ready trace in Definition 6. Hence one has

Proposition 6.2 $p =_{RT} q \;\Leftrightarrow\; \forall \varphi \in \mathcal{L}_{RT}(p \models \varphi \Leftrightarrow q \models \varphi)$. □

Process graph characterization Let $g \in \mathbb{G}^{mr}$ and $\pi :\; s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s_n \in \text{PATHS}(g)$. The *ready trace* of π is given by $RT_N(\pi) := I(s_0) a_1 I(s_1) a_2 \cdots a_n I(s_n)$.
$RT_N(g)$ can now be characterized by:

Proposition 6.3 $RT_N(g) = \{RT_N(\pi) \mid \pi \in \text{PATHS}(g)\}$. □

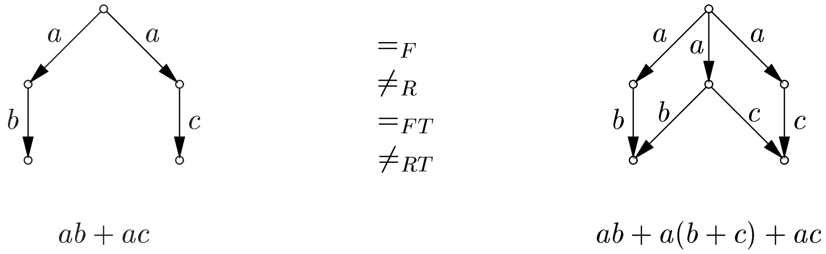
Moreover, $RT(g)$ is the smallest subset of $(Act \cup \mathcal{P}(Act))^*$ containing $RT_N(g)$ and satisfying

$$\sigma X \rho \in RT(g) \Rightarrow \sigma\rho \in RT(g) \wedge \sigma X X \rho \in RT(g).$$

Classification $FT \prec RT$.
Proof: For "$FT \preceq RT$" it suffices to show that $FT(p)$ can be expressed in terms of $RT(p)$:

> $\sigma = \sigma_1\sigma_2 \cdots \sigma_n \in FT(p)$ $(\sigma_i \in Act \cup \mathcal{P}(Act)) \;\Leftrightarrow$
> $\exists \rho = \rho_1\rho_2 \cdots \rho_n \in RT(p)$ $(\rho_i \in Act \cup \mathcal{P}(Act))$ such that for $i = 1, ..., n$ either
> $\sigma_i = \rho_i \in Act$ or $\sigma_i, \rho_i \subseteq Act$ and $\sigma_i \cap \rho_i = \emptyset$.

"$FT \not\succeq RT$" follows from Counterexample 6; see Section 7 for details. □



Counterexample 6: Failures and failure trace equivalent, but not ready or ready trace equivalent

Explicit model In ready trace semantics a process can be represented by a ready trace equivalence class of process graphs, or equivalently by its ready trace set, possibly in the normal form of Definition 6.1. The next proposition gives an explicit characterization of the domain $\mathbb{RT}$ of ready trace sets in this form of process graphs with multiple roots.

Definition 6.3 The *ready trace domain* $\mathbb{RT}$ is the set of subsets RT of $\mathcal{P}(Act) \times (Act \times \mathcal{P}(Act))^*$ satisfying

$$\begin{array}{ll} \text{RT1} & \exists X(X \in \text{RT}), \\ \text{RT2} & \sigma X \in \text{RT} \wedge a \in X \Leftrightarrow \exists Y(\sigma X a Y \in \text{RT}). \end{array}$$

Proposition 6.4 $\text{RT} \in \mathbb{RT} \Leftrightarrow \exists g \in \mathbb{G}^{mr} : RT_N(g) = \text{RT}$.
Proof: "$\Leftarrow$" is evident. For "$\Rightarrow$" let $\text{RT} \in \mathbb{RT}$. Define the *canonical graph* $G(\text{RT})$ of RT by

- $\textsc{nodes}(G(\text{RT})) = \text{RT}$,
- $\textsc{roots}(G(\text{RT})) = \{X \subseteq Act \mid X \in \text{RT}\}$,
- $\textsc{edges}(G(\text{RT})) = \{(\sigma, a, \sigma a Y) \mid \sigma, \sigma a Y \in \textsc{nodes}(G(\text{RT}))\}$.

By RT1, $\textsc{roots}(G(\text{RT})) \neq \emptyset$. Using R2, $G(\text{RT})$ is connected. So $G(\text{RT}) \in \mathbb{G}^{mr}$. Moreover, for every path $\pi \in \textsc{paths}(G(\text{RT}))$ one has $RT_N(\pi) = end(\pi)$. Hence $RT_N(G(\text{RT})) = \text{RT}$. □

If $\textsc{roots}(g)$ would be allowed to be empty, a characterization is obtained by dropping requirement RT1. A characterization of the domain of ready trace sets of process graphs with single roots is given by strengthening RT1 to $\exists! X(X \in \text{RT})$, where $\exists! X$ means "there is exactly one X such that".

Infinite processes An infinitary version of ready trace semantics (RT^∞) is defined analogously to infinitary failure trace semantics. A finite version is not so straightforward; a definition will be proposed in the next section.

Definition 6.4 $\sigma_1\sigma_2 \cdots \in (Act \cup \mathcal{P}(Act))^\infty$ is an *infinite ready trace* of a process $p \in \mathbb{P}$ if there are processes $p_1, p_2, \ldots$ such that $p \overset{\sigma_1}{\ast\!\!\longrightarrow} p_1 \overset{\sigma_2}{\ast\!\!\longrightarrow} \cdots$. Let $RT^\infty(p)$ denote the set of infinite ready traces of p. Two processes p and q are *infinitary ready trace equivalent*, notation $p =^\infty_{RT} q$, if $RT^\infty(p) = RT^\infty(q)$ and $RT(p) = RT(q)$.

Clearly, $RT \prec RT^\infty$; Counterexample 1 shows that the inclusion is strict. Moreover $FT^\infty \prec RT^\infty$.

Proposition 6.5 Let p en q be image finite processes. Then $p =_{RT} q \Leftrightarrow p =^\infty_{RT} q$.
Proof: Exactly as the corresponding part of Proposition 5.3. □

Counterexample 12 will show that in Proposition 6.5 *both* p and q need to be image finite.

7 Readiness semantics and possible-futures semantics

Testing scenario The *readiness machine* has the same layout as the ready trace machine, but, like the failures machine, can not recover from an idle period. By means of the lights the menu of initial actions of the remaining behaviour of an idle process can be recorded, but this happens at most once during an observation of a process, namely at the end. An observation either results in a trace of the process, or in a pair of a trace and a menu of actions by which the observation could have been extended if the observer wouldn't have blocked them. Such a pair is called a *ready pair* of the process, and the set of all ready pairs of a process is its *ready set*.

Definition 7 $\langle \sigma, X\rangle \in Act^* \times \mathcal{P}(Act)$ is a *ready pair* of a process p if there is a process q such that $p \xrightarrow{\sigma} q$ and $I(q) = X$. Let $R(p)$ denote the set of ready pairs of p. Two processes p and q are *ready equivalent*, notation $p =_R q$, if $R(p) = R(q)$. In *readiness semantics* (R) two processes are identified iff they are ready equivalent.

Modal characterization

Definition 7.1 The set $\mathcal{L}_R$ of *readiness formulas* over Act is defined recursively by:

- $\top \in \mathcal{L}_R$.
- $X \in \mathcal{L}_R$ for $X \subseteq Act$.
- If $\varphi \in \mathcal{L}_R$ and $a \in Act$ then $a\varphi \in \mathcal{L}_R$.

The *satisfaction relation* $\models \subseteq \mathbb{P} \times \mathcal{L}_R$ is defined recursively by:

- $p \models \top$ for all $p \in \mathbb{P}$.
- $p \models X$ if $I(p) = X$.
- $p \models a\varphi$ if for some $q \in \mathbb{P}$: $p \xrightarrow{a} q$ and $q \models \varphi$.

X represents the observation of a menu. A readiness formula satisfied by a process p represents either a trace (if it has the form $a_1a_2\cdots a_n\top$) or a ready pair (if it has the form $a_1a_2\cdots a_nX$). Hence one has

Proposition 7.1 $p =_R q \ \Leftrightarrow\ \forall \varphi \in \mathcal{L}_R(p \models \varphi \Leftrightarrow q \models \varphi)$. □

Process graph characterization Let $g \in \mathbb{G}^{mr}$ and $\pi \in \text{PATHS}(g)$. The *ready pair* of π is given by $R(\pi) := \langle T(\pi), I(end(\pi))\rangle$. $R(g)$ can now be characterized by:

Proposition 7.2 $R(g) = \{R(\pi) \mid \pi \in \text{PATHS}(g)\}$. □

Classification $F \prec R \prec RT$, but R and FT are independent.
Proof: For "$F \preceq R$" it suffices to show that $F(p)$ can be expressed in terms of $R(p)$:

$$\langle \sigma, X\rangle \in F(p) \quad\Leftrightarrow\quad \exists Y \subseteq Act : \langle \sigma, Y\rangle \in R(p) \ \wedge\ X \cap Y = \emptyset.$$

For "$R \preceq RT$" it suffices to show that $R(p)$ can be expressed in terms of $RT(p)$:

$$\langle \sigma, X\rangle \in R(p) \quad\Leftrightarrow\quad \sigma X \in RT(p).$$

"$R \not\succeq FT$" (and hence "$R \not\succeq RT$" and "$F \not\succeq FT$") follows from Counterexample 5, in which $R(\mathit{left}) = R(\mathit{right})$ but $FT(\mathit{left}) \neq FT(\mathit{right})$. The first statement follows with Proposition 7.2. Both graphs have 9 paths starting from the root, and hence 9 ready pairs. These are easily seen to be the same at both sides; in the second graph only 4 ready pairs swapped places. The second statement follows since $a\{b\}ce \in FT(\mathit{left}) - FT(\mathit{right})$.

"$R \not\preceq FT$" (and hence "$R \not\preceq F$" and "$RT \not\preceq FT$") follows from Counterexample 6, in which $FT(\mathit{left}) = FT(\mathit{right})$ but $R(\mathit{left}) \neq R(\mathit{right})$. The first statement follows from Corollary 5.1, since the new maximal paths at the right-hand side are both failure trace augmented by the two maximal paths both sides have in common. The second one follows since $\langle a, \{b, c\}\rangle \in R(\mathit{right}) - R(\mathit{left})$. □

Explicit model In readiness semantics a process can be represented by a ready equivalence class of process graphs, or equivalently by its ready set. The next proposition gives an explicit characterization of the domain $\mathbb{R}$ of ready sets of process graphs with multiple roots.

Definition 7.2 The *readiness domain* $\mathbb{R}$ is the set of subsets R of $Act^* \times \mathcal{P}(Act)$ satisfying

$$\begin{array}{ll} \mathrm{R1} & \exists X(\langle \varepsilon, X\rangle \in \mathrm{R}), \\ \mathrm{R2} & \exists X(\langle \sigma, X \cup \{a\}\rangle \in \mathrm{R}) \Leftrightarrow \exists \mathrm{Y}(\langle \sigma a, Y\rangle \in \mathrm{R}). \end{array}$$

Proposition 7.3 $\mathrm{R} \in \mathbb{R} \Leftrightarrow \exists g \in \mathbb{G}^{mr} : R(g) = \mathrm{R}$.
Proof: "$\Leftarrow$" is evident. For "$\Rightarrow$" let $\mathrm{R} \in \mathbb{R}$. Define the *canonical graph* $G(\mathrm{R})$ of R by

- $\textsc{nodes}(G(\mathrm{R})) = \mathrm{R}$,
- $\textsc{roots}(G(\mathrm{R})) = \{\langle \varepsilon, X\rangle \mid \langle \varepsilon, X\rangle \in \mathrm{R}\}$,
- $\textsc{edges}(G(\mathrm{R})) = \{(\langle \sigma, X\rangle, a, \langle \sigma a, Y\rangle) \mid \langle \sigma, X\rangle, \langle \sigma a, Y\rangle \in \textsc{nodes}(G(\mathrm{R})) \wedge a \in X\}$.

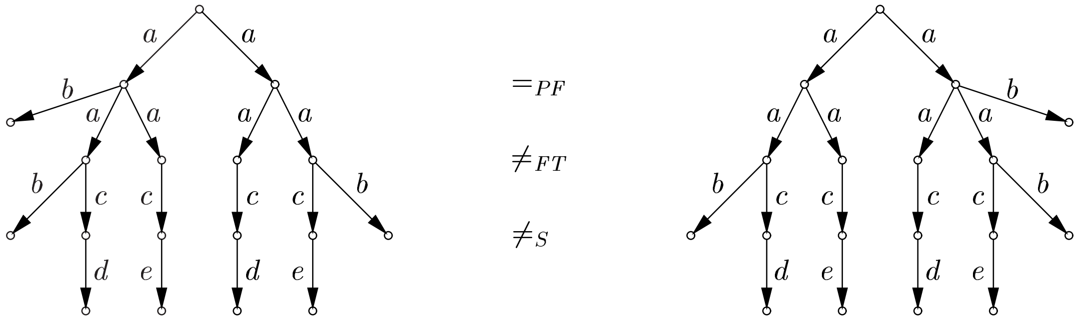
By R1, $\textsc{roots}(G(\mathrm{R})) \neq \emptyset$. Using R2, $G(\mathrm{R})$ is connected. Hence $G(\mathrm{R}) \in \mathbb{G}^{mr}$. Moreover, for every path $\pi \in \textsc{paths}(G(\mathrm{R}))$ one has $R(\pi) = \mathit{end}(\pi)$. From this it follows that $R(G(\mathrm{R})) = \mathrm{R}$. □

If $\textsc{roots}(g)$ would be allowed to be empty, a characterization is obtained by dropping requirement R1. A characterization of the domain of ready sets of process graphs with single roots is given by strengthening R1 to $\exists! X(\langle \varepsilon, X\rangle \in \mathrm{R})$, where $\exists! X$ means "there is exactly one X such that".

Possible-futures and acceptance-refusal semantics Readiness semantics was proposed by Olderog & Hoare [40]. Two preliminary versions stem from Rounds & Brookes [46]: in *possible-futures semantics* (PF) the menu consists of the entire trace set of the remaining behaviour of an idle process, instead of only the set of its initial actions; in *acceptance-refusal semantics* a menu may be any finite subset of initial actions, while also the finite refusal sets of Section 4 are observable.

Definition 7.3 $\langle \sigma, X\rangle \in Act^* \times \mathcal{P}(Act^*)$ is a *possible future* of a process p if there is a process q such that $p \xrightarrow{\sigma} q$ and $T(q) = X$. Let $PF(p)$ denote the set of possible futures of p. Two processes p and q are *possible-futures equivalent*, notation $p =_{PF} q$, if $PF(p) = PF(q)$.

The modal and process graph characterizations of possible-future semantics are straightforward, but a plausible testing scenario has not been proposed. Trivially $R \preceq PF$. That the reverse does not hold, and even that $PF \not\preceq RT$, will follow from Counterexample 10. Counterexample 7 shows that $FT \not\preceq PF$. There $PF(\mathit{left}) = PF(\mathit{right})$ but $FT(\mathit{left}) \neq FT(\mathit{right})$. As for the first statement, both graphs have 18 paths starting from the root, and hence 18 possible futures. These are easily seen to be the same at both sides; in the second graph only 2 possible futures swapped places. The second statement follows since $a\{b\}a\{b\}cd \in FT(\mathit{left}) - FT(\mathit{right})$. Thus possible-future semantics is incomparable with failure trace and ready trace semantics.



$a(b + a(b + cd) + ace) + a(acd + a(ce + b))$ $\qquad$ $a(a(b + cd) + ace) + a(acd + a(ce + b) + b)$

Counterexample 7: Possible-futures equivalent, but not failure trace or simulation equivalent

Definition 7.4 $\langle \sigma, X, Y \rangle \in \mathit{Act}^* \times \mathcal{P}(\mathit{Act}) \times \mathcal{P}(\mathit{Act})$ is an *acceptance-refusal triple* of a process p if X and Y are finite and there is a process q such that $p \xrightarrow{\sigma} q$, $X \subseteq I(q)$ and $Y \cap I(q) = \emptyset$. Let $AR(p)$ denote the set of acceptance-refusal triples of p. Two processes p and q are *acceptance-refusal equivalent*, notation $p =_{AR} q$, if $AR(p) = AR(q)$.

The modal and process graph characterizations are again straightforward. A motivating testing scenario would be the same as for readiness semantics, except that at any time only finitely many switches can be set on free, and only finitely many lamps can be investigated in a finite amount of time. Clearly $p =_R q \Rightarrow p =_{AR} q$, for

$$AR(p) = \{\langle \sigma, X, Y \rangle \mid \exists \langle \sigma, Z \rangle \in R(p) \mid X, Y \text{ finite } \wedge X \subseteq Z \wedge Y \cap Z = \emptyset\}.$$

That this implication is strict follows from Counterexample 4. It is not difficult to see that for finitely branching processes acceptance-refusal equivalence coincides with ready equivalence: $\langle \sigma, X \rangle$ is a ready pair of a process p iff p has an acceptance-refusal triple $\langle \sigma, X, Y \rangle$ with $X \cup Y = \mathit{Cont}_p(\sigma)$ (cf. Definition 4.1).

Infinite processes Note that if in Definition 7.4 the sets X and Y are allowed to be infinite the resulting equivalence would be ready equivalence again. Namely $\langle \sigma, X \rangle$ is a ready pair of a process p iff p has such an acceptance-refusal triple $\langle \sigma, X, \mathit{Act} - Y \rangle$. Thus acceptance-refusal semantics can be regarded as the finite variant of readiness semantics, and will therefore be denoted R^-. The infinitary variant of readiness semantics (R^∞), motivated by observations that may last forever, is defined analogously to F^∞:

Definition 7.5 p and q are *infinitary ready equivalent* if $R(p) = R(q)$ and $T^\infty(p) = T^\infty(q)$.

Clearly, $R \prec R^\infty$; by Counterexample 1 the inclusion is strict. Moreover, $F^\infty \prec R^\infty \prec RT^\infty$.

Proposition 7.4 Let p en q be image finite processes. Then $p =_R q \Leftrightarrow p =_R^\infty q$.
Proof: "$\Leftarrow$" has been established for all processes, and the second "$\Rightarrow$" follows immediately from Proposition 2.4 (as $p =_R q \Rightarrow p =_T q \Rightarrow p =_T^\infty q$). $\Box$

Proposition 7.5 Let $p, q \in \mathbb{P}$ and p is image finite. Then $p =_{AR} q \Leftrightarrow p =_R q$.
Proof: "$\Leftarrow$" holds for all process. I will prove "$\Rightarrow$" assuming that p has the property that for any $\sigma \in Act^*$ there are only finitely many ready pairs $\langle \sigma, X\rangle \in R(p)$. This property (call it *RIF*) is clearly implied by image finiteness. So suppose p has the RIF property and $AR(p) = AR(q)$. I will show that $R(p) = R(q)$.

Suppose $\langle \sigma, Y\rangle \notin R(p)$. By RIF there are only finitely many ready pairs $\langle \sigma, X_i\rangle \in R(p)$. For each of them choose an action $a_i \in Y - X_i$ or $b_i \in X_i - Y$. Let U be the set of all those a_i's, and V the set of the b_i's. Then $\langle \sigma, U, V\rangle \notin AR(p) = AR(q)$ and hence $\langle \sigma, Y\rangle \notin R(q)$.

It follows that $R(q) \subseteq R(p)$, and thus q has the property RIF as well. Now the same argument applies in the other direction, yielding $R(p) \subseteq R(q)$. $\Box$

Inspired by the definition of R^-, a finite version of ready trace semantics (RT^-) can be defined likewise. Here I will just give its modal characterization.

Definition 7.6 The set $\mathcal{L}_{RT}^-$ of *finite ready trace formulas* over Act is given by:

- $\top \in \mathcal{L}_{RT}^-$.
- If $\varphi \in \mathcal{L}_{RT}^-$ and $X \subseteq_{fin} Act$ then $X\varphi \in \mathcal{L}_{RT}^-$ and $\widetilde{X}\varphi \in \mathcal{L}_{RT}^-$.
- If $\varphi \in \mathcal{L}_{RT}^-$ and $a \in Act$ then $a\varphi \in \mathcal{L}_{RT}^-$.

The *satisfaction relation* $\models\, \subseteq \mathbb{P} \times \mathcal{L}_{RT}^-$ is given by the usual clauses for $\top$ and $a\varphi$, and:

- $p \models X\varphi$ if $X \subseteq I(p)$ and $p \models \varphi$.
- $p \models \widetilde{X}\varphi$ if $I(p) \cap X = \emptyset$ and $p \models \varphi$.

Processes p and q are *finite-ready trace equivalent*, notation $p =_{RT}^- q$, if $\forall \varphi \in \mathcal{L}_{RT}^-(p \models \varphi \Leftrightarrow q \models \varphi)$.

As these formulas are expressible in terms of the ones of Definition 6.2, one has $RT^- \prec RT$; Counterexample 4 shows that the inclusion is strict. Also $FT^- \prec RT^-$ and $F^- \prec R^- \prec RT^-$.

Proposition 7.6 Let $p, q \in \mathbb{P}$ and p is image finite. Then $p =_{RT}^- q \Leftrightarrow p =_{RT} q$.
Proof: "$\Leftarrow$" holds for all process. "$\Rightarrow$" follows just as in Proposition 7.5, using the property that for any $a_1 a_2 \cdots a_n \in Act^\omega$ there are only finitely many normal ready traces $X_0 a_1 X_1 a_2 \cdots a_n X_n \in RT_N(p)$. $\Box$

Unlike the semantics T to RT, possible-futures semantics distinguishes between the two processes of Counterexample 1: $\langle a, a^*\rangle \in PF(right) - PF(left)$. Still, $T^\infty \not\prec PF$, as can be seen from the variant of Counterexample 1 in which the left-hand process is appended to the endnodes of both processes. The so obtained systems have the same possible futures, including $\{\langle a^n, a^*\rangle \mid n \in \mathbb{N}\}$, but only the right-hand side has an infinite trace.

For the sake of completeness I include a definition of infinitary possible-futures semantics (PF^∞), such that $PF \prec PF^\infty$ and $R^\infty \prec PF^\infty$. A finite variant of PF has not been explored.

Definition 7.7 $\langle \sigma, X\rangle \in Act^* \times \mathcal{P}(Act^*)$ is an *infinitary possible future* of a process p if there is a process q such that $p \xrightarrow{\sigma} q$ and $T(q) \cup T^\infty(q) = X$. Let $PF^\infty(p)$ denote the set of infinitary possible futures of p. Two processes p and q are *infinitary possible-futures equivalent*, notation $p =_{PF}^\infty q$, if $PF^\infty(p) = PF^\infty(q)$.

8 Simulation semantics

The following concept of *simulation* occurs frequently in the literature (see e.g. PARK [41]).

Definition 8 A *simulation* is a binary relation R on processes, satisfying, for $a \in Act$:

- if pRq and $p \xrightarrow{a} p'$, then $\exists q' :\ q \xrightarrow{a} q'$ and $p'Rq'$.

Process p *can be simulated by* q, notation $p \hookrightarrow q$, if there is a simulation R with pRq.
p and q are *similar*, notation $p \rightleftarrows q$, if $p \hookrightarrow q$ and $q \hookrightarrow p$.

Proposition 8.1 Similarity is an equivalence relation on the domain of processes.
Proof: Symmetry is immediate, so it has to be checked that $p \hookrightarrow p$, and $p \hookrightarrow q \wedge q \hookrightarrow r \ \Rightarrow\ p \hookrightarrow r$.

- The identity relation is a simulation with pRp.
- If R is a simulation with pRq and S is a simulation with qSr, then the relation $R;S$, defined by $x(R;S)z$ iff $\exists y : xRy \wedge ySz$, is a simulation with $p(R;S)r$. □

Hence the relation will be called *simulation equivalence*. In *simulation semantics* (S) two processes are identified iff they are simulation equivalent.

Testing scenario and modal characterization The testing scenario for simulation semantics resembles that for trace semantics, but in addition the observer is, at any time during a run of the investigated process, capable of making arbitrary many copies of the process in its present state and observe them independently. Thus an observation yields a tree rather than a sequence of actions. Such a tree can be coded as an expression in a simple modal language.

Definition 8.1 The class $\mathcal{L}_S$ of *simulation formulas* over *Act* is defined recursively by:

- If I is a set and $\varphi_i \in \mathcal{L}_S$ for $i \in I$ then $\bigwedge_{i\in I} \varphi_i \in \mathcal{L}_S$.
- If $\varphi \in \mathcal{L}_S$ and $a \in Act$ then $a\varphi \in \mathcal{L}_S$.

The *satisfaction relation* $\models\ \subseteq \mathbb{P} \times \mathcal{L}_S$ is defined recursively by:

- $p \models \bigwedge_{i\in I} \varphi_i$ if $p \models \varphi_i$ for all $i \in I$.
- $p \models a\varphi$ if for some $q \in \mathbb{P}$: $p \xrightarrow{a} q$ and $q \models \varphi$.

Let $S(p)$ denote the class of simulation formulas satisfied by the process p: $S(p) = \{\varphi \in \mathcal{L}_S \mid p \models \varphi\}$.
Write $p \sqsubseteq_S q$ if $S(p) \subseteq S(q)$ and $p =_S q$ if $S(p) = S(q)$.

Write $\top$ for $\bigwedge_{i\in\emptyset} \varphi_i$, and $\varphi_1 \wedge \varphi_2$ for $\bigwedge_{i\in\{1,2\}} \varphi_i$. It turns out that $\mathcal{L}_T$ is a sublanguage of $\mathcal{L}_S$.

Proposition 8.2 $p \hookrightarrow q \Leftrightarrow p \sqsubseteq_S q$. Hence $p \rightleftarrows q \Leftrightarrow p =_S q$.
Proof: For "⇒" I have to prove that for any simulation R and for all $\varphi \in \mathcal{L}_S$ one has

$$pRq \Rightarrow (p \models \varphi \Rightarrow q \models \varphi).$$

I will do so with structural induction on φ. Suppose pRq.

- Let $p \models a\varphi$. Then there is a $p' \in \mathbb{P}$ with $p \xrightarrow{a} p'$ and $p' \models \varphi$. As R is a simulation, there must be a $q' \in \mathbb{P}$ with $q \xrightarrow{a} q'$ and $p'Rq'$. So by induction $q' \models \varphi$, and hence $q \models a\varphi$.
- $p \models \bigwedge_{i\in I} \varphi_i \Leftrightarrow \forall i \in I(p \models \varphi_i) \stackrel{\text{ind.}}{\Longrightarrow} \forall i \in I(q \models \varphi_i) \Leftrightarrow q \models \bigwedge_{i\in I} \varphi_i$.

For "$\Leftarrow$" it suffices to establish that $\sqsubseteq_S$ is a simulation.
Suppose $p \sqsubseteq_S q$ and $p \xrightarrow{a} p'$. I have to show that $\exists q' \in \mathbb{P}$ with $q \xrightarrow{a} q'$ and $p' \sqsubseteq_S q'$. Let Q' be

$$\{q' \in \mathbb{P} \mid q \xrightarrow{a} q' \wedge p' \not\sqsubseteq_S q'\}.$$

By Definition 1.1 Q' is a set. For every $q' \in Q'$ there is a formula $\varphi_{q'} \in S(p') - S(q')$. Now

$$a \bigwedge_{q' \in Q'} \varphi_{q'} \in S(p) \subseteq S(q),$$

so there must be a $q' \in \mathbb{P}$ with $q \xrightarrow{a} q'$ and $q' \notin Q'$, which had to be shown. $\Box$

Process graph characterization Simulation equivalence can also be characterized by means of relations between the nodes of two process graphs, rather than between process graphs themselves.

Definition 8.2 Let $g, h \in \mathbb{G}$. A *simulation* of g by h is a binary relation $R \subseteq \textsc{nodes}(g) \times \textsc{nodes}(h)$, satisfying:

- $\textsc{root}(g) R \textsc{root}(h)$.
- If sRt and $(s, a, s') \in \textsc{edges}(g)$, then there is an edge $(t, a, t') \in \textsc{edges}(h)$ such that $s'Rt'$.

This definition is illustrated in Figure 5. Solid lines indicates what is assumed, dashed lines what is required. It follows easily that $g \hookrightarrow h$ iff there exists a simulation of g by h.

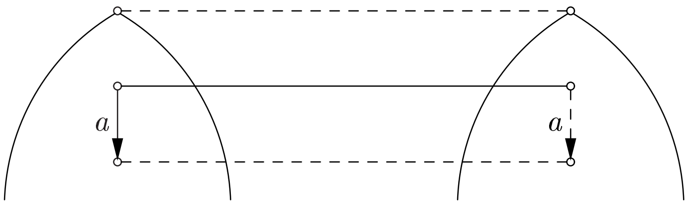
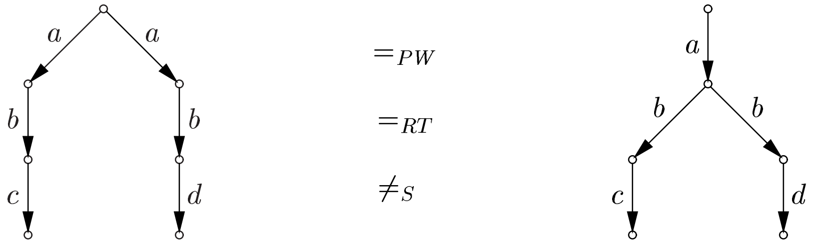


Figure 5: A simulation

For process graphs with multiple roots, the first requirement of Definition 8.2 generalizes to

- $\forall s \in \textsc{roots}(g)\, \exists t \in \textsc{roots}(h) : sRt$.

Classification Simulation semantics (S) is finer than trace semantics ($T \prec S$), but independent of CT, F, R, FT, RT and PF.
Proof: "$T \preceq S$" follows since $\mathcal{L}_T$ is a sublanguage of $\mathcal{L}_S$.
"$S \not\preceq CT$" (and hence "$S \not\preceq RT$", "$S \not\preceq PF$" etc.) follows from Counterexample 2. There *left* $\neq_{CT}$ *right*, although *left* $\rightleftarrows$ *right*; the construction of the two simulations is left to the reader.
"$S \not\succeq RT$" (and hence "$S \not\succeq T$" etc.) follows from Counterexample 8. There $RT(\mathit{left}) = RT(\mathit{left})$, but $S(\mathit{left}) \neq S(\mathit{left})$. The first statement follows from Proposition 6.3 and the insight that it suffices to check the two ready traces contributed by the maximal paths; these are the same for both graphs. The second statement follows since $a(bc\top \wedge bd\top) \in S(\mathit{right}) - S(\mathit{left})$.
"$S \not\succeq PF$" follows from Counterexample 7, where $PF(\mathit{left}) = PF(\mathit{right})$ but $S(\mathit{left}) \neq S(\mathit{left})$. The latter statement follows since $a(b\top \wedge a(b\top \wedge cd\top)) \in S(\mathit{left}) - S(\mathit{right})$. $\Box$



$abc + abd$ $a(bc + bd)$

Counterexample 8: Possible worlds and ready trace equivalent, but not simulation equivalent

Infinite processes In order to make the testing scenario match its formalization in terms of the modal language $\mathcal{L}_S$ even for infinite processes, one has to assume that the amount of copies one can make at any time is infinite. Moreover, although no single copy can be tested forever, due to its infinite branching there may be no upperbound upon the duration of an observation.

One might consider an even more infinitary testing scenario by allowing observations to go on forever on some or all of the copies. However, this would not give rise to a more discriminating equivalence; ordinary simulation equivalences already preserves infinite traces.

Proposition 8.3 If $p \overset{\subset}{\rightarrow} q$ then $T^\infty(p) \subseteq T^\infty(q)$. Hence $T^\infty \prec S$.
Proof: Suppose R is a simulation with $p_0 R q_0$ and $a_1 a_2 \cdots \in T^\infty(p_0)$. Then there are $p_1, p_2, \ldots$ such that $p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} \cdots$. With induction on $i \in \mathbb{N}$ it follows that there are processes q_{i+1} such that $q_i \xrightarrow{a_{i+1}} q_{i+1}$ and $p_{i+1} R q_{i+1}$. Hence $a_1 a_2 \cdots \in T^\infty(q_0)$. □

The most radical way to make the testing scenario finitary, is to allow only finitely many copies to be made in any state of the process. This also puts an upperbound on the duration of any observation. Observations can now be modelled with simulation formulas in which the index sets I of the first clause of Definition 8.1 are always finite. The modal language containing such simulation formulas can equivalently be defined by splitting the construction $\bigwedge_{i \in I}$ into $\top$ and $\wedge$.

Definition 8.3 The set $\mathcal{L}_S^*$ of *finitary simulation formulas* over *Act* is defined recursively by:

- $\top \in \mathcal{L}_S^*$.
- If $\varphi, \psi \in \mathcal{L}_S^*$ then $\varphi \wedge \psi \in \mathcal{L}_S^*$.
- If $\varphi \in \mathcal{L}_S^*$ and $a \in Act$ then $a\varphi \in \mathcal{L}_S^*$.

The *satisfaction relation* $\models \subseteq \mathbb{P} \times \mathcal{L}_S^*$ is defined recursively by:

- $p \models \top$ for all $p \in \mathbb{P}$.
- $p \models \varphi \wedge \psi$ if $p \models \varphi$ and $p \models \psi$.
- $p \models a\varphi$ if for some $q \in \mathbb{P}$: $p \xrightarrow{a} q$ and $q \models \varphi$.

Let $S^*(p)$ denote the set of all finitary simulation formulas that are satisfied by the process p: $S^*(p) = \{\varphi \in \mathcal{L}_S^* \mid p \models \varphi\}$. Two processes p and q are *finitary simulation equivalent*, notation $p =_S^* q$, if $S^*(p) = S^*(q)$.

In contrast, the equivalence $\rightleftarrows$ of Definition 8 is then *infinitary simulation equivalence*. Note however, that contrary to the previous equivalences surveyed, the default version (the one meant when leaving out the adjective "finitary" or "infinitary") is the infinitary one. In general, I use the superscript $*$ for finitary versions and ∞ for infinitary versions. However, for the trace oriented equivalences (Sections 2–7) I leave out the $*$, and for the simulation oriented equivalences (Sections 8–12) I leave out the ∞.

The next proposition, and hence also the essence of Proposition 8.2, stems from in HENNESSY & MILNER [28]. It states that for image finite processes finitary and infinitary simulation equivalence coincide.

Proposition 8.4 Let $p, q \in \mathbb{P}$ be image finite processes. Then $p \rightleftarrows q \Leftrightarrow p =_S^* q$.

Proof: Exactly as the proof of Proposition 8.2, but for "$\Leftarrow$" one shows that the relation $\sqsubseteq_S^{i.f.}$ given by $p \sqsubseteq_S^{i.f.} q$ iff $p \sqsubseteq_S q$ and q is image finite is a simulation, using that, as q is image finite, Q' must be finite. $\square$

In fact, this proposition is a special case of the following one, which is proved likewise.

Proposition 8.5 Let $S_\kappa(p)$ denote the set of all simulation formulas satisfied by p in which all index sets have cardinality less than κ. Let $p, q \in \mathbb{P}$ and assume $|\{q' \mid q \xrightarrow{\sigma} q'\}| < \kappa$ for each $\sigma \in Act^*$. Then $p \stackrel{\subset}{\rightarrow} q \Leftrightarrow S_\kappa(p) \subseteq S_\kappa(q)$. $\square$

Although only q needs to be image finite in order to obtain $p \stackrel{\subset}{\rightarrow} q \Leftrightarrow p \sqsubseteq_S^* q$, Counterexample 12 will show that *both* p and q need to be image finite in the statement of Proposition 8.4.

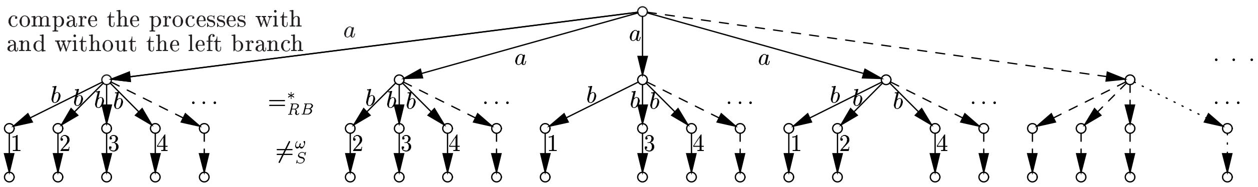
A less radical way to finitize the testing scenario for simulation semantics is to allow infinitely many copies to be made in any state of the process, but put a finite upperbound on the duration of any observation. Observations can then be modelled with simulation formulas in which the index sets can be arbitrary, but there is a finite upperbound on the nesting of the construction $a\varphi$ of the second clause of Definition 8.1.

Definition 8.4 Let $\mathcal{L}_S^\omega = \bigcup_{n=0}^\infty \mathcal{L}_S^n$, where $\mathcal{L}_S^n$ is given by:

- If I is a set and $\varphi_i \in \mathcal{L}_S^n$ for $i \in I$ then $\bigwedge_{i \in I} \varphi_i \in \mathcal{L}_S^n$.
- If $\varphi \in \mathcal{L}_S^n$ and $a \in Act$ then $a\varphi \in \mathcal{L}_S^{n+1}$.

Let $S^\omega(p) = \{\varphi \in \mathcal{L}_S^\omega \mid p \models \varphi\}$ and write $p =_S^\omega q$ if $S^\omega(p) = S^\omega(q)$.

Now $p =_S q \Rightarrow p =_S^\omega q \Rightarrow p =_S^* q$, and for image finite processes all three equivalences coincide. For image infinite processes both implications are strict, as illustrated by Counterexamples 9 and 1.



Counterexample 9: Finitary equivalent, but not S^ω-equivalent

In Counterexample 9 $S^*(with) = S^*(without)$, yet $a \bigwedge_{i=1}^\infty bi\top \in S^\omega(with) - S^\omega(without)$.

In Counterexample 1 $S^\omega(left) = S^\omega(right)$, yet $right \not\stackrel{\subset}{\rightarrow} left$. For the first statement, let $\varphi \in \mathcal{L}_S^\omega$. Then there is an n such that $\varphi \in \mathcal{L}_S^n$. Now parts of trees that are further than n edges away from the root play no rôle in the satisfaction relation for φ. Thus, the validity of φ remains unchanged if in both trees all paths are cut off after n steps. However, the cut versions of both trees are isomorphic, and hence satisfy the same formulas (cf. Corollary 12.1). The second statement follows immediately from Proposition 8.3.

It follows that $T \prec S^* \prec S^\omega \prec S$ and $T \prec T^\infty \prec S$, whereas T^∞ is incomparable with S^* and S^ω. Moreover, S^*, S^ω and S are incomparable with the semantics ranging from CT or F^- to RT^∞.

9 Ready simulation semantics

Testing scenario Of course one can also combine the copying facility with any of the other testing scenarios. The observer can then plan experiments on one of the machines from the Sections 3 to 7 together with a *replicator*, an ingenious device by which one can replicate the machine whenever and as often as one wants. In order to represent observations, the modal languages from Sections 3 to 7 need to be combined with the one from Section 8.

Definition 9 The language $\mathcal{L}_{CS}$ and the corresponding satisfaction relation is defined recursively by combining the clauses of Definition 3.1 (for $\mathcal{L}_{CT}$) with those of Definition 8.1 (for $\mathcal{L}_S$). Likewise, $\mathcal{L}_{FS}$ is obtained by combining $\mathcal{L}_F$ and $\mathcal{L}_S$; $\mathcal{L}_{FTS}$ by combining $\mathcal{L}_{FT}$ and $\mathcal{L}_S$; $\mathcal{L}_{RS}$ by combining $\mathcal{L}_R$ and $\mathcal{L}_S$; and $\mathcal{L}_{RTS}$ by combining $\mathcal{L}_{RT}$ and $\mathcal{L}_S$. For $p \in \mathbb{P}$ and $\mathcal{O} \in \{CS, FS, FTS, RS, RTS\}$ let $\mathcal{O}(p) = \{\varphi \in \mathcal{L}_\mathcal{O} \mid p \models \varphi\}$. Two processes $p, q \in \mathbb{P}$ are

- *completed simulation equivalent*, notation $p =_{CS} q$, if $CS(p) = CS(q)$;
- *failure simulation equivalent*, notation $p =_{FS} q$, if $FS(p) = FS(q)$;
- *failure trace simulation equivalent*, notation $p =_{FTS} q$, if $FTS(p) = FTS(q)$;
- *ready simulation equivalent*, notation $p =_{RS} q$, if $RS(p) = RS(q)$;
- *ready trace simulation equivalent*, notation $p =_{RTS} q$, if $RTS(p) = RTS(q)$.

It is obvious that failure trace simulation equivalence coincides with failure simulation equivalence and ready trace simulation equivalence with ready simulation equivalence ($p \models X\varphi \Leftrightarrow p \models X \wedge \varphi$). Also it is not difficult to see that failure simulation equivalence and ready simulation equivalence coincide ($p \models X \Leftrightarrow p \models \widetilde{Y} \wedge \bigwedge_{a \in X} a\top$, where $Y = Act - X$). So one has

Proposition 9.1 $p =_{FS} q \Leftrightarrow p =_{FTS} q \Leftrightarrow p =_{RTS} q \Leftrightarrow p =_{RS} q$. □

Relational characterizations The two remaining equivalences can be characterized as follows:

Definition 9.1 A *complete simulation* is a binary relation R on processes, satisfying, for $a \in Act$:

- if pRq and $p \xrightarrow{a} p'$, then $\exists q' :\ q \xrightarrow{a} q'$ and $p'Rq'$;
- if pRq then $I(p) = \emptyset \Leftrightarrow I(q) = \emptyset$.

Proposition 9.2 Two processes p and q are completed simulation equivalent if there exists a complete simulation R with pRq and a complete simulation S with qSp.
Proof: A trivial modification of the proof of Proposition 8.2. □

Definition 9.2 A *ready simulation* is a binary relation R on processes, satisfying, for $a \in Act$:

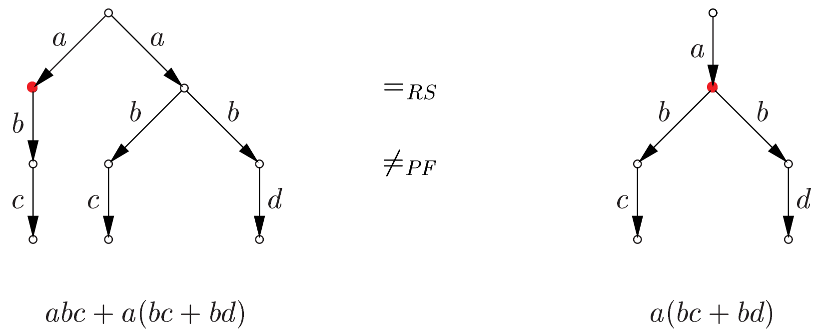
- if pRq and $p \xrightarrow{a} p'$, then $\exists q' :\ q \xrightarrow{a} q'$ and $p'Rq'$;
- if pRq then $I(p) = I(q)$.

Proposition 9.3 Two processes p and q are ready simulation equivalent if there exists a ready simulation R with pRq and a ready simulation S with qSp.
Proof: A trivial modification of the proof of Proposition 8.2. □

A variant of ready simulation equivalence was originally proposed by BLOOM, ISTRAIL & MEYER [12] under the name *GSOS trace congruence*; they provided a modal characterization, to be discussed in Section 10. A relational characterization was first given by LARSEN & SKOU [35] under the name $\frac{2}{3}$-*bisimulation equivalence*. A $\frac{2}{3}$-bisimulation is defined just like a ready simulation, except that the second clause reads "if pRq and $\exists q' : q \xrightarrow{a} q'$ then $\exists p' : p \xrightarrow{a} p'$". This is clearly equivalent.

Classification $RT \prec RS$, $CT \prec CS$ and $S \prec CS \prec RS$. CS is independent of F to RT.
Proof: "$RT \preceq RS$" follows since $\mathcal{L}_{RT}$ is a sublanguage of $\mathcal{L}_{RTS}$, using Proposition 9.1.
"$CT \preceq CS$" and "$S \preceq CS \preceq RS$" follow since $\mathcal{L}_{CT}$ and $\mathcal{L}_S$ are sublanguages of $\mathcal{L}_{CS}$, which is a sublanguage of $\mathcal{L}_{FS}$.
"$RT \not\succeq RS$" follows from Counterexample 8, using "$RS \succeq S$"; similarly $RT \not\succeq CS$ and $CT \not\succeq CS$.
"$S \not\succeq CS$" follows from Counterexample 2, using "$CS \succeq CT$".
"$CS \not\succeq F$" (and hence "$CS \not\succeq RS$") follows from Counterexample 3, in which $F(\textit{left}) \neq F(\textit{right})$ but $\textit{left} =_{CS} \textit{right}$; the construction of the two complete simulations is left to the reader. □

Proposition 9.4 PF is incomparable with CS and RS.
Proof: "$CS \not\preceq PF$" (and hence "$RS \not\preceq PF$") follows from Counterexample 7, using "$CS \succeq S$".



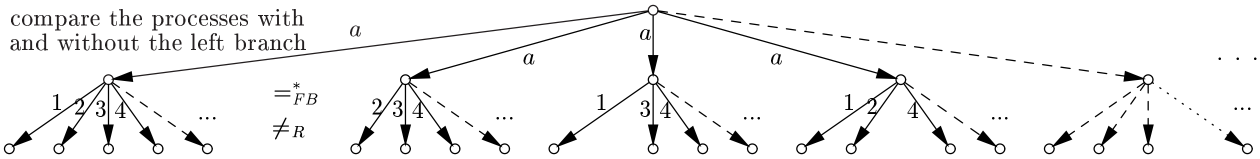
Counterexample 10: Ready simulation equivalent, but not possible-futures equivalent

"$RS \not\succeq PF$" (and hence "$CS \not\succeq PF$") follows from Counterexample 10, which shows two graphs that are ready simulation equivalent but not possible-futures equivalent. Concerning the first claim, note that there exists exactly one simulation of *right* by *left*, namely the one mapping *right* on the right-hand side of *left*. There also exists exactly one simulation of *left* by *right*, which relates the red (or shaded) node on the *left* to the red (or shaded) shaded node on the *right*. Both simulations are ready simulations, as related nodes have the same menu of initial actions. The second claim follows since $\langle a, \{\varepsilon,\ b,\ bc\}\rangle \in PF(\textit{left}) - PF(\textit{right})$. □

Infinite processes For each of the semantics *CS, FS, FTS, RS* and *RTS* a finitary variant (superscripted with a *), motivated by allowing finite replication only, is defined by combining the modal languages $\mathcal{L}_{CT}$, $\mathcal{L}_F$, $\mathcal{L}_{FT}$, $\mathcal{L}_R$ and $\mathcal{L}_{RT}$, respectively, with $\mathcal{L}_S^*$. Likewise, an intermediate variant (superscripted with an ω), motivated by requiring any observation to be over within a finite amount of time, is defined by combining these languages with $\mathcal{L}_S^\omega$. Finally, a finite variant (superscripted with a $-$), motivated by observers that can only engage in finite replication, can only set finitely many switches on free, and can only inspect finitely many lamps in a finite time, is obtained by combining the (obvious) modal languages $\mathcal{L}_F^-$, $\mathcal{L}_{FT}^-$, $\mathcal{L}_R^-$ and $\mathcal{L}_{RT}^-$ with $\mathcal{L}_S^*$ (there is no CS^-). Exactly as in the case of Proposition 9.1 one finds:

Proposition 9.5 $FS^\omega = FTS^\omega = RTS^\omega = RS^\omega$ and $FS^- = FTS^- = RTS^- = RS^-$. Moreover, $FS^* = FTS^*$ and $RTS^* = RS^*$. □

However, as pointed out in SCHNOEBELEN [47], FS^* and RS^* are different: in Counterexample 11



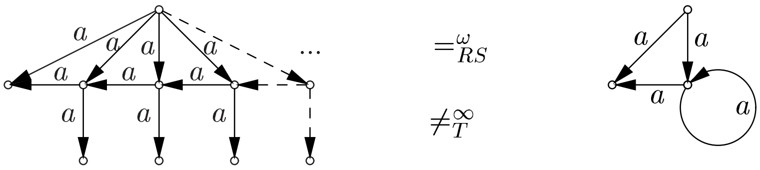
Counterexample 11: Finitary failure simulation equivalent, but not ready equivalent

one has $FS^*(\mathit{with}) = FS^*(\mathit{without})$, but $\langle a, \{1,2,...\}\rangle \in R(\mathit{with}) - R(\mathit{without})$.

Clearly one has $CS^* \prec CS^\omega \prec CS$ and $RS^- \prec FS^* \prec RS^* \prec RS^\omega \prec RS$. The strictness of these inclusions is given by Counterexamples 4, 11, 9 and 1. In addition one has $RT^- \prec RS^-$, $S^* \prec RS^-$, $RT \prec RS^*$, $FT \prec FS^*$, $CT \prec CS^*$ and $S^* \prec CS^* \prec FS^*$; as well as $RT^\infty \prec RS$, $CT^\infty \prec CS$, $S^\omega \prec CS^\omega \prec RS^\omega$ and $S \prec CS \prec RS$. Counterexamples against further inclusions have already been provided.

Proposition 9.6 Let $p, q \in \mathbb{P}$ be image finite. Then $p =_{CS} q \Leftrightarrow p =^*_{CS} q$ and $p =_{RS} q \Leftrightarrow p =^-_{RS} q$.
Proof: Two trivial modifications of the proof of Proposition 8.4. In the second one, one uses that if $\forall \varphi \in \mathcal{L}^-_{RS}(p \models \varphi \Rightarrow q \models \varphi)$ then surely $I(p) = I(q)$. □

In fact, if it is merely known that only q is image finite it follows already that $p \sqsubseteq_{CS} q \Leftrightarrow p \sqsubseteq^*_{CS} q$ and $p \sqsubseteq_{RS} q \Leftrightarrow p \sqsubseteq^-_{RS} q$. However, the following variant of Counterexample 1 shows that in the statement of Proposition 9.6 it is essential that *both* p and q are image finite. In Counterexample 12 *right* is image finite—in fact, it is even finitely branching—but *left* is not. It turns out that *left* $=^\omega_{RS}$ *right* (and hence *left* $=^-_{RS}$ *right*, *left* $=^*_{CS}$ *right*, *left* $=_{RT}$ *right*, *left* $=_F$ *right*, etc.) but *left* $\neq^\infty_T$ *right* (and hence *left* $\neq^\infty_F$ *right*, *left* $\neq^\infty_{RT}$ *right*, *left* $\neq_{CS}$ *right*, *left* $\neq_{RS}$ *right*, etc.).



Counterexample 12: Finitary ready simulation equivalent but not infinitary equivalent

For general (non-image-finite) processes, no relational characterizations of the finite, finitary and intermediate equivalences are known.

Testing scenario An alternative and maybe more natural testing scenario for finitary ready simulation semantics (or simulation semantics) can be obtained by exchanging the replicator for an *undo*-button on the (ready) trace machine (Figure 6). It is assumed that all intermediate states

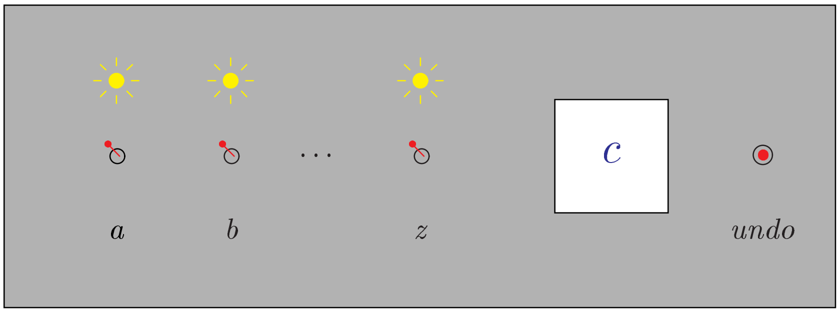


Figure 6: The ready simulation machine

that are past through during a run of a process are stored in a memory inside the black box. Now pressing the *undo*-button causes the machine to shift one state backwards. In the initial state pressing the button has no effect. An observation now consists of a (ready) trace, enriched with *undo*-actions. Such observations can easily be translated into finitary (ready) simulation formulas.

10 Reactive versus generative testing scenarios

In the testing scenarios presented so far, a process is considered to perform actions and make choices autonomously. The investigated behaviours can therefore be classified as *generative processes*. The observer merely restricts the spontaneous behaviour of the generative machine by cutting off some possible courses of action. An alternative view of the investigated processes can be obtained by considering them to react on stimuli from the environment and be passive otherwise. *Reactive machines* can be obtained out of the generative machines presented so far by replacing the switches by buttons and the display by a green light. Initially the process waits patiently until the observer

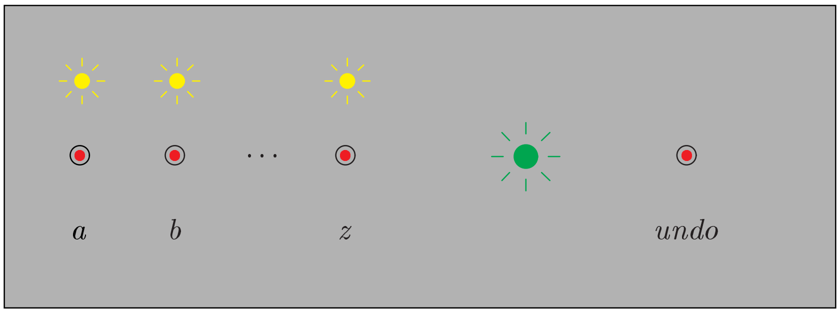


Figure 7: The reactive ready simulation machine

tries to press one of the buttons. If the observer tries to press an a-button, the machine can react in two different ways: if the process can not start with an a-action the button will not go down and the observer may try another one; if the process can start with an a-action it will do so and the button goes down. Furthermore the green light switches on. During the execution of a no buttons can be pressed. As soon as the execution of a is completed the light switches off, so that the observer knows that the process is ready for a new trial. Reactive machines as described above originate from MILNER [37, 38].

One family of testing scenarios with reactive machines can be obtained by allowing the observer to try to depress more than one button at a time. In order to influence a particular choice, the observer could already start exercising pressure on buttons during the execution of the preceding action (when no button can go down). When this preceding action is finished, at most one of the buttons will go down. These testing scenarios are equipotent with the generative ones: putting pressure on a button is equivalent to setting the corresponding switch on 'free'; moreover an action a appearing in the display is mimicked by the a-button going down, and the disappearance of a from the display by the green light going off.

Another family of testing scenarios is obtained by allowing the user to try only one button at a time. They are equipotent with those generative testing scenarios in which at any time only one switch can be set on 'free'. Next I will discuss the equivalences that originate from these scenarios.

First consider the reactive machine that resembles the failure trace machine, thus without menu-lights and *undo*-button. An observation on such a machine consists of a sequence of accepted and refused actions, indicating which buttons went down in a sequence of trials of the user. Such a sequence can be seen as a failure trace where all refusal sets are singletons. Call the resulting semantics FT^1. Clearly, the failure trace set of any process p satisfies

$$\sigma(X \cup Y)\rho \in FT(p) \;\Leftrightarrow\; \sigma XY\rho \in FT(p).$$

Thus, any failure trace $\sigma\{a_1, \ldots, a_n\}\rho$ can be rewritten as (contains the same information as) $\sigma\{a_1\}\{a_2\}\cdots\{a_n\}\rho$. It follows that the singleton-failure trace set $FT^1(p)$ of a process p contains as much information as its finite-failure trace set $FT^-(p)$, so the semantics FT^1 coincides with FT^-.

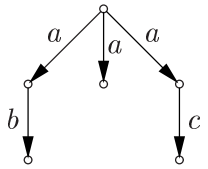
In order to arrive at a reactive counterpart to failures semantics, one could suppose that an observer continues an experiment only as long as all buttons he tries to depress actually go down; when a button refuses to go down, he will not try another one. This testing scenario gives rise to the variant F^1 of failures semantics in which all refusal sets are singletons.

Definition 10 $\langle\sigma, a\rangle \in Act^* \times Act$ is a *singleton-failure pair* of a process p if there is a process q such that $p \xrightarrow{\sigma} q$ and $a \notin I(q)$. Let $F^1(p)$ denote the set of singleton-failure pairs of p. Two processes p and q are *singleton-failures equivalent*, $p =^1_F q$, if $T(p) = T(q)$ and $F^1(p) = F^1(q)$.

Unlike for F and F^-, $F^1(p) = F^1(q)$ does not always imply that $T(p) = T(q)$, so one has to keep track of traces explicitly. These model observations ended by the observer before stagnation occurs.

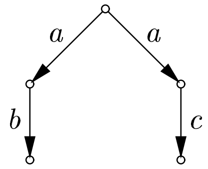
Singleton-failures semantics (F^1) is situated strictly between trace (T) and finite-failures semantics (F^-). For Counterexample 2 shows two processes with $T(\mathit{left}) = T(\mathit{right})$ but $\langle a, b\rangle \in F^1(\mathit{left}) - F^1(\mathit{right})$, and Counterexample 13 shows two processes with $F^1(\mathit{left}) = F^1(\mathit{right})$ (both contain $\langle a, b\rangle$ and $\langle a, c\rangle$), but $\langle a, \{b, c\}\rangle \in F(\mathit{left}) - F(\mathit{right})$. Furthermore, F^1 is independent of CT, S and CS, for in Counterexample 13 one has $CT(\mathit{left}) \neq CT(\mathit{right})$, in Counterexample 8 one has $\mathit{left} =^1_F \mathit{right}$ but $\mathit{left} \neq_S \mathit{right}$, and in Counterexample 3 one has $\mathit{left} =_{CS} \mathit{right}$ but $\langle a, c\rangle \in F^1(\mathit{left}) - F^1(\mathit{right})$.

Adding the *undo*-button to the reactive failure trace machine gives a semantics FS^1 characterized by the modal language $\mathcal{L}^*_S$ to which has been added a modality "Can't(a)", with $p \models$ Can't(a) iff $a \notin I(p)$. This modality denotes a failed attempt to depress the a-button. If fact, BLOOM, ISTRAIL & MEYER studied the coarsest equivalence finer than trace equivalence that is a congruence



$ab + a + ac$

$=^1_F$ $\quad \neq_{CT}$ $\quad \neq^-_F$



$ab + ac$

Counterexample 13: Singleton-failures equivalent, but not completed trace or failures equivalent

for the class of so-called *GSOS-operators*, and characterized this *GSOS trace congruence* by the modal language above; its formulas were called *denial formulas*. As in the modal language $\mathcal{L}^-_{FS}$ one has $p \models \widetilde{X \cup Y} \Leftrightarrow p \models \widetilde{X} \wedge \widetilde{Y}$, and Can't$(a)$ is the same as $\widetilde{\{a\}}$, it follows that the language of denial formulas is equally expressive as $\mathcal{L}^-_{FS}$, and hence FS^1 coincides with FS^- and RS^-.

If the menu-lights are added to the reactive failure trace machine considered above one can observe ready trace sets, and the green light is redundant. Likewise, adding menu-lights to the reactive failure scenario would give readiness semantics, and adding them to the reactive failure simulation machine would yield ready simulation. If the green light (as well as the menu-lights) are removed from the reactive failure trace machine, one can only test trace equivalence, since any refusal may be caused by the last action not being ready yet. Likewise, removing the green light from the reactive failure simulation machine (with *undo*-button) yields (finitary) simulation semantics. Reactive machines on which only one button at a time is depressed appear to be unsuited for testing completed trace, completed simulation and failures equivalence.

11 2-nested simulation semantics

2-nested simulation equivalence popped up naturally in GROOTE & VAANDRAGER [25] as the coarsest congruence with respect to a large and general class of operators that is finer than completed trace equivalence.

Definition 11 A *2-nested simulation* is a simulation contained in simulation equivalence ($\rightleftarrows$). Two processes p and q are *2-nested simulation equivalent*, notation $p =_{2S} q$, if there exists a 2-nested simulation R with pRq and a 2-nested simulation S with qSp.

Modal characterization A modal characterization of this notion is obtained by the fragment of the infinitary Hennessy-Milner logic (cf. Definition 12.1) without nested negations.

Definition 11.1 The class $\mathcal{L}_{2S}$ of *2-nested simulation formulas* over *Act* is defined recursively by:

- If I is a set and $\varphi_i \in \mathcal{L}_{2S}$ for $i \in I$ then $\bigwedge_{i\in I} \varphi_i \in \mathcal{L}_{2S}$.
- If $\varphi \in \mathcal{L}_{2S}$ and $a \in Act$ then $a\varphi \in \mathcal{L}_{2S}$.
- If $\varphi \in \mathcal{L}_S$ then $\neg\varphi \in \mathcal{L}_{2S}$.

Note that $\mathcal{L}_S \subseteq \mathcal{L}_{2S}$. The *satisfaction relation* $\models \subseteq \mathbb{P} \times \mathcal{L}_{2S}$ is defined recursively by:

- $p \models \bigwedge_{i\in I} \varphi_i$ if $p \models \varphi_i$ for all $i \in I$.
- $p \models a\varphi$ if for some $q \in \mathbb{P}$: $p \xrightarrow{a} q$ and $q \models \varphi$.
- $p \models \neg\varphi$ if $p \not\models \varphi$.

Proposition 11.1 $p =_{2S} q \;\Leftrightarrow\; \forall\varphi \in \mathcal{L}_{2S}(p \models \varphi \Leftrightarrow q \models \varphi)$.
Proof: A trivial modification of the proof of Proposition 8.2. □

Testing scenario In order to obtain a testing scenario for this equivalence one has to introduce the rather unnatural notion of a *lookahead* [25]: The *2-nested simulation machine* is a variant of the ready trace machine with replicator, where in an idle state the machine not only tells which actions are on the menu, but even which simulation formulas are (not) satisfied in the current state.

Classification $RS \prec 2S$ and $PF \prec 2S$.
Proof: For "$RS \preceq 2S$" it suffices to show that each 2-nested simulation is a ready simulation. This follows since $p \rightleftarrows q \Rightarrow I(p) = I(q)$. $PF \prec 2S$ is easily established using that $T \prec S$. That both inclusions are strict follows immediately from the fact that RS and PF are incomparable (Proposition 9.4). □

Infinite processes Exactly as for ready simulations semantics, 5 versions of 2-nested simulation semantics can be defined that differ for infinite processes. $2S^-$ is the semantics whose modal characterization has the constructs $\top$, $\wedge$, $a\varphi$ and $\neg\varphi'$ with $\varphi' \in \mathcal{L}^*_S$. The constructs $\widetilde{X}$ and X for $X \subseteq_{fin} Act$ are expressible in this logic. $F2S^*$ additionally has the construct $\widetilde{X}$, and $R2S^*$ the construct X, for $X \subseteq Act$. Finally $2S^\omega$ is characterized by the class of 2-nested simulation formulas with a finite upperbound on the nesting of the $a\varphi$ construct. The constructs $\widetilde{X}$ and X for $X \subseteq Act$ are expressible in $\mathcal{L}^\omega_{2S}$, and hence also in $\mathcal{L}_{2S}$.

We have $2S^- \prec F2S^* \prec R2S^* \prec 2S^\omega \prec 2S$. The strictness of these inclusions is given by Counterexamples 4, 11, 9 and 1. In addition one has $RS^- \prec 2S^-$, $FS^* \prec F2S^*$, $RS^* \prec R2S^*$, $RS^\omega \prec 2S^\omega$ and $RS \prec 2S$; as well as $PF^\infty \prec 2S$. Counterexample 1 shows that $PF \not\prec 2S^\omega$: $2S^\omega(\mathit{left}) = 2S^\omega(\mathit{right})$ (cf. Proposition 12.10), but $\langle a, a^*\rangle \in PF(\mathit{right}) - PF(\mathit{left})$.

Proposition 11.2 Let $p, q \in \mathbb{P}$ be image finite. Then $p =_{2S} q \Leftrightarrow p =^-_{2S} q$.
Proof: An easy modification of the proof of Proposition 8.4, also using its result. □

12 Bisimulation semantics

The concept of *bisimulation equivalence* stems from MILNER [37]. Its formulation below is due to PARK [41].

Definition 12 A *bisimulation* is a binary relation R on processes, satisfying, for $a \in Act$:

- if pRq and $p \xrightarrow{a} p'$, then $\exists q' : \; q \xrightarrow{a} q'$ and $p'Rq'$;
- if pRq and $q \xrightarrow{a} q'$, then $\exists p' : \; p \xrightarrow{a} p'$ and $p'Rq'$.

Two processes p and q are *bisimilar*, notation $p \leftrightarroweq q$, if there exists a bisimulation R with pRq.

The relation $\leftrightarroweq$ is again a bisimulation. As for similarity, one easily checks that bisimilarity is an equivalence relation on $\mathbb{P}$. Hence the relation will be called *bisimulation equivalence*. In *bisimulation semantics* (B) two processes are identified iff they are bisimulation equivalent. Note that the concept of bisimulation does not change if in the definition above the action relations $\xrightarrow{a}$ were replaced by generalized action relations $\xrightarrow{\sigma}$.

Modal characterization

Definition 12.1 The class $\mathcal{L}_B$ of *infinitary Hennessy-Milner formulas* over *Act* is defined by:

- If I is a set and $\varphi_i \in \mathcal{L}_B$ for $i \in I$ then $\bigwedge_{i\in I} \varphi_i \in \mathcal{L}_B$.
- If $\varphi \in \mathcal{L}_B$ and $a \in Act$ then $a\varphi \in \mathcal{L}_B$.
- If $\varphi \in \mathcal{L}_B$ then $\neg\varphi \in \mathcal{L}_B$.

The *satisfaction relation* $\models \subseteq \mathbb{P} \times \mathcal{L}_B$ is defined recursively by:

- $p \models \bigwedge_{i\in I} \varphi_i$ if $p \models \varphi_i$ for all $i \in I$.
- $p \models a\varphi$ if for some $q \in \mathbb{P}$: $p \xrightarrow{a} q$ and $q \models \varphi$.
- $p \models \neg\varphi$ if $p \not\models \varphi$.

Let $B(p)$ denote the class of all infinitary Hennessy-Milner formulas satisfied by the process p: $B(p) = \{\varphi \in \mathcal{L}_B \mid p \models \varphi\}$. Write $p \sqsubseteq_B q$ if $B(p) \subseteq B(q)$ and $p =_B q$ if $B(p) = B(q)$.

Proposition 12.1 $p \sqsubseteq_B q \Leftrightarrow p =_B q$.
Proof: If $\varphi \in B(q) - B(p)$ then $\neg\varphi \in B(p) - B(q)$. □

Proposition 12.2 $p \leftrightarroweq q \Leftrightarrow p =_B q$.
Proof: For "$\Rightarrow$" I have to prove that for any bisimulation R and for all $\varphi \in \mathcal{L}_B$ one has

$$pRq \Rightarrow (p \models \varphi \Leftrightarrow q \models \varphi).$$

I will do so with structural induction on φ. Suppose pRq.

- Let $p \models a\varphi$. Then there is a $p' \in \mathbb{P}$ with $p \xrightarrow{a} p'$ and $p' \models \varphi$. As R is a bisimulation, there must be a $q' \in \mathbb{P}$ with $q \xrightarrow{a} q'$ and $p'Rq'$. So by induction $q' \models \varphi$, and hence $q \models a\varphi$.
  By symmetry one also obtains $q \models a\varphi \Rightarrow p \models a\varphi$.
- $p \models \bigwedge_{i\in I} \varphi_i \Leftrightarrow \forall i \in I (p \models \varphi_i) \stackrel{\text{ind.}}{\Longleftrightarrow} \forall i \in I (q \models \varphi_i) \Leftrightarrow q \models \bigwedge_{i\in I} \varphi_i$.
- $p \models \neg\varphi \Leftrightarrow p \not\models \varphi \stackrel{\text{ind.}}{\Longleftrightarrow} q \not\models \varphi \Leftrightarrow q \models \neg\varphi$.

For "$\Leftarrow$" it suffices to establish that $\sqsubseteq_B$ is a simulation (Proposition 12.1 then implies that $=_B = \sqsubseteq_B = \sqsubseteq_B^{-1}$ is a bisimulation). This goes exactly as in the proof of Proposition 8.2. □

Testing scenario The testing scenario for bisimulation semantics, as presented in Milner [37], is the oldest and most powerful testing scenario, from which most others have been derived by omitting some of its features. It was based on a reactive failure trace machine with replicator, but additionally the observer is equipped with the capacity of *global testing*. Global testing is described in Abramsky [1] as: "the ability to enumerate all (of finitely many) possible 'operating environments' at each stage of the test, so as to guarantee that all nondeterministic branches will be pursued by various copies of the subject process". Milner [37] implemented global testing by assuming that

"(i) It is the *weather* at any moment which determines the choice of transition (in case of ambiguity [...]);

(ii) The weather has only finitely many states—at least as far as choice-resolution is concerned;

(iii) We can control the weather."

Now it can be ensured that all possible moves a process can perform in reaction on a given a-experiment will be investigated by simply performing the experiment in all possible weather conditions. Unfortunately, as remarked in MILNER [38], the second assumption implies that the amount of different moves an investigated process can perform in response to any given experiment is bounded by the number of possible weather conditions (i.e. $\exists n \in \mathbb{N}\ \forall p \in \mathbb{P}\ \forall a \in Act$: $|\{q \in \mathbb{P} \mid p \xrightarrow{a} q\}| < n$). So for general application this condition has to be dropped, thereby losing the possibility of effective implementation of the testing scenario.

An observation in the global testing scenario can be represented as an infinitary Hennessy-Milner formula $\varphi \in \mathcal{L}_B$. This is essentially a simulation formula in which it is possible to indicate that certain branches are not present. A formula $\neg\varphi$ says that by making sufficiently many copies of the investigated process, and exposing them to all possible weather conditions, it can be observed that none of these copies permits the observation φ.

Remark: Let $[a]\varphi$ denote $\neg a \neg\varphi$. Now the negation in $\mathcal{L}_B$ can be eliminated in favour of the modalities $[a]$ and infinitary disjunction $\bigvee_{i\in I}$. A formula $[a]\varphi$ says that in all possible weather conditions, after an a-move it is always possible to make the observation φ.

In order to justify the observations of $\mathcal{L}_B$ in a generative testing scenario no switches or menu-lights are needed; the architecture of the completed trace machine suffices. However, in order to warrant negative observations, one has to assume that actions take only a finite amount of time, and idling can be detected (either by observations that last forever, or by means of the display becoming empty). Adding switches and or menu-lights does not increase the discriminating power of the observers. It would give rise to observations that can be modelled as formulas in languages $\mathcal{L}_{FB}$, $\mathcal{L}_{RTB}$, etc., obtained by combining $\mathcal{L}_F$, $\mathcal{L}_{RT}$, etc. with $\mathcal{L}_B$. These observations can already be expressed in $\mathcal{L}_B$: $p \models \widetilde{X} \Leftrightarrow p \models \bigwedge_{a\in X} \neg a\top$ and $p \models X\varphi \Leftrightarrow p \models (\bigwedge_{a\notin X} \neg a\top) \wedge (\bigwedge_{a\in X} a\top) \wedge \varphi$.

A different implementation of global testing is given in LARSEN & SKOU [35]. They assumed that every transition in a transition system has a certain probability of being taken. Therefore an observer can with an arbitrary high degree of confidence assume that all transitions have been examined, simply by repeating an experiment many times.

As argued among others in BLOOM, ISTRAIL & MEYER [12], global testing in the above sense is a rather unrealistic testing ability. Once you assume that the observer is really as powerful as in the described scenarios, in fact more can be tested then only bisimulation equivalence: in the testing scenario of Milner also the correlation between weather conditions and transitions being taken by the investigated process can be recovered, and in that of Larsen & Skou one can determine the relative probabilities of the various transitions.

Process graph characterization Also bisimulation equivalence can be characterized by means of relations between the nodes of two process graphs.

Definition 12.2 Let $g, h \in \mathbb{G}$. A *bisimulation* between g and h is a binary relation $R \subseteq \text{NODES}(g) \times \text{NODES}(h)$, satisfying:

- $\text{ROOT}(g) R \text{ROOT}(h)$.
- If sRt and $(s, a, s') \in \text{EDGES}(g)$, then there is an edge $(t, a, t') \in \text{EDGES}(h)$ such that $s'Rt'$.
- If sRt and $(t, a, t') \in \text{EDGES}(h)$, then there is an edge $(s, a, s') \in \text{EDGES}(g)$ such that $s'Rt'$.

This definition is illustrated in Figure 8. Solid lines indicates what is assumed, dashed lines what is required. It follows easily that $g \leftrightarrows h$ iff there exists a bisimulation between g and h.

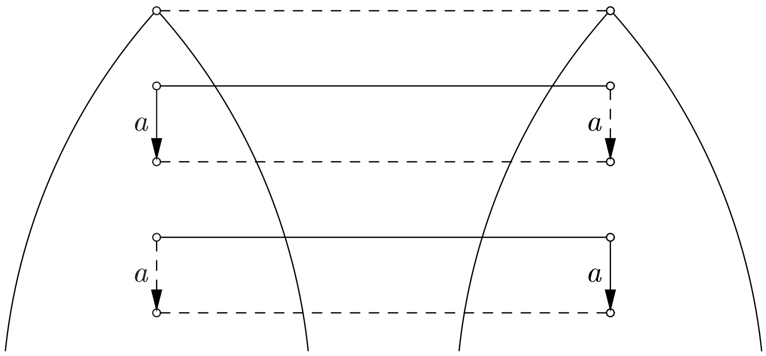
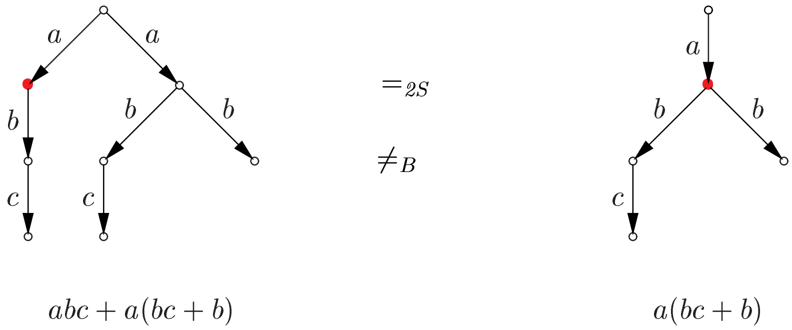


Figure 8: A bisimulation

For process graphs with multiple roots, the first requirement of Definition 12.2 generalizes to

- $\forall s \in \text{ROOTS}(g)\, \exists t \in \text{ROOTS}(h) : sRt$.
- $\forall t \in \text{ROOTS}(h)\, \exists s \in \text{ROOTS}(g) : sRt$.

Classification $2S \prec B$.
Proof: "$2S \preceq B$" follows since $\mathcal{L}_{2S}$ is a sublanguage of $\mathcal{L}_B$.
"$2S \not\succeq B$" follows from Counterexample 14, which shows two graphs that are 2-nested simulation equivalent, but not bisimulation equivalent. Concerning the first claim, as in Counterexample 10 there exists exactly one simulation of *left* by *right*, which relates the red (or shaded) node on the *left* to the red (or shaded) node on the *right*. Unlike in Counterexample 10, this simulation is 2-nested, for the two subgraphs originating from the two red (or shaded) nodes are simulation equivalent, as are the graphs *left* and *right* themselves. Likewise, the simulation mapping *right* on the right-hand side of *left* is also 2-nested. The second claim follows since $a\neg b \neg c \top \in B(\mathit{left}) - B(\mathit{right})$. □



Counterexample 14: 2-nested simulation equivalent, but not bisimulation equivalent

Thus bisimulation equivalence is the finest semantic equivalence treated so far. The following shows however that on $\mathbb{G}$ graph isomorphism is even finer, i.e. isomorphic graphs are always bisimilar. In fact, a graph isomorphism can be seen as a bijective bisimulation. That not all bisimilar graphs are isomorphic will follow from Counterexample 15.

Proposition 12.3 For $g, h \in \mathbb{G}$, $g \cong h$ iff there exists a bisimulation R between g and h, satisfying

- If sRt and uRv then $s = u \Leftrightarrow t = v$. (*)

Proof: Suppose $g \cong h$. Let $f : \text{NODES}(g) \rightarrow \text{NODES}(h)$ be a graph isomorphism. Define $R \subseteq \text{NODES}(g) \times \text{NODES}(h)$ by sRt iff $f(s) = t$. Then it is routine to check that R satisfies all clauses of Definition 12.2 and (*). Now suppose R is a bisimulation between g and h satisfying (*). Define $f : \text{NODES}(g) \rightarrow \text{NODES}(h)$ by $f(s) = t$ iff sRt. Since g is connected it follows from the definition of a bisimulation that for each s such a t can be found. Furthermore direction "$\Rightarrow$" of (*) implies that $f(s)$ is uniquely determined. Hence f is well-defined. Now direction "$\Leftarrow$" of (*) implies that f is injective. From the connectedness of h if follows that f is also surjective, and hence a bijection. Finally, the clauses of Definition 12.2 imply that f is a graph isomorphism. □

Corollary 12.1 If $g \cong h$ then g and h are equivalent according to all semantic equivalences encountered so far. □

Non-well-founded sets Another characterization of bisimulation semantics can be given by means of ACZEL's universe $\mathcal{V}$ of non-well-founded sets [4]. This universe is an extension of the Von Neumann universe of well-founded sets, where the axiom of foundation (every chain $x_0 \ni x_1 \ni \cdots$ terminates) is replaced by an *anti-foundation axiom.*

Definition 12.3 Let $\mathcal{B}$ denote the unique function $\mathcal{M} : \mathbb{P} \rightarrow \mathcal{V}$ satisfying

$$\mathcal{M}(p) = \{\langle a, \mathcal{M}(q)\rangle \mid p \xrightarrow{a} q\}$$

for all $p \in \mathbb{P}$. Two processes p and q are *branching equivalent* (my terminology) if $\mathcal{B}(p) = \mathcal{B}(q)$.

It follows from Aczel's anti-foundation axiom that such a function exists. In fact the axiom amounts to saying that systems of equations like the one above have unique solutions. In [4] there is also a section on communicating systems. There two processes are identified iff they are branching equivalent.

A similar idea underlies the semantics of DE BAKKER & ZUCKER [9], but there the domain of processes is a complete metric space and the definition of $\mathcal{B}$ above only works for finitely branching processes, and only if $=$ is interpreted as *isometry*, rather then equality, in order to stay in well-founded set theory. For finitely branching processes the semantics of De Bakker and Zucker coincides with the one of Aczel and also with bisimulation semantics. This is observed in VAN GLABBEEK & RUTTEN [22], where also a proof can be found of the next proposition, saying that bisimulation equivalence coincides with branching equivalence.

Proposition 12.4 Let $p, q \in \mathbb{P}$. Then $p \leftrightarroweq q \;\Leftrightarrow\; \mathcal{B}(p) = \mathcal{B}(q)$.

Proof: "$\Leftarrow$": Let B be the relation defined by pBq iff $\mathcal{B}(p) = \mathcal{B}(q)$; then it suffices to prove that B is a bisimulation. Suppose pBq and $p \xrightarrow{a} p'$. Then $\langle a, \mathcal{B}(p')\rangle \in \mathcal{B}(p) = \mathcal{B}(q)$. So by the definition of $\mathcal{B}(q)$ there must be a process q' with $\mathcal{B}(p') = \mathcal{B}(q')$ and $q \xrightarrow{a} q'$. Hence $p'Bq'$, which had to be proved. The second requirement for B being a bisimulation follows by symmetry.

"$\Rightarrow$": Let $\mathcal{B}^*$ denote the unique solution of $\mathcal{M}^*(p) = \{\langle a, \mathcal{M}^*(r')\rangle \mid \exists r : r \leftrightarroweq p \wedge r \xrightarrow{a} r'\}$. As for $\mathcal{B}$ it follows from the anti-foundation axiom that such a unique solution exists. From the symmetry and transitivity of $\leftrightarroweq$ it follows that

$$p \leftrightarroweq q \;\;\Rightarrow\;\; \mathcal{B}^*(p) = \mathcal{B}^*(q). \tag{1}$$

Hence it remains to be proven that $\mathcal{B}^* = \mathcal{B}$. This can be done by showing that $\mathcal{B}^*$ satisfies the equations $\mathcal{M}(p) = \{\langle a, \mathcal{M}(q)\rangle \mid p \xrightarrow{a} q\}$, which have $\mathcal{B}$ as unique solution. So it has to be established that $\mathcal{B}^*(p) = \{\langle a, \mathcal{B}^*(q)\rangle \mid p \xrightarrow{a} q\}$. The direction "$\supseteq$" follows directly from the reflexivity of $\leftrightarroweq$. For "$\subseteq$", suppose $\langle a, X\rangle \in \mathcal{B}^*(p)$. Then $\exists r : r \leftrightarroweq p$, $r \xrightarrow{a} r'$ and $X = \mathcal{B}^*(r')$. Since $\leftrightarroweq$ is a bisimulation, $\exists p' : p \xrightarrow{a} p'$ and $r' \leftrightarroweq p'$. From (1) it follows that $X = \mathcal{B}^*(r') = \mathcal{B}^*(p')$. Therefore $\langle a, X\rangle \in \{\langle a, \mathcal{B}^*(q)\rangle \mid p \xrightarrow{a} q\}$, which had to be established. $\square$

Infinite processes The following predecessor of bisimulation equivalence was proposed in Hennessy & Milner [27, 28].

Definition 12.4 Let $p, q \in \mathbb{P}$. Then:

- $p \sim_0 q$ is always true.
- $p \sim_{n+1} q$ if for all $a \in Act$:
  - $p \xrightarrow{a} p'$ implies $\exists q' : q \xrightarrow{a} q'$ and $p' \sim_n q'$;
  - $q \xrightarrow{a} q'$ implies $\exists p' : p \xrightarrow{a} p'$ and $p' \sim_n q'$.
- p and q are *observationally equivalent*, notation $p \sim q$, if $p \sim_n q$ for every $n \in \mathbb{N}$.

Hennessy and Milner provided the following modal characterization of observational equivalence on image finite processes.

Definition 12.5 The set $\mathcal{L}_{\mathrm{HM}}$ of *Hennessy-Milner formulas* over Act is defined recursively by:

- $\top \in \mathcal{L}_{\mathrm{HM}}$.
- If $\varphi, \psi \in \mathcal{L}_{\mathrm{HM}}$ then $\varphi \wedge \psi \in \mathcal{L}_{\mathrm{HM}}$.
- If $\varphi \in \mathcal{L}_{\mathrm{HM}}$ and $a \in Act$ then $a\varphi \in \mathcal{L}_{\mathrm{HM}}$.
- If $\varphi \in \mathcal{L}_{\mathrm{HM}}$ then $\neg\varphi \in \mathcal{L}_{\mathrm{HM}}$.

The *satisfaction relation* $\models \subseteq \mathbb{P} \times \mathcal{L}_{\mathrm{HM}}$ is defined recursively by:

- $p \models \top$ for all $p \in \mathbb{P}$.
- $p \models \varphi \wedge \psi$ if $p \models \varphi$ and $p \models \psi$.
- $p \models a\varphi$ if for some $q \in \mathbb{P}$: $p \xrightarrow{a} q$ and $q \models \varphi$.
- $p \models \neg\varphi$ if $p \not\models \varphi$.

The modal logic above is now known as the *Hennessy-Milner logic* (*HML*). Let $HM(p)$ denote the set of all Hennessy-Milner formulas that are satisfied by the process p: $HM(p) = \{\varphi \in \mathcal{L}_{\mathrm{HM}} \mid p \models \varphi\}$. Two processes p and q are *HML-equivalent*, notation $p =_B^- q$, if $HM(p) = HM(q)$.

Theorem 2.2 in Hennessy & Milner [27, 28] says that $\sim$ and $=_B^-$ coincide for image finite processes. This result will be strengthened by Proposition 12.6. Below I provide a modal characterization of $\sim$ that is valid for arbitrary processes.

Definition 12.6 Let $\mathcal{L}_B^\omega = \bigcup_{n=0}^\infty \mathcal{L}_B^n$, where $\mathcal{L}_B^n$ is given by:

- If I is a set and $\varphi_i \in \mathcal{L}_B^n$ for $i \in I$ then $\bigwedge_{i\in I} \varphi_i \in \mathcal{L}_B^n$.
- If $\varphi \in \mathcal{L}_B^n$ and $a \in Act$ then $a\varphi \in \mathcal{L}_B^{n+1}$.

- If $\varphi \in \mathcal{L}_B^n$ then $\neg\varphi \in \mathcal{L}_B^n$.

Let $B^\omega(p) = \{\varphi \in \mathcal{L}_B^\omega \mid p \models \varphi\}$ and write $p =_B^\omega q$ if $B^\omega(p) = B^\omega(q)$.

Proposition 12.5 $p \sim_n q \Leftrightarrow \forall \varphi \in \mathcal{L}_B^n (p \models \varphi \Leftrightarrow q \models \varphi)$ for all $n \in \mathbb{N}$. Hence $p \sim q \Leftrightarrow p =_B^\omega q$.
Proof: *Induction Base:* Formulas in $\mathcal{L}_B^0$ do not contain the construct $a\varphi$. Hence for such formulas ψ the statement $p \models \psi$ is independent of p. Thus $\forall p, q \in \mathbb{P} : \forall \varphi \in \mathcal{L}_B^0 (p \models \varphi \Leftrightarrow q \models \varphi)$.
Induction Step: Suppose $p \sim_{n+1} q$. I now use structural induction on φ.

- Let $p \models a\varphi$ with $a\varphi \in \mathcal{L}_B^{n+1}$. Then there is a $p' \in \mathbb{P}$ with $p \xrightarrow{a} p'$ and $p' \models \varphi \in \mathcal{L}_B^n$. As $p \sim_{n+1} q$, there must be a $q' \in \mathbb{P}$ with $q \xrightarrow{a} q'$ and $p' \sim_n q'$. So by induction $q' \models \varphi$, and hence $q \models a\varphi$.
  By symmetry one also obtains $q \models a\varphi \Rightarrow p \models a\varphi$.
- $p \models \bigwedge_{i \in I} \varphi_i \Leftrightarrow \forall i \in I (p \models \varphi_i) \overset{\text{ind.}}{\Longleftrightarrow} \forall i \in I (q \models \varphi_i) \Leftrightarrow q \models \bigwedge_{i \in I} \varphi_i$.
- $p \models \neg\varphi \Leftrightarrow p \not\models \varphi \overset{\text{ind.}}{\Longleftrightarrow} q \not\models \varphi \Leftrightarrow q \models \neg\varphi$.

Now suppose $\forall \varphi \in \mathcal{L}_B^{n+1} (p \models \varphi \Leftrightarrow q \models \varphi)$ and $p \xrightarrow{a} p'$. Considering the symmetry in the definitions involved, all I have to show is that $\exists q' \in \mathbb{P}$ with $q \xrightarrow{a} q'$ and $p' \sim_n q'$. Let Q' be

$$\{q' \in \mathbb{P} \mid q \xrightarrow{a} q' \wedge p' \not\sim_n q'\}.$$

By Definition 1.1 Q' is a set. For every $q' \in Q'$ there must, by induction, be a formula $\varphi_{q'} \in \mathcal{L}_B^n$ with $p' \models \varphi_{q'}$ but $q' \not\models \varphi_{q'}$ (use negation if necessary). Now $p \models a \bigwedge_{q' \in Q'} \varphi_{q'} \in \mathcal{L}_B^{n+1}$ and therefore $q \models a \bigwedge_{q' \in Q'} \varphi_{q'}$. So there must be a $q' \in \mathbb{P}$ with $q \xrightarrow{a} q'$ and $q' \notin Q'$, which had to be shown. $\square$

Comparing their modal characterizations ($=_B$ of $\leftrightarrows$ and $=_B^\omega$ of $\sim$) one finds

$$p \leftrightarrows q \quad \Rightarrow \quad p \sim q \quad \Rightarrow \quad p =_B^- q.$$

Theorem 2.1 in Hennessy & Milner [27, 28] says, essentially, that for image finite processes the relation $\sim$ satisfies the defining properties of a bisimulation (cf. Definition 12). Inspired by this insight, Park [41] proposed the concise formulation of bisimulation equivalence employed in Definition 12. It follows immediately that if $p, q \in \mathbb{P}$ are image finite, then $p \leftrightarrows q \Leftrightarrow p \sim q$. The following strengthening of this result is due to Hollenberg [32].

Proposition 12.6 Let $p, q \in \mathbb{P}$ and p is image finite. Then $p \leftrightarrows q \Leftrightarrow p =_B^- q$.
Proof: Write pBq iff $p =_B^- q$ and p is image finite. It suffices to establish that B is a bisimulation.

- Suppose pBq and $q \xrightarrow{a} q'$. I have to show that $\exists r \in \mathbb{P}$ with $p \xrightarrow{a} r$ and $HM(r) = HM(q')$. Let R be
  $$\{r \in \mathbb{P} \mid p \xrightarrow{a} r \wedge HM(r) \neq HM(q')\}.$$
  As p is image finite, R is finite. For every $r \in R$ take a formula $\varphi_r \in HM(q') - HM(r)$ (note that if $\psi \in HM(r) - HM(q')$ then $\neg\psi \in HM(q') - HM(r)$). Now
  $$a \bigwedge_{r \in R} \varphi_r \in HM(q) = HM(p),$$
  so there must be a $r \in \mathbb{P}$ with $p \xrightarrow{a} r$ and $r \models \bigwedge_{r \in R} \varphi_r$. The latter implies $r \notin R$, i.e. $HM(r) = HM(q')$, which had to be shown.

– Suppose pBq and $p \xrightarrow{a} p'$. I have to show that $\exists q' \in \mathbb{P}$ with $q \xrightarrow{a} q'$ and $HM(p') = HM(q')$. Let S be
$$\{s \in \mathbb{P} \mid p \xrightarrow{a} s \wedge HM(s) \neq HM(p')\}.$$
As p is image finite, S is finite. For every $s \in S$ take a formula $\varphi_s \in HM(p') - HM(s)$. Now
$$a \bigwedge_{s \in S} \varphi_s \in HM(p) = HM(q),$$
so there must be a $q' \in \mathbb{P}$ with $q \xrightarrow{a} q'$ and $q' \models \bigwedge_{s \in S} \varphi_s$. By the previous item in this proof, $\exists r \in \mathbb{P}$ with $p \xrightarrow{a} r$ and $HM(r) = HM(q')$, hence $r \models \bigwedge_{s \in S} \varphi_s$. The latter implies $r \notin S$, so $HM(r) = HM(p')$. Thus $HM(p') = HM(q')$, which had to be shown. □

By Counterexample 12, a result like the one above does not hold for (ready) simulation semantics.

For the sake of completeness, two more variants of bisimulation equivalence can be considered. Let FB^* be characterized by the Hennessy-Milner logic augmented with formulas $\widetilde{X}$, and RB^* by the Hennessy-Milner logic augmented with formulas X, for $X \subseteq Act$.

Then $B^- \prec FB^* \prec RB^* \prec B^\omega \prec B$, and for image finite processes all five equivalences coincide. The strictness of these inclusions is given by Counterexamples 4, 11, 9 and 1:

Proposition 12.7 $CT \not\preceq B^-$, and hence $FB^* \not\preceq B^-$.
Proof: Counterexample 4 shows two processes with $CT(\mathit{left}) \neq CT(\mathit{right})$. It remains to be shown that $HM(\mathit{left}) = HM(\mathit{right})$, i.e. that for all $\varphi \in \mathcal{L}_{\mathrm{HM}}$: $\mathit{left} \models \varphi \Leftrightarrow \mathit{right} \models \varphi$. Using Definition 12.5 it is sufficient to restrict attention to formulas φ which are of the form $a(\bigwedge_{i\in I} b_i\top \wedge \bigwedge_{j\in J} \neg b_j\top)$ with I and J finite sets of indices. It is not difficult to see that each such formula that is satisfied on one side is also satisfied on the other side. □

Proposition 12.8 $R \not\preceq FB^*$, and hence $RB^* \not\preceq FB^*$.
Proof: Counterexample 11, shows two processes with $R(\mathit{with}) \neq R(\mathit{without})$. It remains to be shown that $FB^*(\mathit{with}) = FB^*(\mathit{without})$. The argument is the same as in the previous proof, but this time focusing on formulas of the form $a(\widetilde{X} \wedge \bigwedge_{i\in I} i\top \wedge \bigwedge_{j\in J} \neg j\top)$ with I and J finite sets of numbers and X a possibly infinite set of numbers (= actions). □

Proposition 12.9 $S^\omega \not\preceq RB^*$, and hence $RB^\omega \not\preceq RB^*$.
Proof: Counterexample 9 shows two processes with $S^\omega(\mathit{with}) \neq S^\omega(\mathit{without})$. It remains to be shown that $RB^*(\mathit{with}) = RB^*(\mathit{without})$. The argument is the same as in the previous proofs—this time using formulas $a(\{b\} \wedge \bigwedge_{i\in I} bi\top \wedge \bigwedge_{j\in J} \neg bj\top)$ with I and J finite sets of numbers. □

Proposition 12.10 $T^\infty \not\preceq B^\omega$, and hence $B \not\preceq B^\omega$. In addition, $PF \not\preceq B^\omega$.
Proof: Counterexample 1 shows two processes with $T^\infty(\mathit{left}) \neq T^\infty(\mathit{right})$. As remarked at the end of Section 7, also $PF(\mathit{left}) \neq PF(\mathit{right})$. It remains to be shown that $\mathit{left} =^\omega_B \mathit{right}$, i.e. that for all $n \in \mathbb{N}$: $\mathit{left} \sim_n \mathit{right}$. In order to establish $p \sim_n q$ for two trees p and q, the parts of p and q that are further than n edges away from the root play no rôle, and can just as well be omitted. As the cut versions of *left* and *right* are isomorphic, by Corollary 12.1 surely $\mathit{left} \sim_n \mathit{right}$. □

In addition one has $2S^- \prec B^-$, $F2S^* \prec FB^*$, $R2S^* \prec RB^*$, $2S^\omega \prec B^\omega$ and $2S \prec B$.

13 Tree semantics

Definition 13 Let $g \in \mathbb{G}$. The *unfolding* of g is the graph $U(g) \in \mathbb{G}$ defined by

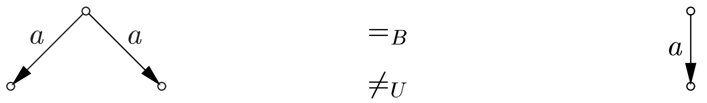
- $\textsc{nodes}(U(g)) = \textsc{paths}(g)$,
- $\textsc{root}(U(g)) = \textsc{root}(g)$, i.e. the empty path, starting and ending at the root of g,
- $(\pi, a, \pi') \in \textsc{edges}(U(g))$ iff π' extends π by one edge, which is labelled a.

Two processes p and q are *tree equivalent*, notation $p =_U q$, if their unfoldings are isomorphic, i.e. if $U(G(p)) \cong U(G(p))$. In *tree semantics* (U) two processes are identified iff they are tree equivalent.

It is easy to see that the unfolding of any process graph is a tree, and the unfolding of a tree is isomorphic to itself. It follows that up to isomorphism every tree equivalence class of process graphs contains exactly one tree, which can be obtained from an arbitrary member of the class by means of unfolding.

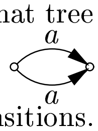
Proposition 13.1 Let $g \in \mathbb{G}$. Then $U(g) \,\underline{\leftrightarrow}\, g$. Hence $g =_U h \Rightarrow g \,\underline{\leftrightarrow}\, h$.
Proof: As is easily verified, $\{(\pi, end(\pi)) \mid \pi \in \textsc{paths}(g)\}$ is a bisimulation between $U(g)$ and g. $\Box$

Tree semantics is employed in Winskel [50]. No plausible testing scenario or modal characterization is known for it. Proposition 13.1 shows that $B \preceq U$. That $B \not\succeq U$ follows from Counterexample 15.



$a + a$ $\qquad\qquad$ a

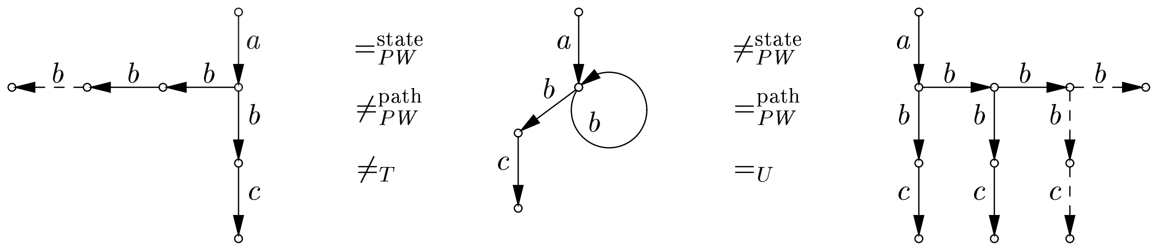
Counterexample 15: Bisimulation equivalent, but not tree equivalent

Although above tree equivalence is defined entirely in terms of action relations, such a definition is in fact misleading, as action relations abstract from an aspect of system behaviour that tree semantics tries to capture. The problem can best be explained by considering the process  that can proceed from its initial to its final state by performing one of two different a-transitions. In tree semantics, such a process should be considered equivalent to the leftmost process of Counterexample 15, and hence different from the rightmost one. However, action relations only tell whether a process p can evolve into q by performing an a-action; they do not tell in how many ways this can happen. So in labelled transition systems as defined in this paper the mentioned process is represented as $\circ\xrightarrow{a}\circ$ and hence considered tree equivalent to the rightmost process of Counterexample 15. The mishap that ensues this way will be illustrated in Section 17.

Tree semantics on labelled transitions systems as in Section 1.1 is a sensible notion only if one knows that each transition in the system can be taken in only one way. In general, more satisfactory domains for defining tree equivalence are labelled transition systems in which the transitions (p, a, q) are equipped with a multiplicity, telling in how many different ways this transition can be taken, or process graphs $g = (\textsc{nodes}(g), \textsc{root}(g), \textsc{edges}(g), begin, end, label)$ in which $\textsc{nodes}(g)$ and $\textsc{edges}(g)$ are sets, $\textsc{root}(g) \in \textsc{nodes}(g)$, $begin$, end : $\textsc{edges}(g) \to \textsc{nodes}(g)$ and *label* : EDGES$(g) \rightarrow Act$. The functions *begin*, *end* and *label* associate with every edge a triple $(s, a, t) \in \text{NODES}(g) \times Act \times \text{NODES}(g)$, but contrary to the situation in Definition 1.3 the identity of an edge is not completely determined by such a triple. On such process graphs, the notions PATHS, unfolding and tree equivalence are defined exactly as for the process graphs of Definition 1.3.

14 Possible worlds semantics

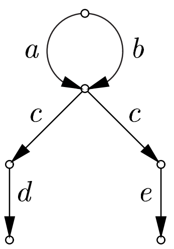
In VEGLIONI & DE NICOLA [49], a nondeterministic process is viewed as a set of deterministic ones: its *possible worlds*. Two processes are said to be *possible worlds equivalent* iff they have the same possible worlds. Two different approaches by which a nondeterministic process can be resolved into a set of deterministic ones need to be distinguished; I call them the *state-based* and the *path-based* approach. In the state-based approach a deterministic process h is obtained out of a nondeterministic process $g \in \mathbb{G}$ by choosing, for every state s of g and every action $a \in I(s)$ a single edge $s \xrightarrow{a} s'$. Now h is the reachable part of the subgraph of g consisting of the chosen edges. In the path-based approach on the other hand, one chooses for every path $\pi \in \text{PATHS}(g)$ and every action $a \in I(end(\pi))$ a single edge $end(\pi) \xrightarrow{a} s'$ to continue with. The chosen edges may now be different for different paths ending in the same state. The difference between the two approaches is illustrated in Counterexample 16. In the state-based approach, the process in the middle has



Counterexample 16: State-based versus path-based possible worlds equivalence

two possible worlds, depending on which of the two b-edges is chosen. These worlds are essentially abc and ab^∞. In the path-based approach, the process in the middle has countably many possible worlds, namely $ab^n c$ for $n \geq 1$ and ab^∞.

In [49], Veglioni & De Nicola take the state-based approach: "once we have resolved the underspecification present in a state s by saying, for example, $s \xrightarrow{a} s$, then, we cannot choose $s \xrightarrow{a} 0$ in the same possible world." However, they provide a denotational characterization of possible worlds semantics on finite processes, namely by inductively allocating sets of deterministic trees to BCCSP expressions (cf. Section 17), which can be regarded as path-based. In addition, they give an operational characterization of possible world semantics, essentially following the state-based approach outlined above. They claim that both characterizations agree. This, however, is not the case, as Counterexample 17 reveals a difference between the two approaches even on finite processes. In the path-based approach the process displayed has a possible world $acd + bce$ (i.e. a process with branches acd and bce), which it has not in the state-based approach. As it turns out, the complete axiomatization they provide w.r.t. BCCSP is correct for the path-based, denotational characterization, but is unsound for the state-based, operational characterization. To be precise: their operational semantics fails to be compositional w.r.t. BCCSP.



Counterexample 17: State-based versus path-based possible worlds equivalence for finite processes

Counterexample 16 shows that a suitable formulation[8] of the state-based approach to possible worlds semantics is incomparable with any of the semantics encountered so far. The processes *left* and *middle* are state-based possible worlds equivalent, yet $abbc \in T(middle) - T(left)$. Furthermore, the processes *right* and *middle* are tree equivalent, yet in the state-based approach one has $abbc \in PW(right) - PW(middle)$.

Below I propose a formalization of the path-based approach to possible worlds semantics that, on finite processes, agrees with the denotational characterization of [49].

Definition 14 A process p is a *possible world* of a process q if p is deterministic and $p \sqsubseteq_{RS} q$. Let $PW(q)$ denote the class of possible worlds of q. Two processes q and r are *possible worlds equivalent*, notation $q =_{PW} r$, if $PW(q) = PW(r)$. In *possible worlds semantics* (PW) two processes are identified iff they are possible worlds equivalent. Write $q \sqsubseteq_{PW} r$ iff $PW(q) \subseteq PW(r)$.

It can be argued that the philosophy underlying possible worlds semantics is incompatible with the view on labelled transition systems taken in this paper. The informal explanation of the action relations in Section 1.1 implies for instance that the right-hand process graph of Counterexample 8 has a state in which a has happened already and both bc and bd are possible continuations. In the possible worlds philosophy on the other hand, this process graph is just a compact representation of the set of deterministic processes $\{abc, abd\}$. None of the two processes in this set has such a state.

This could be a reason not to treat possible worlds semantics on the same footing as the other semantics of this paper. However, one can give up on thinking of non-deterministic processes as sets of deterministic ones, and justify possible worlds semantics—at least the path-based version of Definition 14—by an appropriate testing scenario. This makes it fit in the present paper.

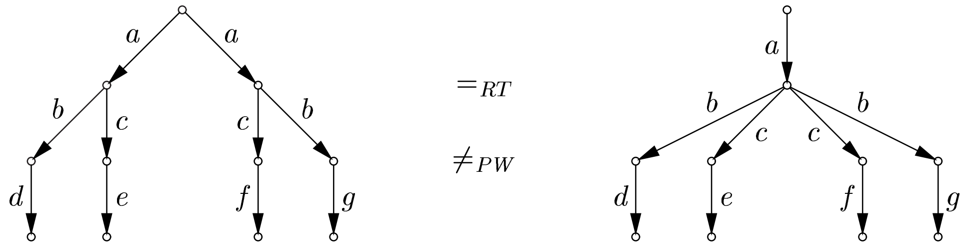
Testing scenario A testing scenario for possible worlds semantics can be obtained by making one change in the reactive testing scenario of failure simulation semantics. Namely in each state only as many copies of the process can be made as there are actions in Act, and, for $a \in Act$, the first test on copy p_a of p is pressing the a-button. If it goes down, one goes on testing that copy, but is has already changed its state; if it does not go down, the test on p_a ends.

Modal characterization On well-founded processes, a modal characterization of possible worlds semantics can be obtained out of the modal characterization of ready simulation semantics by changing the modality $\bigwedge_{i\in I} \varphi_i$ into $\bigwedge_{a\in X} a\varphi_a$ with $X \subseteq Act$. Possible worlds of a well-founded

[8]Let two processes be *possible worlds equivalent* iff each possible world of the one is $\sim$-equivalent to a possible world of the other, where $\sim$ is any of the equivalences treated in this paper. Theorem 6 will imply that the choice of $\sim$ is immaterial.

process p can be simply encoded as modal formulas in the resulting language. Probably, this modal characterization applies to image finite processes as well. For processes that are neither well-founded nor image finite this characterization is not exact, as it fails to distinguish the two processes of Counterexample 1.

Classification $RT \prec PW \prec RS$. PW is independent of S, CS and PF.
Proof: "$PW \preceq RS$"[9] follows by the transitivity of $\sqsubseteq_{RS}$.
"$RT \preceq PW$" holds as σ is a ready trace of $p \in \mathbb{P}$ iff it is a ready trace of a possible world of p.
"$S \not\preceq PW$" (and hence "$RS \not\preceq PW$") follows from Counterexample 8. There $S(\mathit{left}) \neq S(\mathit{right})$, but $PW(\mathit{left}) = PW(\mathit{right}) = \{abc, abd\}$.
"$PF \not\preceq PW$" follows since $PF \not\preceq RS$.
"$CS \not\succeq PW$" follows since $CS \not\succeq RT$, and "$PF \not\succeq PW$" since $PF \not\succeq RT$.
Finally, "$RT \not\succeq PW$" follows from Counterexample 18, taken from [49]. There the first process denotes two possible worlds, whereas the second one denotes four. □



$a(bd + ce) + a(cf + bg)$ $\qquad$ $a(bd + ce + cf + bg)$

Counterexample 18: Ready trace equivalent, but not possible worlds equivalent

Infinite processes The version of possible worlds semantics defined above is the infinitary one. Note that $RT^\infty \prec PW$. Exactly as above one even establishes $p \sqsubseteq_{RS} q \Rightarrow p \sqsubseteq_{PW} q \Rightarrow p \sqsubseteq_{RT}^{\infty} q$, i.e. $RT^\infty \preceq^* PW \preceq^* RS$. Finitary versions could be defined by means of the modal characterization given above. I will not pursue this here.

15 Summary

In Sections 2–14 fifteen semantics were defined that are different for finitely branching processes. These are abbreviated by T, CT, F^1, F, R, FT, RT, PF, S, CS, RS, PW, $2S$, B and U. For each of these semantics $\mathcal{O}$, except U, a modal language $\mathcal{L}_\mathcal{O}$ (a set of modal formulas φ) has been defined:

$\mathcal{L}_T$	$\varphi ::= \top \mid a\varphi' \ (\varphi' \in \mathcal{L}_T)$	the *(partial) trace formulas*
$\mathcal{L}_{CT}$	$\varphi ::= \top \mid a\varphi' \ (\varphi' \in \mathcal{L}_{CT}) \mid 0$	the *completed trace formulas*
$\mathcal{L}_{F^1}$	$\varphi ::= \top \mid a\varphi' \ (\varphi' \in \mathcal{L}_{F^1}) \mid \widetilde{a} \ (a \in Act)$	the *singleton-failure formulas*
$\mathcal{L}_F$	$\varphi ::= \top \mid a\varphi' \ (\varphi' \in \mathcal{L}_F) \mid \widetilde{X} \ (X \subseteq Act)$	the *failure formulas*
$\mathcal{L}_R$	$\varphi ::= \top \mid a\varphi' \ (\varphi' \in \mathcal{L}_R) \mid X \ (X \subseteq Act)$	the *readiness formulas*
$\mathcal{L}_{FT}$	$\varphi ::= \top \mid a\varphi' \ (\varphi' \in \mathcal{L}_{FT}) \mid \widetilde{X}\varphi' \ (X \subseteq Act,\ \varphi' \in \mathcal{L}_{FT})$	the *failure trace formulas*

[9] The counterexample against "$PW \preceq RS$" given in [49] is incorrect. The two processes displayed there are not ready simulation equivalent.

$\mathcal{L}_{RT}$	$\varphi ::= \top \mid a\varphi' \ (\varphi' \in \mathcal{L}_{RT}) \mid X\varphi' \ (X \subseteq Act,\ \varphi' \in \mathcal{L}_{RT})$	the *ready trace formulas*
$\mathcal{L}_{PF}$	$\varphi ::= a\varphi' \ (\varphi' \in \mathcal{L}_{PF}) \mid \bigwedge_{i\in I}\varphi_i \wedge \bigwedge_{j\in J}\neg\varphi'_j \ (\varphi_i,\ \varphi'_j \in \mathcal{L}_{T})$	the *possible-futures formulas*
$\mathcal{L}_{S}$	$\varphi ::= a\varphi' \ (\varphi' \in \mathcal{L}_{S}) \mid \bigwedge_{i\in I}\varphi_i \ (\varphi_i \in \mathcal{L}_{S})$	the *simulation formulas*
$\mathcal{L}_{CS}$	$\varphi ::= a\varphi' \ (\varphi' \in \mathcal{L}_{CS}) \mid \bigwedge_{i\in I}\varphi_i \ (\varphi_i \in \mathcal{L}_{CS}) \mid 0$	the *completed simulation formulas*
$\mathcal{L}_{RS}$	$\varphi ::= a\varphi' \ (\varphi' \in \mathcal{L}_{RS}) \mid \bigwedge_{i\in I}\varphi_i \ (\varphi_i \in \mathcal{L}_{RS}) \mid X \ (X \subseteq Act)$	the *ready simulation formulas*
$\mathcal{L}_{PW}$	$\bigwedge_{a\in X} a\varphi_a \ (\varphi_a \in \mathcal{L}_{PW},\ X \subseteq Act) \mid X \ (X \subseteq Act)$	the *possible worlds formulas*
$\mathcal{L}_{2S}$	$\varphi ::= a\varphi' \ (\varphi' \in \mathcal{L}_{2S}) \mid \bigwedge_{i\in I}\varphi_i \ (\varphi_i \in \mathcal{L}_{2S}) \mid \neg\varphi' \ (\varphi' \in \mathcal{L}_{S})$	the *2-nested simulation formulas*
$\mathcal{L}_{B}$	$\varphi ::= a\varphi' \ (\varphi' \in \mathcal{L}_{B}) \mid \bigwedge_{i\in I}\varphi_i \ (\varphi_i \in \mathcal{L}_{B}) \mid \neg\varphi' \ (\varphi' \in \mathcal{L}_{B})$	the *bisimulation formulas.*

All these languages can be regarded as sublanguages of $\mathcal{L}_B$, the *infinitary Hennessy-Milner logic*, namely by considering the constructs not in $\mathcal{L}_B$ as abbreviations:

$$\begin{array}{llll} \top := \bigwedge_{i\in\emptyset}\varphi_i & \widetilde{X} := \bigwedge_{a\in X}\neg a\top & \widetilde{X}\varphi' := \widetilde{X}\wedge\varphi' & 0 := \widetilde{Act} \\ \varphi_1\wedge\varphi_2 := \bigwedge_{i\in\{1,2\}}\varphi_i & X := \bigwedge_{a\in X} a\top \wedge \bigwedge_{a\notin X}\neg a\top & X\varphi' := X\wedge\varphi' & \widetilde{a} := \neg a\top \end{array}$$

On any labelled transition system $\mathbb{P}$, the satisfaction relation $\models\ \subseteq \mathbb{P}\times\mathcal{L}_B$ is given by:

$p \models a\varphi$ if for some $q\in\mathbb{P}$: $p \xrightarrow{a} q \wedge q \models \varphi$; $\quad p \models \bigwedge_{i\in I}\varphi_i$ if $\forall i\in I: p\models\varphi_i$; $\quad p\models\neg\varphi$ if $p\not\models\varphi$.

For each semantics $\mathcal{O}\in\{T, CT, F^1, F, R, FT, RT, PF, S, CS, RS, PW, 2S, B\}$ this definition specializes to the sublanguage $\mathcal{L}_{\mathcal{O}}$. Now a *modal characterization* of $\mathcal{O}$-equivalence[10] is given by:

$$p =_{\mathcal{O}} q \ \Leftrightarrow\ \forall\varphi\in\mathcal{L}_{\mathcal{O}}(p\models\varphi \Leftrightarrow q\models\varphi).$$

In the cases $\mathcal{O}\in\{T, CT, F^1, F, R, FT, RT\}$ $\mathcal{O}$-equivalence was defined by $p =_{\mathcal{O}} q \Leftrightarrow \mathcal{O}(p) = \mathcal{O}(q)$. Writing $\mathcal{O}_{\text{modal}}(p)$ for $\{\varphi\in\mathcal{L}_{\mathcal{O}} \mid p\models\varphi\}$, it can be observed that the formulas in $\mathcal{O}_{\text{modal}}(p)$ are mild syntactic variations of the elements in $\mathcal{O}(p)$. Thus, the modal characterization is a rather trivial restatement of the original definition of the equivalence. The modal characterization of PF is fairly easy to check. This is left to the reader. In the cases $\mathcal{O}\in\{S, CS, RS, 2S, B\}$ the modal characterization of $=_{\mathcal{O}}$ has been proven equivalent to a relational characterization in Propositions 8.2, 9.2, 9.3, 11.1 and 12.2. It is a matter of taste which one is taken to be the official definition.

The same applies to the modal characterizations of the $\mathcal{O}$-preorders[10], given by

$$p \sqsubseteq_{\mathcal{O}} q \ \Leftrightarrow\ \forall\varphi\in\mathcal{L}_{\mathcal{O}}(p\models\varphi \Rightarrow q\models\varphi).$$

For each of the semantics $T, CT, F^1, F, R, FT, RT, S, CS, RS, PW, B$ a testing scenario has been proposed in which the modal formulas satisfied by a process p are interpreted as the possible observations that can be made on a suitable machine interacting with p. In particular, the formula $a\varphi$ represents the observation of "a" appearing in the display of a generative machine (or the a-button going down on a reactive machine) followed by the observation φ. The formula $\widetilde{X}$ represents the display of the generative machine becoming empty, while X is the set of actions that are allowed to happen by the environment/observer, i.e. the ones whose switches are set "free". In particular, 0 represents the display becoming empty while all actions are free (in the absence of switches). On a reactive machine, $\widetilde{a}$ represents the a-button refusing to go down, and $\widetilde{X}$ means that none of the a-buttons for $a\in X$ go down when they all receive pressure. X represents the menu lights for the actions in X being lit while the machine is idling. $\top$ represents the act of the observer terminating his observations, and $\bigwedge_{i\in I}\varphi_i$ represents the observations that can be made on $|I|$

[10]In case $\mathcal{O} = PW$ the modal characterization is known to be valid for well-founded processes only.

copies of the investigated process in its current state, obtained by means of a replication facility. Finally $\neg\varphi$ represents the observation that φ cannot be observed—an observation which occurs when copies of the investigated process are exposed to all possible weather conditions, and in none of them the observation φ is made. A testing scenario for a particular semantics is obtained by allowing machines that are equipped with (only) those features corresponding with its modal characterization.

I write $\mathcal{O} \preceq \mathcal{N}$ if semantics $\mathcal{O}$ makes at least as much identifications as semantics $\mathcal{N}$, i.e. if $=_\mathcal{O} \supseteq =_\mathcal{N}$. Clearly, if $\mathcal{L}_\mathcal{O}$ is a sublanguage of $\mathcal{L}_\mathcal{N}$ it must be that $\mathcal{O} \preceq \mathcal{N}$. This immediately yields[11] the following theorem, whose proof has also appeared in the various subsections entitled "classification".

Theorem 1 $T \preceq CT \preceq F \preceq R \preceq RT,\ T \preceq F^1 \preceq F \preceq FT \preceq RT \preceq PW \preceq RS \preceq 2S \preceq B \preceq U,$ $R \preceq PF \preceq 2S,\ T \preceq S \preceq CS \preceq RS$ and $CT \preceq CS$.

Theorem 1 is illustrated in Figure 1. There, however, singleton-failures semantics and completed simulation semantics are missing, since they did not occur in the literature, and appear to be of minor interest. The theorem applies to any labelled transition system $(\mathbb{P}, \rightarrow)$. Whether the inclusions are strict depends on the choice of $(\mathbb{P}, \rightarrow)$. In the subsections "classification" a number of counterexamples have been presented, showing that on $\mathbb{G}$ all semantic notions mentioned in Theorem 1 are different and $\mathcal{O} \preceq \mathcal{N}$ holds only if this follows from that theorem. Moreover, all relevant examples use finite processes only.

Let $\mathbb{H}$ be the set of finite connected process graphs. Here *finite* is used in the sense of Definition 1.2; a process graph $g \in \mathbb{G}$ is finite iff $\text{PATHS}(g)$ is finite, which is the case iff g is acyclic and has only finitely many nodes and edges. Now the next theorem follows.

Theorem 2 Let $\mathcal{O},\mathcal{N} \in \{T, CT, F^1, F, R, FT, RT, PF, S, CS, RS, PW, 2S, B, U\}$. Then $\mathcal{O} \not\preceq \mathcal{N}$, and even $\mathcal{O} \not\preceq_\mathbb{H} \mathcal{N}$, unless $\mathcal{O} \preceq \mathcal{N}$ follows from Theorem 1 (and the fact that $\preceq$ is a partial order).

The following theorem says that the inclusion hierarchy of the preorders *T, CT, F^1, F, R, FT, RT, PF, S, CS, RS, PW, 2S* and *B* is the same as the inclusion hierarchy of the corresponding equivalences (there is no preorder for U).

Theorem 3 Let $\mathcal{O},\mathcal{N} \in \{T, CT, F^1, F, R, FT, RT, PF, S, CS, RS, PW, 2S, B\}$. Then $\mathcal{O} \preceq^* \mathcal{N}$ iff $\mathcal{O} \preceq \mathcal{N}$.

Proof: Clearly, if $\mathcal{L}_\mathcal{O}$ is a sublanguage of $\mathcal{L}_\mathcal{N}$ it must be that $p \sqsubseteq_\mathcal{N} q \Rightarrow p \sqsubseteq_\mathcal{O} q$, i.e. $\mathcal{O} \preceq^* \mathcal{N}$. This yields "if" (except for $RT \preceq^* PW \preceq^* RS$, which have been established in Section 14). "Only if" is immediate (cf. Section 1.4). □

When the restriction to finitely branching processes is dropped, there exists a finitary and an infinitary variant of each of these semantics, depending on whether or not infinite observations are taken into account (I do not consider the finitary version of PW or the infinitary version of F^1 though). These versions are notationally distinguished by means of superscripts "*" and "∞", respectively; the unsubscripted abbreviation will refer to the infinitary versions in case of simulation-like semantics (treated in Sections 8–12) and to the finitary versions for the *decorated*

[11] The statements involving PW and U do not follow this way, but have been established in Sections 13 and 14.

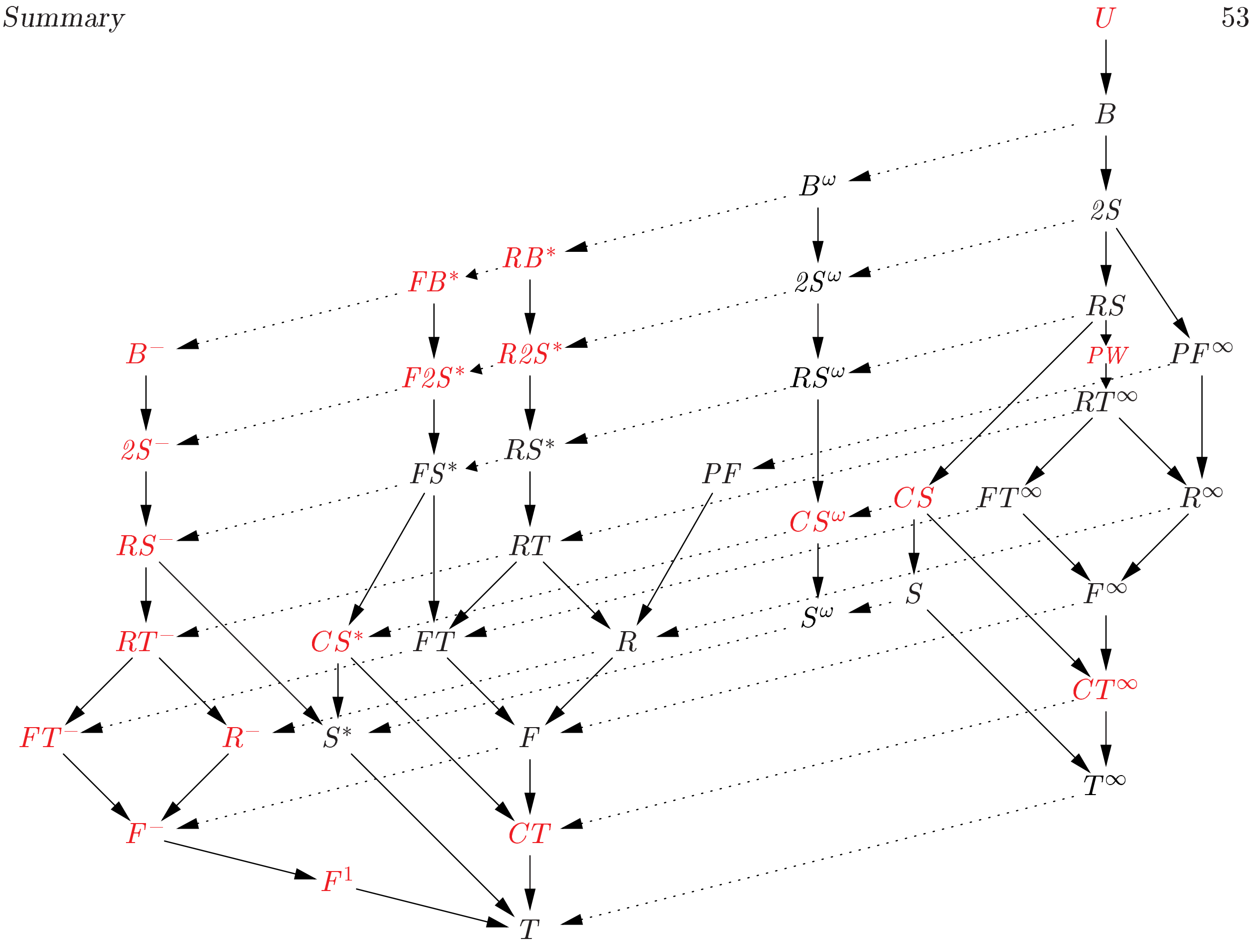


Figure 9: The infinitary linear time – branching time spectrum

trace semantics (treated in Section 2–7). The modal characterizations summarized above apply to the default (= unsubscripted) versions. Modal characterizations of T^∞, CT^∞, F^∞, R^∞, resp. FT^∞ and RT^∞, are obtained by allowing traces, resp. failure traces or ready traces, of infinite length as modal formulas; a modal characterization of PF^∞ is obtained by replacing the reference to T by one to T^∞. Modal characterizations of S^*, CS^*, etc. are obtained by requiring the index sets I to be finite. For the simulation-like semantics also an intermediate variant is considered—superscripted with "ω"—based on the assumption that observers can investigate arbitrary many copies of a process in parallel, but have only a finite amount of time to do so. Modally, this corresponds to the restriction to modal formulas with a finite upperbound on the number of nestings of the $a\varphi$ construct. For the semantics that incorporate refusal sets, the finitary versions come in two variants, depending on whether the refusal sets are required to be finite (superscript "$-$") or not (the default assumption). A similar distinction is made for semantics where menus of actions can be observed: in R^-, RT^- and RS^- the modal formula X is replaced by $\bigwedge_{a\in Y}\neg a\top \wedge \bigwedge_{a\in Z} a\top$, where the sets of actions Y and Z are required to be finite. Finally, whereas failure simulation semantics, modally characterized by

$\mathcal{L}_{FS}\quad \varphi ::= a\varphi' \ (\varphi' \in \mathcal{L}_{FS}) \mid \bigwedge_{i\in I}\varphi_i \ (\varphi_i \in \mathcal{L}_{FS}) \mid \widetilde{X} \ (X \subseteq Act)$ the *failure simulation formulas*,

coincides with ready simulation semantics, its finitary version (FS^*) can be distinguished from RS^*. The intermediate notions FS^ω and RS^ω coincide again, as do FS^- and RS^-. By analogy, new semantics FB^* and RB^* can be defined by adding the modality $\widetilde{X}$ resp. X to $\mathcal{L}_B^-$. These modalities would be redundant on top of $\mathcal{L}_B^\omega$ or $\mathcal{L}_B$. A similar situation occurs for 2-nested simulation.

All semantics encountered are displayed in Figure 9, in which the $\preceq$-relation is represented by solid and dotted arrows.

Theorem 4 For all semantics $\mathcal{O}$ and $\mathcal{N}$ defined so far, the formula $\mathcal{O} \preceq \mathcal{N}$ holds iff there is a path $\mathcal{O} \to \cdots \to \mathcal{N}$ (consisting of solid and dotted arrows alike) in Figure 9. Furthermore, semantics connected by dotted arrows coincide for image finite processes.

Proof: That $T^\infty \preceq S$ has been established in Proposition 8.3; that $CT^\infty \preceq CS$, $RT^\infty \preceq RS$ and $PF^\infty \preceq 2S$ follows in the same way. $R^\infty \preceq PW \preceq RS$ has been established in Section 14. All other implications $\mathcal{O} \preceq \mathcal{N}$ follow from the observation that the modal language $\mathcal{L}_\mathcal{O}$ is included in $\mathcal{L}_\mathcal{N}$. The latter statement has been established in Propositions 2.4, 4.6, 5.3, 6.5, 7.4, 7.5, 7.6, 8.4, 9.6, 11.2 and 12.6 (except that the case of possible-futures semantics is left to the reader). In order to show that on $\mathbb{G}$ there are no inclusions that are not indicated in Figure 9, is suffices, in view of Theorem 2, the already established parts of Theorem 4, and the fact that $\preceq$ is a partial order, to show that $CT \not\preceq B^-$, $R \not\preceq FB^*$, $S^\omega \not\preceq RB^*$, $T^\infty \not\preceq B^\omega$, $PF \not\preceq B^\omega$ and $T^\infty \not\preceq PF$. This has been done in Propositions 12.7, 12.8, 12.9 and 12.10, and at the end of Section 7. $\square$

Again, the inclusion hierarchy for the preorders is the same as for the equivalences.

Theorem 5 For all semantics $\mathcal{O}$ and $\mathcal{N}$ defined so far, the formula $\mathcal{O} \preceq^* \mathcal{N}$ holds iff there is a path $\mathcal{O} \to \cdots \to \mathcal{N}$ (consisting of solid and dotted arrows alike) in Figure 9.

Proof: That $p \sqsubseteq_S q \Rightarrow p \sqsubseteq_T^\infty q$ has been established in Proposition 8.3; that $p \sqsubseteq_{CS} q \Rightarrow p \sqsubseteq_{CT}^\infty q$, $p \sqsubseteq_{RS} q \Rightarrow p \sqsubseteq_{RT}^\infty q$ and $p \sqsubseteq_{2S} q \Rightarrow p \sqsubseteq_{PF}^\infty q$ follows in the same way. In Section 14 it has been established that $p \sqsubseteq_{RS} q \Rightarrow p \sqsubseteq_{PW} q \Rightarrow p \sqsubseteq_{RT}^\infty q$. All other implications $p \sqsubseteq_\mathcal{O} q \Rightarrow p \sqsubseteq_\mathcal{N} q$ follow from the observation that the modal language $\mathcal{L}_\mathcal{O}$ is included in $\mathcal{L}_\mathcal{N}$. The "only if" part is an immediate consequence of Theorem 4. $\square$

16 Deterministic and saturated processes

If the labelled transition system $\mathbb{P}$ on which the semantic equivalences of Section 15 are defined is large enough, then they are all different and $\mathcal{O} \preceq_\mathbb{P} \mathcal{N}$ holds only if this is indicated in Figure 9. However, for certain labelled transition systems much more identifications can be made. Is has been remarked already that for image finite processes all semantics that are connected by dotted arrows coincide. In this section various other classes of processes are examined on which parts of the linear time – branching time spectrum collapse. All results of this section, expect for Propositions 16.1 and 16.2, will be used in the completeness proofs in Section 17.

Recall that a process p is *deterministic* iff $p \xrightarrow{\sigma} q \wedge p \xrightarrow{\sigma} r \Rightarrow q = r$.

Remark: If p is deterministic and $p \xrightarrow{\sigma} p'$ then also p' is deterministic. Hence any domain of processes on which action relations are defined, has a subdomain of deterministic processes with the inherited action relations. (A similar remark can be made for image finite processes.)
Proof: Suppose $p' \xrightarrow{\rho} q$ and $p' \xrightarrow{\rho} r$. Then $p \xrightarrow{\sigma\rho} q$ and $p \xrightarrow{\sigma\rho} r$, so $q = r$. $\square$

Theorem 6 (PARK [41]) On a domain of deterministic processes all semantics in the infinitary linear time – branching time spectrum coincide.

Proof: Because of Theorem 4 it suffices to show that $g =_T h \Rightarrow g =_U h$ for any two deterministic process graphs $g, h \in \mathbb{G}$. Note that a process graph $g \in \mathbb{G}$ is deterministic iff for every trace $\sigma \in T(g)$ there is exactly one path $\pi \in \text{PATHS}(g)$ with $T(\pi) = \sigma$. Now let g and h be deterministic process graphs with $g =_T h$. Then the relation $i \subseteq \text{PATHS}(g) \times \text{PATHS}(h)$ that relates $\pi \in \text{PATHS}(g)$ with $\pi' \in \text{PATHS}(h)$ iff $T(\pi) = T(\pi')$ clearly is an isomorphism between $U(g)$ and $U(h)$. □

Thus, if two processes p and q are both deterministic, then $p =_T q \Leftrightarrow p =_F^1 q \Leftrightarrow p \leftrightarroweq q \Leftrightarrow p =_U q$. In case only one of them is deterministic, this cannot be concluded, for in Counterexamples 2 and 15 the right-hand processes are deterministic. However, in such cases one still has $p =_F^1 q \Leftrightarrow p \leftrightarroweq q$. In fact, a stronger statement holds: if q is deterministic, then $p \sqsubseteq_F^1 q \Leftrightarrow p \leftrightarroweq q$.

Lemma 16.1 If $p \sqsubseteq_F^1 q$ then $I(p) = I(q)$.
Proof: Let $p \sqsubseteq_F^1 q$, i.e. $T(p) \subseteq T(q)$ and $F^1(p) \subseteq F^1(q)$. Then $a \in I(p) \Leftrightarrow a \in T(p) \Rightarrow a \in T(q) \Leftrightarrow a \in I(q)$ and $a \notin I(p) \Leftrightarrow \langle \varepsilon, a \rangle \in F^1(p) \Rightarrow \langle \varepsilon, a \rangle \in F^1(q) \Leftrightarrow a \notin I(q)$. □

Proposition 16.1 If q is deterministic then $p \sqsubseteq_F^1 q \Leftrightarrow p \leftrightarroweq q$.
Proof: Let R be the binary relation on $\mathbb{P}$ defined by pRq iff q is deterministic and $p \sqsubseteq_F^1 q$, then it suffices to prove that R is a bisimulation. Suppose pRq and $p \xrightarrow{a} p'$. Then $a \in I(p) = I(q)$. So there is a process $q' \in \mathbb{P}$ with $q \xrightarrow{a} q'$. As q is deterministic, so is q'. Now let $\langle \sigma, b \rangle \in F^1(p')$. Then $\exists r : p' \xrightarrow{\sigma} r \wedge b \notin I(r)$. Hence $p \xrightarrow{a\sigma} r$ and $\langle a\sigma, b \rangle \in F^1(p) \subseteq F^1(q)$. So there must be a process s with $q \xrightarrow{a\sigma} s \wedge b \notin I(s)$. By the definition of the generalized action relations $\exists t : q \xrightarrow{a} t \xrightarrow{\sigma} s$, and since q is deterministic, $t = q'$. Thus $\langle \sigma, b \rangle \in F^1(q')$. From this it follows that $F^1(p') \subseteq F^1(q')$. Similarly one finds $T(p') \subseteq T(q')$, hence $p' \sqsubseteq_F^1 q'$.

Now suppose pRq and $q \xrightarrow{a} q'$. Then $a \in I(q) = I(p)$. So there is a process $p' \in \mathbb{P}$ with $p \xrightarrow{a} p'$. Exactly as above it follows that q' is deterministic and $p' \sqsubseteq_F^1 q'$. □

Call a process p *deterministic up to* $\equiv$, for $\equiv$ an equivalence relation or preorder, if there exists a deterministic process p' with $p \equiv p'$. Now the above proposition implies that determinism up to $\leftrightarroweq$ coincides with determinism up to $=_F^1$, and even with determinism up to $\sqsubseteq_F^1$. In contrast, *any* process is deterministic up to $=_T$, as the canonical graphs constructed in the proof of Proposition 2.3 are deterministic. Furthermore, determinism up to $=_U$ is just determinism, for $g \in \mathbb{G}$ is deterministic iff $U(g)$ is, and determinism is preserved under isomorphism.

The following notion of *determinacy* was proposed in ENGELFRIET [18].

Definition 16.1 Let $\equiv$ be an equivalence relation on $\mathbb{P}$. A process $p \in \mathbb{P}$ is $\equiv$-*determinate* if $p \xrightarrow{\sigma} q \wedge p \xrightarrow{\sigma} r \Rightarrow q \equiv r$.

Note that $=$-determinacy is determinism. Furthermore, if $\mathcal{O} \leq \mathcal{N}$ then $=_\mathcal{O}$-determinacy is implied by $=_\mathcal{N}$-determinacy. Besides $=_T$-determinacy, $=_F$-determinacy and $\leftrightarroweq$-determinacy, Engelfriet also considers $=_I$-determinacy, where $=_I$ is given by $p =_I q$ iff $I(p) = I(q)$. Clearly $=_I$ is coarser than any of the equivalences of Section 15: $p =_T q \Rightarrow p =_I q$. Moreover, $=_I$ is even coarser than most of the preorders: $p \sqsubseteq_F^1 q \Rightarrow p =_I q$, as established in Lemma 16.1.

Engelfriet established the following three results:

(1) $\underline{\leftrightarrow}$-determinacy and $=_I$-determinacy are the same. Hence $\equiv$-determinacy is the same for all equivalences $\equiv$ of Section 15, except U. Therefore, he just calls this *determinacy.*

(2) For determinate processes, bisimulation equivalence and trace equivalence (and hence all equivalences in between) are the same.

(3) Determinacy is preserved under failures equivalence (and hence under $\underline{\leftrightarrow}$). Even stronger, if q is determinate and $p \sqsubseteq_F q$, then p is determinate and $p \underline{\leftrightarrow} q$. (In [18], $\sqsupseteq_F$ is written $\subseteq_{\mathrm{f}}$.)

Using Proposition 16.1 I show that both $=_I$-determinacy and $\underline{\leftrightarrow}$-determinacy coincide with determinism up to $\underline{\leftrightarrow}$, from which (1), (2) and (3) follow.

Proposition 16.2 Let $p \in \mathbb{P}$. The following are equivalent:
(a) p is $\underline{\leftrightarrow}$-determinate
(b) p is $=_I$-determinate
(c) p is deterministic up to $=_R$
(d) p is deterministic up to $\underline{\leftrightarrow}$.

Proof: "(a) $\Rightarrow$ (b)" is immediate as $=_I$ is coarser than $\underline{\leftrightarrow}$.

"(b) $\Rightarrow$ (c)": Suppose p is $=_I$-determinate. Let $G(T(p))$ be the canonical graph of the trace set of p as defined in the proof of Proposition 2.3. By construction, $G(T(p))$ is deterministic and $T(p) = T(G(T(p)))$. It remains to be shown that $p =_R G(T(p))$.

As p is $=_I$-determinate, one has $\langle\sigma, X\rangle, \langle\sigma, Y\rangle \in R(p) \Rightarrow X = Y$. Hence $\langle\sigma, X\rangle \in R(p)$ iff $\sigma \in T(p) \wedge X = \{a \in Act \mid \sigma a \in T(p)\}$, i.e. $R(p)$ is completely determined by $T(p)$. As also $G(T(p))$ is $=_I$-determinate (for it is even deterministic), also $R(G(T(p)))$ is completely determined by $T(G(T(p)))$: $\langle\sigma, X\rangle \in R(G(T(p)))$ iff $\sigma \in T(G(T(p))) \wedge X = \{a \in Act \mid \sigma a \in T(G(T(p)))\}$. It follows that $R(p) = R(G(T(p)))$.

"(c) $\Rightarrow$ (d)" has been established in Proposition 16.1.

"(d) $\Rightarrow$ (a)": Suppose $p \underline{\leftrightarrow} q$ and q is deterministic. Let $p \xrightarrow{\sigma} p'$ and $p \xrightarrow{\sigma} p''$. Then $\exists q' : q \xrightarrow{\sigma} q' \wedge p' \underline{\leftrightarrow} q'$ and $\exists q'' : q \xrightarrow{\sigma} q'' \wedge p'' \underline{\leftrightarrow} q''$. As q is deterministic, $q' = q''$. Hence $p' \underline{\leftrightarrow} p''$. It follows that p is $\underline{\leftrightarrow}$-determinate. $\square$

Now (1) is part of Proposition 16.2. (2) is a generalization of Theorem 6, that is now implied by it: Suppose p and q are determinate and $p =_T q$. By Proposition 16.2 there are deterministic processes p' with $p \underline{\leftrightarrow} p'$ and q' with $q \underline{\leftrightarrow} q'$. Hence $p' =_T q'$, so by Theorem 6 $p' \underline{\leftrightarrow} q'$. Thus $p \underline{\leftrightarrow} q$, yielding (2). (3) holds even for F^1 instead of F. For let q be determinate and $p \sqsubseteq_F^1 q$. Then there is a deterministic process q' with $q \underline{\leftrightarrow} q'$. Hence $p \sqsubseteq_F^1 q'$. By Proposition 16.1 $p \underline{\leftrightarrow} q'$, so p is determinate and $p \underline{\leftrightarrow} q$.

Note that a process p is deterministic iff for $\pi, \pi' \in \textsc{paths}(G(p))$ one has $T(\pi) = T(\pi') \Rightarrow \pi = \pi'$. For this reason, determinism could have been called *trace determinism*, and the notions of *ready trace determinism* and *completed trace determinism* can be defined analogously.

Definition 16.2 A process p is *ready trace deterministic* if for $\pi, \pi' \in \textsc{paths}(G(p))$ one has $RT(\pi) = RT(\pi') \Rightarrow \pi = \pi'$. It is *completed trace deterministic* if for $\pi, \pi' \in \textsc{paths}(G(p))$ one has $T(\pi) = T(\pi') \wedge \Big(I(end(\pi)) = \emptyset \Leftrightarrow I(end(\pi')) = \emptyset\Big) \Rightarrow \pi = \pi'$.

A process $p \in \mathbb{P}$ is ready trace deterministic iff there is are no $p', q, r \in \mathbb{P}$ and $a \in Act$ such that p' is reachable from p, $p' \xrightarrow{a} q$, $p' \xrightarrow{a} r$, $I(q) = I(r)$ and $q \neq r$. For trace determinism the condition $I(q) = I(r)$ is dropped, and for completed trace determinism it is weakened to

$I(q) = \emptyset \Leftrightarrow I(r) = \emptyset$. Note that if p is ready (or completed) trace deterministic and $p \xrightarrow{\sigma} p'$ then so is p'. Now the following variants of Theorem 6 can be established.

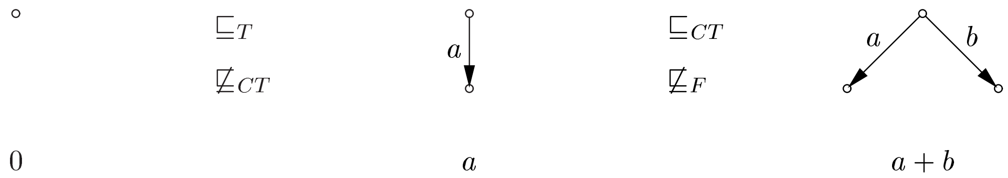
Proposition 16.3 If g and $h \in \mathbb{G}$ are ready trace deterministic then $g =_{RT} h \Leftrightarrow g =_U h$. Likewise, if g and $h \in \mathbb{G}$ are completed trace deterministic then $g =_{CT} h \Leftrightarrow g =_U h$.
Proof: Let g and h be ready trace deterministic process graphs with $g =_{RT} h$. Then the relation $i \subseteq \text{PATHS}(g) \times \text{PATHS}(h)$ that relates $\pi \in \text{PATHS}(g)$ with $\pi' \in \text{PATHS}(h)$ iff $RT(\pi) = RT(\pi')$ clearly is an isomorphism between $U(g)$ and $U(h)$. The proof of the second statement goes likewise. □

For completed trace deterministic processes, the equivalences $=_T$ and $=_{CT}$ are different, as can be seen from Counterexample 2. For ready trace deterministic processes, the equivalences $=_T$, $=_{CT}$, $=^1_F$, $=_F$, $=_{FT}$, $=_R$, $=_{RT}$, $=_S$ and $=_{CS}$ are all different, as can be seen from Counterexamples 2, 3, 5, 6 and 13. Theorem 6 and Proposition 16.3 do not generalize to the corresponding preorders, for in Counterexample 19 one finds two deterministic processes *middle* and *right* with *middle* $\sqsubseteq_{CT}$ *right* but *middle* $\not\sqsubseteq_B$ *right*, and in Counterexample 3 one finds two ready trace deterministic processes *right* and *left* with *right* $\sqsubseteq_{RT}$ *left* but *right* $\not\sqsubseteq_B$ *left*. However, the following variants of these results can be obtained.

Proposition 16.4 If q is ready trace deterministic then $p \sqsubseteq_{RT} q \Leftrightarrow p \sqsubseteq_{RS} q$.
Likewise, if q is completed trace deterministic then $p \sqsubseteq_{CT} q \Leftrightarrow p \sqsubseteq_{CS} q$,
and if q is (trace) deterministic then $p \sqsubseteq_T q \Leftrightarrow p \sqsubseteq_S q$.
Proof: Let R be the binary relation on $\mathbb{P}$ given by pRq if q is ready trace deterministic and $p \sqsubseteq_{RT} q$. For the first statement it suffices to prove that R is a ready simulation. Clearly pRq implies $I(p) = I(q)$. Now suppose pRq and $p \xrightarrow{a} p'$. Let X be $I(p')$. Then $aX \in RT(p) \subseteq RT(q)$. So there is a process $q' \in \mathbb{P}$ with $q \xrightarrow{a} q'$ and $I(q') = X$. Now let $\sigma \in RT(p')$. Then $\exists r : p' \overset{\sigma}{\ast\!\!\longrightarrow} r$. Hence $p \overset{aX\sigma}{\ast\!\!\longrightarrow} r$ and $aX\sigma \in RT(p) \subseteq RT(q)$. So there must be a process s with $q \overset{aX\sigma}{\ast\!\!\longrightarrow} s$. By the definition of the ready trace relations $\exists t : q \xrightarrow{a} t \overset{\sigma}{\ast\!\!\longrightarrow} s \wedge I(t) = X$, and since q is ready trace deterministic, $t = q'$. Thus $\sigma \in RT(q')$. From this it follows that $RT(p') \subseteq RT(q')$, implying $p'Rq'$.

This finishes the proof of the first statement. The proofs of the other two statements go the same way (but involving a trivial case distinction for the completed trace deterministic one). □

Together, Propositions 16.1 and 16.4 imply that on a domain of deterministic processes only three of the preorders of Section 15 are different, namely $\sqsubseteq_T$, $\sqsubseteq_{CT}$ and $\sqsubseteq^1_F$, coinciding with $\sqsubseteq_S$, $\sqsubseteq_{CS}$ and $\sqsubseteq_B$, respectively. That these three are indeed different is shown in Counterexample 19.



Counterexample 19: The trace, completed trace, and failures preorders are all different on deterministic processes

Definition 16.3 A process p is *cross saturated* if $p \xrightarrow{\sigma} q \xrightarrow{a} r \wedge p \xrightarrow{\sigma} s \xrightarrow{a} t \Rightarrow q \xrightarrow{a} t$.

Thus a process graph $g \in \mathbb{G}$ is cross saturated iff for any $\pi, \pi' \in \text{PATHS}(g)$ and $a \in Act$ such that $a \in I(end(\pi))$ and $T(\pi') = T(\pi)a$ one has $(end(\pi), a, end(\pi')) \in \text{EDGES}(g)$.

Proposition 16.5 If g and $h \in \mathbb{G}$ are cross saturated then $g =_R h \Leftrightarrow g \,\underline{\leftrightarrow}\, h$.
Proof: Let g and $h \in \mathbb{G}$ be cross saturated and suppose that $R(g) = R(h)$. Define the relation $R \subseteq \text{NODES}(g) \times \text{NODES}(h)$ by sRt iff there are $\pi \in \text{PATHS}(g)$ and $\rho \in \text{PATHS}(h)$ with $end(\pi) = s$, $end(\rho) = t$ and $R(\pi) = R(\rho)$. It suffices to show that R is a bisimulation between g and h.
As $I(\text{ROOT}(g)) = I(\text{ROOT}(h))$ one clearly has $\text{ROOT}(g) R \text{ROOT}(h)$.
Now suppose sRt and $(s, a, s') \in \text{EDGES}(g)$. Let π and ρ be such that $end(\pi) = s$, $end(\rho) = t$ and $R(\pi) = R(\rho)$, and let π' be the extension of π with (s, a, s'). Now $a \in I(end(\pi)) = I(end(\rho))$. Choose $\rho' \in \text{PATHS}(h)$ with $R(\rho') = R(\pi')$ (using that $R(g) = R(h)$). Then $T(\rho') = T(\pi') = T(\pi)a = T(\rho)a$, so $(t, a, end(\rho')) \in \text{EDGES}(h)$. Moreover, $s' R\, end(\rho')$.
The remaining requirement of Definition 12.2 follows by symmetry. □

Proposition 16.6 If $h \in \mathbb{G}$ is cross saturated then $g \sqsubseteq_R h \Leftrightarrow g \sqsubseteq_{RS} h$.
Proof: Exactly as above. □

Definition 16.4 A process p is *saturated* if $\langle\sigma, X\rangle \in R(p) \wedge \langle\sigma, Y \cup Z\rangle \in R(p) \Rightarrow \langle\sigma, X \cup Y\rangle \in R(p)$.

Proposition 16.7 If p is finitely branching and q is saturated then $p \sqsubseteq_F q \Leftrightarrow p \sqsubseteq_R q$. Thus if both p and q are finitely branching and saturated then $p =_F q \Leftrightarrow p =_R q$.
Proof: Suppose p is finitely branching, q is saturated and $p \sqsubseteq_F q$. Let $\langle\sigma, Y\rangle \in R(p)$. Then Y is finite. In case $Y = \emptyset$ one has $\langle\sigma, Act\rangle \in F(p) \subseteq F(q)$, implying $\langle\sigma, \emptyset\rangle \in R(q)$, as desired. So assume $Y \neq \emptyset$. Then, for all $a \in Y$, $\langle\sigma a, \emptyset\rangle \in F(p) \subseteq F(q)$ so $\exists Z_a \subseteq Act$ with $\langle\sigma, \{a\} \cup Z_a\rangle \in R(q)$. Hence, using Definition 16.4 with $Z = \emptyset$, one obtains $\langle\sigma, Y \cup \bigcup_{a\in Y} Z_a\rangle \in R(q)$. As $\langle\sigma, Act - Y\rangle \in F(p) \subseteq F(q)$ it must be that $\langle\sigma, X\rangle \in R(q)$ for some $X \subseteq Y$. Now Definition 16.4 gives $\langle\sigma, Y\rangle \in R(q)$. □

Definition 16.5 A process p is *RT-saturated* if

$$\sigma X \rho \in RT_N(p) \wedge \sigma Y \in RT_N(p) \Rightarrow \sigma(X \cup Y)\rho \in RT_N(p).$$

Proposition 16.8 If p is finitely branching and q is RT-saturated then $p \sqsubseteq_{FT} q \Leftrightarrow p \sqsubseteq_{RT} q$. Thus if both p and q are finitely branching and RT-saturated then $p =_{FT} q \Leftrightarrow p =_{RT} q$.
Proof: Suppose p is finitely branching, q is RT-saturated and $p \sqsubseteq_{FT} q$. With induction on $k \in \mathbb{N}$ I will show that whenever $X_0 a_1 X_1 a_2 \cdots a_n X_n \in RT(p)$ then there are $Y_i \subseteq X_i$ for $i = k+1, \ldots, n$ such that $X_0 a_1 X_1 a_2 \cdots a_k X_k a_{k+1} Y_{k+1} a_{k+2} \cdots a_n Y_n \in RT(q)$. The case $k = n$, together with Proposition 6.1, completes the proof of the proposition.
Induction base ($k = 0$): Let $X_0 a_1 X_1 a_2 \cdots a_n X_n \in RT(p)$. Write $\overline{X}$ for $Act - X$.
Then $\overline{X_0} a_1 \overline{X_1} a_2 \cdots a_n \overline{X_n} \in FT(p) \subseteq FT(q)$. Hence there are $Y_i \subseteq X_i$ for $i = 0, ..., n$ such that $Y_0 a_1 Y_1 a_2 \cdots a_n Y_n \in RT(q)$. As $p \sqsubseteq_{FT} q \Rightarrow p \sqsubseteq_T q \Rightarrow I(p) \subseteq I(q)$, we have $Y_0 = X_0$.
Induction step: Take $k > 0$ and suppose the statement has been established for $k - 1$.
Let $X_0 a_1 X_1 a_2 \cdots a_n X_n \in RT(p)$. Then, by induction, there are $Y_i \subseteq X_i$ for $i = k, ..., n$ such that $X_0 a_1 X_1 a_2 \cdots a_{k-1} X_{k-1} a_k Y_k a_{k+1} Y_{k+1} a_{k+2} \cdots a_n Y_n \in RT(q)$. Moreover, for every $b \in X_k$, $X_0 a_1 X_1 a_2 \cdots a_k X_k b \in RT(p)$, so, again using the induction hypothesis, there must be a $Z_b \subseteq X_k$ such that $X_0 a_1 X_1 \cdots X_{k-1} a_k Z_b b \in RT(q)$, and hence $X_0 a_1 X_1 \cdots X_{k-1} a_k (Z_b \cup \{b\}) \in RT(q)$. As X_k is finite and $Y_k \cup \bigcup_{b \in X_k} (Z_b \cup \{b\}) = X_k$, the RT-saturation of q gives
$X_0 a_1 \cdots a_k X_k a_{k+1} Y_{k+1} a_{k+2} \cdots a_n Y_n \in RT(q)$, which had to be established. □

17 Complete axiomatizations

17.1 A language for finite, concrete, sequential processes

Consider the following basic CCS- and CSP-like language BCCSP for finite, concrete, sequential processes over a given alphabet *Act*:

inaction : 0 (called *nil* or *stop*) is a constant, representing a process that refuses to do any action.

action : a is a unary operator for any action $a \in Act$. The expression ap represents a process, starting with an a-action and proceeding with p.

choice : $+$ is a binary operator. $p + q$ represents a process, first being involved in a choice between its summands p and q, and then proceeding as the chosen process.

The set $\mathsf{T}(\text{BCCSP})$ of (closed) *process expressions* or *terms* over this language is defined as usual:

- $0 \in \mathsf{T}(\text{BCCSP})$,
- $ap \in \mathsf{T}(\text{BCCSP})$ for any $a \in Act$ and $p \in \mathsf{T}(\text{BCCSP})$,
- $p + q \in \mathsf{T}(\text{BCCSP})$ for any $p, q \in \mathsf{T}(\text{BCCSP})$.

Subterms $a0$ may be abbreviated by a. Brackets are used for disambiguation only, assuming associativity of $+$, and letting a bind stronger than $+$. If $P = \{p_1, ..., p_n\}$ is a finite nonempty multiset of BCCSP expressions, then ΣP abbreviates $p_1 + \cdots + p_n$. This expression is determined only up to associativity and commutativity of $+$. Let $\Sigma\emptyset := 0$. An expression ap' is called a *summand* of p if, up to associativity and commutativity of $+$, p can be written as ΣP with $ap' \in P$.

On $\mathsf{T}(\text{BCCSP})$ action relations $\xrightarrow{a}$ for $a \in Act$ are defined as the predicates on $\mathsf{T}(\text{BCCSP})$ generated by the *action rules* of Table 1. Here a ranges over Act and p and q over $\mathsf{T}(\text{BCCSP})$.

$$ap \xrightarrow{a} p \qquad \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'} \qquad \frac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'}$$

Table 1: Action rules for BCCSP

A trivial structural induction shows that $p \xrightarrow{a} p'$ iff ap' is a summand of p. Now all semantic equivalences of Sections 2–14 are well-defined on $\mathsf{T}(\text{BCCSP})$, and for each of the semantics it is determined when two process expressions denote the same process.

The following theorem says that, apart from U, all these semantics are *compositional* w.r.t. BCCSP, i.e. all semantic equivalences are *congruences* for BCCSP.

Theorem 7 Let $p, q, r, s \in \mathsf{T}(\text{BCCSP})$ and let $\mathcal{O}$ be any of the semantics of Section 15 except U. Then

$$p =_{\mathcal{O}} q \ \wedge \ r =_{\mathcal{O}} s \quad \Rightarrow \quad ap =_{\mathcal{O}} aq \ \wedge \ p + r =_{\mathcal{O}} q + s.$$

Proof: Each of the semantics $\mathcal{O}$ has a modal characterization, given by $p =_{\mathcal{O}} q \Leftrightarrow \mathcal{O}(p) = \mathcal{O}(q)$, where $\mathcal{O}(p)$ is the set of modal formulas of the appropriate form satisfied by p. Let $\mathcal{O}^+(p) := \{a\varphi \mid a\varphi \in \mathcal{O}(p)\}$ be the set of such formulas which are of the form $a\varphi$. For each choice of $\mathcal{O}$ one easily verifies that $\mathcal{O}(p)$ is completely determined by $\mathcal{O}^+(p)$, i.e. $\mathcal{O}(p) = \mathcal{O}(q) \Leftrightarrow \mathcal{O}^+(p) = \mathcal{O}^+(q)$. One also verifies easily that $\mathcal{O}^+(0) = \emptyset$, $\mathcal{O}^+(ap) = \{a\varphi \mid \varphi \in \mathcal{O}(p)\}$ and $\mathcal{O}^+(p+q) = \mathcal{O}^+(p) \cup \mathcal{O}^+(q)$. From this the theorem follows immediately. □

For each such choice of $\mathcal{O}$ one easily verifies that moreover $\mathcal{O}(p) \subseteq \mathcal{O}(q) \Leftrightarrow \mathcal{O}^+(p) \subseteq \mathcal{O}^+(q)$. From this it follows that all the preorders of Section 15 are *precongruences* for BCCSP:

Theorem 7b Let $p, q, r, s \in \mathsf{T}(\text{BCCSP})$ and let $\mathcal{O}$ be any of the semantics of Section 15 but U. Then

$$p \sqsubseteq_{\mathcal{O}} q \ \wedge \ r =_{\mathcal{O}} s \quad \Rightarrow \quad ap \sqsubseteq_{\mathcal{O}} aq \ \wedge \ p + r \sqsubseteq_{\mathcal{O}} q + s. \qquad \square$$

Tree semantics, when defined merely in terms of the action relations on $\mathsf{T}(\text{BCCSP})$, fails to be compositional w.r.t. BCCSP. The expression $a0 + a0$ has only a single outgoing a-transition, namely to the expression 0. Thus, by Definition 13, $a0 + a0 =_U a0$. Likewise $b(a0 + a0) =_U ba0$. However, $b(a0 + a0) + b0 \neq_U ba0 + ba0$, as the first process has two outgoing b-transitions and the second process only one. If follows that tree equivalence as defined above is not compositional w.r.t. $+$.

$\mathsf{T}(\text{BCCSP})$ can be turned into a labelled transition system with multiplicities by assuming a different transition $p \xrightarrow{a} q$ for every different proof of $p \xrightarrow{a} q$ from the action rules of Table 1. On such a transition system tree equivalence is compositional w.r.t. BCCSP.

A straightforward structural induction shows that any process $p \in \mathsf{T}(\text{BCCSP})$ is finite in the sense of Definition 1.2. Hence the process graph $G(p)$ is finite as well. The next proposition establishes that moreover, up to bisimulation equivalence, any finite process graph can be represented by a BCCSP expression. In fact, all finite process graphs displayed in this paper have been annotated by their representing BCCSP expressions.

Definition 17.1 Let $\langle\!\langle \cdot \rangle\!\rangle : \mathbb{H} \to \mathsf{T}(\text{BCCSP})$ be a mapping satisfying $\langle\!\langle g \rangle\!\rangle = \Sigma\{a\langle\!\langle h \rangle\!\rangle \mid g \xrightarrow{a} h\}$.

A straightforward induction on the length of the longest path of finite process graphs teaches that such a mapping exists and is completely determined up to associativity and commutativity of $+$.

Proposition 17.1 Let $g \in \mathbb{H}$. Then there is a $p \in \mathsf{T}(\text{BCCSP})$ with $G(p) \leftrightarroweq g$. In fact, $G(\langle\!\langle g \rangle\!\rangle) \leftrightarroweq g$.
Proof: It suffices to show that the relation $\{h, G(\langle\!\langle h \rangle\!\rangle) \mid h \in \mathbb{H}\}$ is a bisimulation. Suppose $h \xrightarrow{a} h'$. Then $a\langle\!\langle h' \rangle\!\rangle$ is a summand of $\langle\!\langle h \rangle\!\rangle$, so $\langle\!\langle h \rangle\!\rangle \xrightarrow{a} \langle\!\langle h' \rangle\!\rangle$, and by Proposition 1.1 $G(\langle\!\langle h \rangle\!\rangle) \xrightarrow{a} G(\langle\!\langle h' \rangle\!\rangle)$. Vice versa, let $G(\langle\!\langle h \rangle\!\rangle) \xrightarrow{a} h''$. Then, by Proposition 1.1, $h'' = G(p')$ for some $p' \in \mathsf{T}(\text{BCCSP})$ with $\langle\!\langle h \rangle\!\rangle \xrightarrow{a} p'$. Thus ap' must be a summand of $\langle\!\langle h \rangle\!\rangle$. By Definition 17.1 $p' = \langle\!\langle h' \rangle\!\rangle$ for some $h' \in \mathbb{H}$ with $h \xrightarrow{a} h'$. As h' is related to $h'' = G(\langle\!\langle h' \rangle\!\rangle)$, also this requirement is satisfied. $\square$

Corollary 17.1 Let $p \in \mathsf{T}(\text{BCCSP})$. Then $p \leftrightarroweq \langle\!\langle G(p) \rangle\!\rangle$.
Proof: By the above $G(\langle\!\langle G(p) \rangle\!\rangle) \leftrightarroweq G(p)$. Now apply Corollary 1.1. $\square$

Corollary 17.2 Let $g, h \in \mathbb{H}$ and let $\mathcal{O}$ be any of the semantics of Section 15. Then

$$g \sqsubseteq_{\mathcal{O}} h \Leftrightarrow \langle\!\langle g \rangle\!\rangle \sqsubseteq_{\mathcal{O}} \langle\!\langle h \rangle\!\rangle \qquad \text{and} \qquad g =_{\mathcal{O}} h \Leftrightarrow \langle\!\langle g \rangle\!\rangle =_{\mathcal{O}} \langle\!\langle h \rangle\!\rangle.$$

Proof: Let $g \sqsubseteq_{\mathcal{O}} h$. By the above $G(\langle\!\langle g \rangle\!\rangle) \leftrightarroweq g \sqsubseteq_{\mathcal{O}} h \leftrightarroweq G(\langle\!\langle h \rangle\!\rangle)$. Now apply Corollary 1.1.
For "$\Leftarrow$" let $\langle\!\langle g \rangle\!\rangle \sqsubseteq_{\mathcal{O}} \langle\!\langle h \rangle\!\rangle$. By Corollary 1.1 and Proposition 17.1 $g \leftrightarroweq G(\langle\!\langle g \rangle\!\rangle) \sqsubseteq_{\mathcal{O}} G(\langle\!\langle h \rangle\!\rangle) \leftrightarroweq h$. $\square$

17.2 Axiomatizing the equivalences

In Table 2, complete axiomatizations can be found for twelve of the fifteen semantic equivalences of this paper that differ on BCCSP. Axioms for singleton-failures, 2-nested simulation and possible-futures semantics are more cumbersome, and the corresponding testing notions are less plausible. Therefore they have been omitted. The axiomatization of tree semantics (U) requires action relations with multiplicities. Although rather trivial, I will not formally establish its soundness and completeness here. In order to formulate the axioms, variables have to be added to the language as usual. In the axioms they are supposed to be universally quantified. Most of the axioms are axiom schemes, in the sense that there is one axiom for each substitution of actions from *Act* for the parameters a, b, c. Some of the axioms are conditional equations, using an auxiliary operator I. Thus provability is defined according to the standards of either first-order logic with equality or conditional equational logic. I is a unary operator that calculates the set of initial actions of a process expression, coded as a process expression again.

Theorem 8 For each of the semantics $\mathcal{O} \in \{T,\ S,\ CT,\ CS,\ F,\ R,\ FT,\ RT,\ PW,\ RS,\ B\}$ two process expressions $p, q \in \mathsf{T}(\text{BCCSP})$ are $\mathcal{O}$-equivalent iff they can be proved equal from the axioms marked with "+" in the column for $\mathcal{O}$ in Table 2. The axioms marked with "v" or "ω" are valid in $\mathcal{O}$-semantics but not needed for the proof.

	U	B	RS	PW	RT	FT	R	F	CS	CT	S	T
$(x+y)+z \ = \ x+(y+z)$	+	+	+	+	+	+	+	+	+	+	+	+
$x+y \ = \ y+x$	+	+	+	+	+	+	+	+	+	+	+	+
$x+0 \ = \ x$	+	+	+	+	+	+	+	+	+	+	+	+
$x+x \ = \ x$		+	+	+	+	+	+	+	+	+	+	+
$I(x)=I(y) \ \Rightarrow \ a(x+y) \ = \ a(x+y)+ay$			+	v	v	v	v	v	v	v	v	v
$a(bx+by+z) \ = \ a(bx+z)+a(by+z)$				+	v	v	v	v		v		v
$I(x)=I(y) \ \Rightarrow \ ax+ay \ = \ a(x+y)$					+	+	v	v		v		v
$ax+ay \ = \ ax+ay+a(x+y)$						+		v		v		v
$a(bx+u)+a(by+v) = a(bx+by+u)+a(by+v)$							+	+		v		v
$ax+a(y+z) \ = \ ax+a(x+y)+a(y+z)$								+		ω		v
$a(x+by+z) \ = \ a(x+by+z)+a(by+z)$									+	v	v	v
$a(bx+u)+a(cy+v) \ = \ a(bx+cy+u+v)$										+		v
$a(x+y) \ = \ a(x+y)+ay$											+	v
$ax+ay \ = \ a(x+y)$												+
$I(0) \ = \ 0$	+	+	+	+	+	+	+	+	+	+	+	+
$I(ax) \ = \ a0$	+	+	+	+	+	+	+	+	+	+	+	+
$I(x+y) \ = \ I(x)+I(y)$	+	+	+	+	+	+	+	+	+	+	+	+

Table 2: Complete axiomatizations for the equivalences

Proof: "If" (*soundness*): In the light of Theorem 7 it suffices to show that the closed instances of the indicated axioms are valid in the corresponding semantics. This is straightforward.

"Only if" (*completeness*): Let $T_{\mathcal{O}}$ be the set of axioms marked with "+" in the column for $\mathcal{O}$. Write $T_{\mathcal{O}} \vdash p = q$ if the equation $p = q$ is provable from $T_{\mathcal{O}}$. I have to show that

$$p =_{\mathcal{O}} q \;\Rightarrow\; T_{\mathcal{O}} \vdash p = q \tag{2}$$

for any $p, q \in \mathsf{T}(\text{BCCSP})$. For the cases $\mathcal{O} \in \{B,\ S,\ RS,\ CS\}$ I will show that

$$p \sqsubseteq_{\mathcal{O}} q \;\Rightarrow\; T_{\mathcal{O}} \vdash q = q + p \tag{3}$$

for any $p, q \in \mathsf{T}(\text{BCCSP})$, from which (2) follows immediately. This will be done with structural induction on p and q. So assume $p \sqsubseteq_{\mathcal{O}} q$ and (3) has been proven for all pairs of smaller expressions $p', q' \in \mathsf{T}(\text{BCCSP})$. Provided $T_{\mathcal{O}}$ contains at least the first four axioms of Table 2, one has $T_{\mathcal{O}} \vdash q = q + p$ iff $T_{\mathcal{O}} \vdash q = q + ap'$ for every summand ap' of p.

Take $\mathcal{O} = B$, so $p \sqsubseteq_B q$. Let ap' be a summand of p. Then $p \xrightarrow{a} p'$, so $\exists q' : q \xrightarrow{a} q'$ and $p' =_B q'$. By induction $T_B \vdash p' = p' + q' = q'$, using Proposition 12.1. Furthermore, aq' must be a summand of q, so $T_B \vdash q = q + aq' = q + ap'$ and therefore $T_B \vdash q = q + p$.

Take $\mathcal{O} = S$, so $p \sqsubseteq_S q$. Let ap' be a summand of p. Then $p \xrightarrow{a} p'$, so $\exists q' : q \xrightarrow{a} q'$ and $p' \sqsubseteq_S q'$. By induction $T_S \vdash q' = q' + p'$, so $T_S \vdash aq' = a(q' + p') = a(q' + p') + ap' = aq' + ap'$. Furthermore, aq' must be a summand of q, so $T_S \vdash q = q + aq' = q + ap'$ and thus $T_S \vdash q = q + p$.

Take $\mathcal{O} = RS$, so $p \sqsubseteq_{RS} q$. Let ap' be a summand of p. Then $p \xrightarrow{a} p'$, so $\exists q' : q \xrightarrow{a} q'$ and $p' \sqsubseteq_{RS} q'$. Now $I(p') = I(q')$ and hence $T_{RS} \vdash I(p') = I(q')$. By induction $T_{RS} \vdash q' = q' + p'$, so $T_{RS} \vdash aq' = a(q' + p') = a(q' + p') + ap' = aq' + ap'$. Furthermore, aq' must be a summand of q, so $T_{RS} \vdash q = q + aq' = q + ap'$ and thus $T_{RS} \vdash q = q + p$.

Take $\mathcal{O} = CS$, so $p \sqsubseteq_{CS} q$. Let ap' be a summand of p. Then $p \xrightarrow{a} p'$, so $\exists q' : q \xrightarrow{a} q'$ and $p' \sqsubseteq_{CS} q'$. In case $I(p') = \emptyset$ it must be that $I(q') = \emptyset$ as well, and hence $T_{CS} \vdash p' = q' = 0$. Otherwise, $T_{CS} \vdash p' = bp'' + r$ and by induction $T_{CS} \vdash q' = q' + p'$, so $T_{CS} \vdash aq' = a(q' + p') = a(q' + bp'' + r) = a(q' + bp'' + r) + a(bp'' + r) = a(q' + p') + ap' = aq' + ap'$. Furthermore, aq' must be a summand of q, so in both cases $T_{CS} \vdash q = q + aq' = q + ap'$ and thus $T_{CS} \vdash q = q + p$.

Take $\mathcal{O} = PW$. Suppose $p =_{PW} q$. The axiom $a(bx + by + z) = a(bx + z) + a(by + z)$ allows to rewrite p and q to BCCSP expressions $p' = \Sigma_{i\in I} a_i p_i$ and $q' = \Sigma_{j\in J} a_j q_j$ with p_i and q_j deterministic. For expressions of this form it is easy to establish that $p' =_{PW} q' \Leftrightarrow p' \leftrightarroweq q'$. Using the soundness of the axiom employed, and the completeness of $T_B \subseteq T_{PW}$ for $\leftrightarroweq$, it follows that $T_{PW} \vdash p = p' = q' = q$.

For F and R (as well as B) a proof is given in Bergstra, Klop & Olderog [11] by means of *graph transformations*. A similar proof for RT can be found in Baeten, Bergstra & Klop [6]. This method, applied to semantics $\mathcal{O}$, requires the definition of a class $\mathbb{H}^*$ of finite process graphs that contains at least all finite process trees, and a binary relation $\overset{\mathcal{O}}{\leadsto} \subseteq \mathbb{H}^* \times \mathbb{H}^*$ — a system of *graph transformations*—such that the following can be established:

1. $\overset{\mathcal{O}}{\leadsto}$, used as a rewriting system, is *terminating* on $\mathbb{H}^*$, i.e. any reduction sequence $g_0 \overset{\mathcal{O}}{\leadsto} g_1 \overset{\mathcal{O}}{\leadsto} \cdots$ leads (in finitely many steps) to a *normal form*, a graph that cannot be further transformed,
2. if $g \overset{\mathcal{O}}{\leadsto} h$ then (a) $g =_{\mathcal{O}} h$ and (b) $T_{\mathcal{O}} \vdash \langle\!\langle g \rangle\!\rangle = \langle\!\langle h \rangle\!\rangle$
3. and two normal forms are bisimilar iff they are $\mathcal{O}$-equivalent.

Now the completeness proof goes as follows: Suppose $p =_{\mathcal{O}} q$. As $\textsc{paths}(G(p))$ and $\textsc{paths}(G(q))$ are finite, $U(G(p))$ and $U(G(q))$ belong to $\mathbb{H}^*$, and by requirement 1 they can be rewritten to normal forms g and h. Using Corollary 1.1, Proposition 13.1 and requirement 2(a) above

$$g =_{\mathcal{O}} U(G(p)) \leftrightarroweq G(p) =_{\mathcal{O}} G(q) \leftrightarroweq U(G(q)) =_{\mathcal{O}} h.$$

Thus, with requirement 3, $g \,\underline{\leftrightarrow}\, h$; and Corollaries 17.1 and 17.2 yield

$$p \,\underline{\leftrightarrow}\, \langle\!\langle G(p)\rangle\!\rangle \,\underline{\leftrightarrow}\, \langle\!\langle U(G(p))\rangle\!\rangle, \qquad \langle\!\langle g\rangle\!\rangle \,\underline{\leftrightarrow}\, \langle\!\langle h\rangle\!\rangle, \qquad \langle\!\langle U(G(q))\rangle\!\rangle \,\underline{\leftrightarrow}\, \langle\!\langle G(q)\rangle\!\rangle \,\underline{\leftrightarrow}\, q.$$

Requirement 2(b) and the completeness result for bisimulation semantics proved above finally give

$$T_{\mathcal{O}} \vdash p = \langle\!\langle U(G(p))\rangle\!\rangle = \langle\!\langle g\rangle\!\rangle = \langle\!\langle h\rangle\!\rangle = \langle\!\langle U(G(q))\rangle\!\rangle = q.$$

I will now apply this method to T, RT, CT, R, F and FT. In the cases of T, RT and CT, $\mathbb{H}^*$ is taken to be $\mathbb{H}^{tree}$, the class of finite process trees.

Take $\mathcal{O} = T$. Let $\stackrel{T}{\leadsto}$ be the graph transformation that converts g into h, notation $g \stackrel{T}{\leadsto} h$, iff g is a finite tree with edges (s,a,t) and (s,a,u) with $t \neq u$, and h is obtained by *identifying* t and u. Formally speaking, the nodes of h are those of g, except that t and u are omitted and a fresh node v has been added instead. Often v is taken to be the (equivalence) class $\{t,u\}$. Define the function $'$: $\textsc{nodes}(g) \rightarrow \textsc{nodes}(h)$ by $t' = v$, $u' = v$ and $w' = w$ for $w \neq t, u$. Now $\textsc{edges}(h) = \{(p', a, q') \mid (p,a,q) \in \textsc{edges}(g)\}$ and $\textsc{root}(h) = \textsc{root}(g)'$. This graph transformation is illustrated in Figure 10.

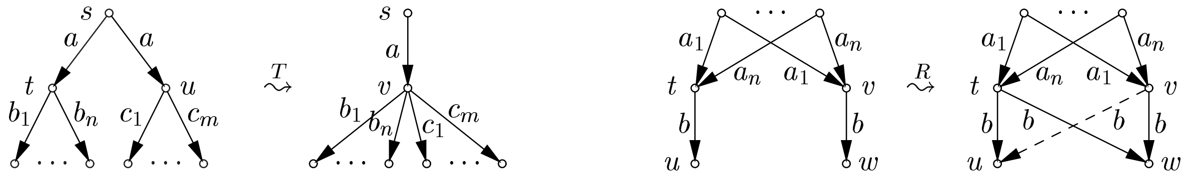


Figure 10: Graph transformations

If g is a finite tree and $g \stackrel{T}{\leadsto} h$ then so is h. Moreover, h has fewer nodes than g. Hence $\stackrel{T}{\leadsto}$ is terminating on $\mathbb{H}^{tree}$. The normal forms are exactly the finite deterministic trees. Now requirement 3 has been established by Theorem 6. Requirement 2(a) is trivial, and for 2(b) observe that any application of $\stackrel{T}{\leadsto}$ corresponds to an application of the axiom $ax + ay \;=\; a(x+y)$.

Take $\mathcal{O} = RT$. Let $\stackrel{RT}{\leadsto}$ be the same graph transformation as $\stackrel{T}{\leadsto}$, except that it only applies if $I(t) = I(u)$. This time the normal forms are the ready trace deterministic trees, and requirement 3 has been established by Proposition 16.3. Again requirement 2(a) is easy to check, and for 2(b) it suffice to observe that any application of $\stackrel{RT}{\leadsto}$ corresponds to an application of the axiom $I(x) \;=\; I(y) \;\;\Rightarrow\;\; ax + ay \;=\; a(x+y)$.

Take $\mathcal{O} = CT$. Let $\stackrel{CT}{\leadsto}$ be the same graph transformation as $\stackrel{T}{\leadsto}$, except that it only applies if $I(t) = \emptyset \Leftrightarrow I(u) = \emptyset$. This time the normal forms are the completed trace deterministic trees, and again requirement 3 has been established by Proposition 16.3. Once more requirement 2(a) is easy to check, and for 2(b) observe that any application of $\stackrel{CT}{\leadsto}$ corresponds to an application of the law $(I(x) = 0 \;\Leftrightarrow\; I(y) = 0) \;\;\Rightarrow\;\; ax + ay \;=\; a(x+y)$. This law falls outside conditional equational logic, but it can be reformulated equationally by considering the two cases $I(x) = 0 = I(y)$ and $I(x) \neq 0 \neq I(y)$. In the first case it must be that $T_B \vdash x = 0 = y$ and hence the law follows from the third and fourth axiom of Table 2. In the second, observe that $I(p) \neq 0$ iff p has the form $bq + r$ with $b \in Act$. Hence the law can be reformulated as $a(bx+u) + a(cy+v) \;=\; a(bx+cy+u+v)$.

A process graph $g \in \mathbb{G}$ is called *history unambiguous* [11] if any two paths from the root to the same node give rise to the same trace, i.e. if for $\pi, \pi' \in \textsc{paths}(g)$ one has $end(\pi) = end(\pi') \Rightarrow T(\pi) = T(\pi')$. The *history* or *trace* $T(s)$ of a node s in such a graph g is defined as $T(\pi)$ for π an arbitrary path from the root of g to s. Observe that trees are history unambiguous. In the next two completeness proofs (the cases R and F) $\mathbb{H}^*$ is taken to be the class $\mathbb{H}^{hu}$ of finite, history unambiguous, connected process graphs. For $g \in \mathbb{H}^{hu}$ and $t, v \in \text{NODES}(g)$ let $t \sim v$ abbreviate

$$\forall s \in \text{NODES}(g),\ a \in Act : (s,a,t) \in \text{EDGES}(g) \Leftrightarrow (s,a,v) \in \text{EDGES}(g).$$

Take $\mathcal{O} = R$. Let $\overset{R}{\leadsto}$ be the graph transformation with $g \overset{R}{\leadsto} h$ iff g has edges (t,b,u) and (v,b,w) with $t \sim v$, and h is obtained by adding a new edge (t,b,w). This graph transformation is illustrated in Figure 10. Note that by applying $\overset{R}{\leadsto}$ twice, one can also add the edge (v,b,u) (indicated with a dashed arrow in Figure 10) if it isn't there already. If g is a finite, history unambiguous process graph and $g \overset{R}{\leadsto} h$ then so is h. Moreover, h has more edges than g. As there is an upperbound to the number of edges of graphs that can be obtained from a given graph $g \in \mathbb{H}^{hu}$ by applying $\overset{R}{\leadsto}$ (namely $n \times l \times n$, where n is the number of nodes in g, and l the number of different edge-labels occurring in g), $\overset{R}{\leadsto}$ is terminating on $\mathbb{H}^{hu}$ (requirement 1). It is easy to see that $\overset{R}{\leadsto}$ does not add new ready pairs. This gives requirement 2(a). For 2(b) observe that an application of $\overset{R}{\leadsto}$ corresponds to a number of applications of $a(bx+u)+a(by+v) \ = \ a(bx+by+u)+a(by+v)$. Finally, requirement 3 follows from Proposition 16.5 and the following

CLAIM: The normal forms w.r.t. $\overset{R}{\leadsto}$ are cross saturated.

PROOF OF THE CLAIM: Let $g \in \mathbb{H}^{hu}$ be a normal form w.r.t. $\overset{R}{\leadsto}$. With induction to the length of $T(u)$ I will show that, for $u, w \in \text{NODES}(g)$,

$$\text{if } T(u) = T(w) \text{ then } u \sim w. \tag{4}$$

This implies that g is cross saturated, for if π, π' and a are as in the remark below Definition 16.3, there must be an edge $(end(\pi), a, u)$ in g. Now $T(u) = T(\pi)a = T(end(\pi'))$, so also $(end(\pi), a, end(\pi')) \in \text{EDGES}(g)$.

Induction base: If $length(T(u)) = 0$, one has $u = w = \text{ROOT}(g)$ and the statement is trivial.

Induction step: Let $T(u) = T(w) \neq \varepsilon$, and let $(t,b,u) \in \text{EDGES}(g)$. By symmetry, it suffices to show that $(t,b,w) \in \text{EDGES}(g)$. As g is connected and history unambiguous, there must be an edge (v,b,w) with $T(t) = T(v)$. By induction $t \sim v$. As g is in normal form it must have an edge (t,b,w).

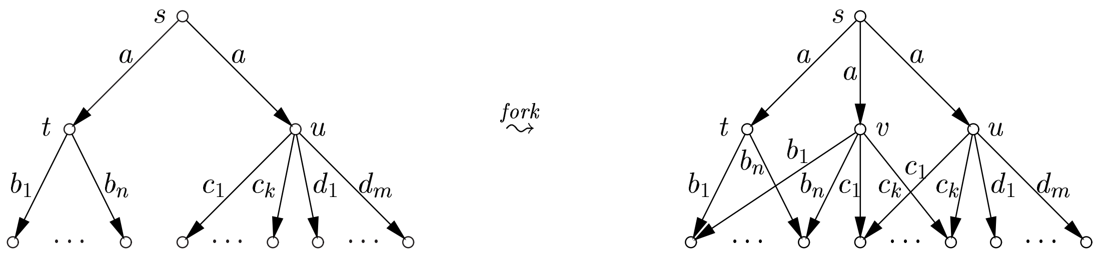


Figure 11: Fork

Take $\mathcal{O} = F$. Let $\overset{fork}{\leadsto}$ be the graph transformation with $g \overset{fork}{\leadsto} h$ iff g has edges (s,a,t) and (s,a,u), $\exists Y \subseteq I(u)$ such that h is given by

- $\text{NODES}(h) = \text{NODES}(g) \dot{\cup} \{v\}$
- $\text{ROOT}(h) = \text{ROOT}(g)$
- $\text{EDGES}(h) = \text{EDGES}(g) \cup \{(s,a,v)\}$
  $\cup\{(v,b,w) \mid (t,b,w) \in \text{EDGES}(g)\} \cup \{(v,b,w) \mid (u,b,w) \in \text{EDGES}(g) \wedge b \in Y\}$

and $|R(h)| > |R(g)|$. This graph transformation is illustrated in Figure 11. Note that for any path $\pi \in \text{PATHS}(h)$ not ending in v, a path $\pi' \in \text{EDGES}(g)$ can be found with $T(\pi') = T(\pi)$ and $end(\pi') = end(\pi)$, namely by circumventing the possible portion through v along t or u. Thus, such paths do not give rise to new ready or failure pairs. For any path $\pi \in \text{PATHS}(h)$ ending in v there is a path $\pi' \in \text{EDGES}(g)$ with $T(\pi') = T(\pi)$ and $end(\pi') = t$. As $I(t) \subseteq I(v)$, also such paths do not give rise to new failure pairs. Hence one has $R(h) = R(g) \cup \{\langle T(t), I(t) \cup Y\rangle\}$ and $F(h) = F(g)$. Note that if $g \in \mathbb{H}^{hu}$ and $g \overset{fork}{\rightsquigarrow} h$, then also $h \in \mathbb{H}^{hu}$. Let $\overset{F}{\rightsquigarrow}$ be $\overset{R}{\rightsquigarrow} \cup \overset{fork}{\rightsquigarrow}$. As $g \overset{fork}{\rightsquigarrow} h \Rightarrow g =_F h$ and $g \overset{R}{\rightsquigarrow} h \Rightarrow g =_R h \Rightarrow g =_F h$, requirement 2(a) is satisfied. For 2(b) observe that an application of $\overset{fork}{\rightsquigarrow}$ corresponds to an application of the axiom $ax + a(y+z) \;=\; ax + a(x+y) + a(y+z)$.

The requirement $|R(h)| > |R(g)|$ says that the transformation may only take place if it actually increases the ready set of the transformed graph. Note that if $g \overset{F}{\rightsquigarrow} h$ then $T(g) = T(h)$. As there is an upperbound to the number of ready pairs of graphs g with a given trace set (namely $|T(g)| \times 2^l$, where l is the number of different edge-labels occurring in g), a reduction sequence $g_0 \overset{F}{\rightsquigarrow} g_1 \overset{F}{\rightsquigarrow} \cdots$ on $\mathbb{H}^{hu}$ can contain only finitely many occurrences of $\overset{fork}{\rightsquigarrow}$. After the last such occurrence it leads in finitely many steps to a normal form, because $\overset{R}{\rightsquigarrow}$ is terminating on $\mathbb{H}^{hu}$. Hence also $\overset{F}{\rightsquigarrow}$ is terminating on $\mathbb{H}^{hu}$ (requirement 1).

Suppose g is a normal form w.r.t. $\overset{F}{\rightsquigarrow}$ and $\langle\sigma, X\rangle \in R(g) \wedge \langle\sigma, Y \cup Z\rangle \in R(g)$. Then g has nodes t and u with $T(t) = T(u) = \sigma$, $I(t) = X$ and $Y \subseteq I(u)$. As g must be a normal form w.r.t. $\overset{R}{\rightsquigarrow}$, it satisfies (4) and hence $t \sim u$. As g is connected, there are edges (s, a, t) and (s, a, u) in g. As g must also be a normal form w.r.t. $\overset{fork}{\rightsquigarrow}$, $\langle\sigma, X \cup Y\rangle \in R(g)$. Thus normal forms w.r.t. $\overset{F}{\rightsquigarrow}$ are saturated as well as cross saturated, and hence requirement 3 follows by Propositions 16.7 and 16.5.

Take $\mathcal{O} = FT$. Let $\overset{sf}{\rightsquigarrow}$ (*symmetric fork*) be the graph transformation consisting of those instances of $\overset{fork}{\rightsquigarrow}$ where $Y = I(u)$, but with the requirement $|R(h)| > |R(g)|$ relaxed to $|RT_N(h)| > |RT_N(g)|$. Let $\mathbb{H}^*$ be $\mathbb{H}^{tree}$, and define $\overset{sfu}{\rightsquigarrow}$ by $g \overset{sfu}{\rightsquigarrow} h$ if $g \overset{sf}{\rightsquigarrow} h'$ and $h = U(h')$. Thus $\overset{sfu}{\rightsquigarrow}$ is the variant of $\overset{sf}{\rightsquigarrow}$ in which the target is unfolded into a tree. Let $\overset{FT}{\rightsquigarrow}$ be $\overset{RT}{\rightsquigarrow} \cup \overset{sfu}{\rightsquigarrow}$. As there is an upperbound to the number of normal ready traces of graphs with a given finite trace set, $\overset{FT}{\rightsquigarrow}$ is terminating on $\mathbb{H}^*$ (requirement 1). The normal forms are exactly the finite RT-saturated ready trace deterministic process trees, so requirement 3 follows from Propositions 16.3 and 16.8. It follows immediately from Corollary 5.1 that $g \overset{sf}{\rightsquigarrow} h \Rightarrow g =_{FT} h$. Hence Proposition 13.1 gives $g \overset{sfu}{\rightsquigarrow} h \Rightarrow g =_{FT} h$. Moreover, $g \overset{RT}{\rightsquigarrow} h \Rightarrow g =_{RT} h \Rightarrow g =_{FT} h$, which yields requirement 2(a). For 2(b) observe that an application of $\overset{sf}{\rightsquigarrow}$ corresponds to an application of the axiom $ax + ay \;=\; ax + ay + a(x+y)$, and as $h \leftrightarroweq U(h)$ Corollary 17.2 gives $T_B \vdash \langle\!\langle h \rangle\!\rangle = \langle\!\langle U(h) \rangle\!\rangle$ for $h \in \mathbb{H}$. □

In Theorem 8 the fifth and seventh axioms of Table 2 may be replayed by

$$a\sum_{i=1}^{n}(b_i x_i + b_i y_i) = a\sum_{i=1}^{n}(b_i x_i + b_i y_i) + a\sum_{i=1}^{n} b_i y_i \quad \text{and} \quad a\sum_{i=1}^{n} b_i x_i + a\sum_{i=1}^{n} b_i y_i = a\sum_{i=1}^{n}(b_i x_i + b_i y_i).$$

These laws derive the same closed substitution instances. Thus none of the axiomatizations require the operator I, or conditional equations. However, the laws above are axioms schemes which have instances for any choice of $n \in \mathbb{N}$. Even if *Act* is finite, the axiomatizations involving these laws are infinite.

Theorem 9 Suppose *Act* is infinite. For each of the semantics $\mathcal{O} \in \{T,\ S,\ CT,\ F,\ R,\ FT,\ RT,\ RS,\ B,\ U\}$ two BCCSP expressions with variables are $\mathcal{O}$-equivalent iff they can be proved equal from the axioms marked with '+' or 'ω' in the column for $\mathcal{O}$ in Table 2. It follows that the axioms marked with 'v' are derivable.

Proof: For $\mathcal{O} \in \{T,\ CT,\ F,\ R,\ FT,\ RT,\ B\}$ this has been established in GROOTE [23]. His proof for F, R, FT and RT can be applied to S and RS as well. The proof for U is rather trivial, but omitted here. $\square$

Groote also showed that if *Act* is finite, Theorem 9 does not hold for F, R, FT and RT. But for B and CT it suffices to assume that *Act* is nonempty, and for T it suffices to assume that *Act* has at least two elements. I do not know which cardinality restriction on *Act* is needed in the cases of S and RS. A complete axiomatization for open terms for completed simulation or possible worlds semantics has so far not been provided.

17.3 Axiomatizing the preorders

In Table 3, complete axiomatizations can be found for the eleven preorders corresponding to the equivalences axiomatized in Table 2 (there is no preorder for tree semantics (U)). This time provability is defined according to the standards of either first-order logic with inequality or conditional inequational logic, i.e. it may be used that $\sqsubseteq$ is reflexive and transitive and satisfies the precongruence properties of Theorem 7b. For any semantics $\mathcal{O}$ the $\mathcal{O}$-preorder and $\mathcal{O}$-equivalence are related by $p =_{\mathcal{O}} q \Leftrightarrow p \sqsubseteq_{\mathcal{O}} q \wedge q \sqsubseteq_{\mathcal{O}} p$. Thus either $p = q$ is taken to be an abbreviation of $p \sqsubseteq q \wedge q \sqsubseteq p$ or the conditional axioms $p = q \Rightarrow p \sqsubseteq q$ and $p \sqsubseteq q \wedge q \sqsubseteq p \Rightarrow p = q$ are considered part of the axiomatizations. In the latter case, the axioms of Table 3 also constitute complete axiomatizations of the equivalences.

The three axioms in Table 3 in which the inequality is written "$\sqsubseteq$" represent strengthenings of the corresponding axioms in Table 2. The axioms in which the inequality is written "$\sqsupseteq$" are merely slick reformulations of the corresponding axioms in Table 2, and could be replaced by them. Unlike in Table 2, the characteristic axiom for the readiness preorder (the ninth) is now a substitution instance of the characteristic axiom for the failures preorder (the tenth).

Note that the characteristic axiom for the ready simulation preorder (the fifth) derives all closed instances of $I(x) = I(y) \Rightarrow ax \sqsubseteq a(x + y)$, which gives the fifth axiom of Table 2. Hence all closed instances of the characteristic axiom for the ready trace preorder (the seventh) are derivable from the fifth and eighth axioms. It follows that conditional (in)equations, involving the operator I, or unbounded sums, are no longer needed in the axiomatizations of ready simulation and failure trace semantics.

Theorem 10 For each of the semantics $\mathcal{O} \in \{T,\ S,\ CT,\ CS,\ F,\ R,\ FT,\ RT,\ PW,\ RS,\ B\}$ one has $p \sqsubseteq_{\mathcal{O}} q$ for $p, q \in \mathsf{T}(\text{BCCSP})$ iff $p \sqsubseteq q$ can be proved from the axioms marked with "+" in the column for $\mathcal{O}$ in Table 3. The axioms marked with "v" are valid in $\mathcal{O}$-semantics but not needed for the proof.

Proof: "If" (*soundness*): In the light of Theorem 7b it suffices to show that the closed instances of the indicated axioms are valid in the corresponding semantics. This is straightforward.

"Only if" (*completeness*): Let $T^*_{\mathcal{O}}$ be the set of axioms marked with "+" in the column for $\mathcal{O}$. Write $T^*_{\mathcal{O}} \vdash p \sqsubseteq q$ if the inequation $p \sqsubseteq q$ is provable from $T^*_{\mathcal{O}}$. I have to show that

$$p \sqsubseteq_{\mathcal{O}} q \;\Rightarrow\; T^*_{\mathcal{O}} \vdash p \sqsubseteq q \tag{5}$$

	B	RS	PW	RT	FT	R	F	CS	CT	S	T
$(x+y)+z \;=\; x+(y+z)$	+	+	+	+	+	+	+	+	+	+	+
$x+y \;=\; y+x$	+	+	+	+	+	+	+	+	+	+	+
$x+0 \;=\; x$	+	+	+	+	+	+	+	+	+	+	+
$x+x \;=\; x$	+	+	+	+	+	+	+	+	+	+	+
$ax \;\sqsubseteq\; ax+ay$		+	+	+	+	+	+	v	v	v	v
$a(bx+by+z) \;=\; a(bx+z)+a(by+z)$			+	v	v	v	v		v		v
$I(x) \;=\; I(y) \;\Rightarrow\; ax+ay \;=\; a(x+y)$				+	v	v	v		v		v
$ax+ay \;\sqsupseteq\; a(x+y)$					+		v		v		v
$a(bx+u)+a(by+v) \;\sqsupseteq\; a(bx+by+u)$						+	v		v		v
$ax+a(y+z) \;\sqsupseteq\; a(x+y)$							+		v		v
$ax \;\sqsubseteq\; ax+y$								+	+	v	v
$a(bx+u)+a(cy+v) \;=\; a(bx+cy+u+v)$									+		v
$x \;\sqsubseteq\; x+y$										+	+
$ax+ay \;=\; a(x+y)$											+
$I(0) \;=\; 0$	+	+	+	+	+	+	+	+	+	+	+
$I(ax) \;=\; a0$	+	+	+	+	+	+	+	+	+	+	+
$I(x+y) \;=\; I(x)+I(y)$	+	+	+	+	+	+	+	+	+	+	+

Table 3: Complete axiomatizations for the preorders

for any $p, q \in \mathsf{T}(\text{BCCSP})$. The case $\mathcal{O} = B$ follows from Proposition 12.1 and Theorem 8. For the cases $\mathcal{O} \in \{S,\ CS,\ RS\}$ (5) will be established with structural induction on p and q. So assume $p \sqsubseteq_{\mathcal{O}} q$ and (5) has been proven for all pairs of smaller expressions $p', q' \in \mathsf{T}(\text{BCCSP})$.

Take $\mathcal{O} = S$, so $p \sqsubseteq_S q$. Using the axiom $x \sqsubseteq x + y$ one finds that $T_S^* \vdash p \sqsubseteq q$ if for every summand ap' of p there is a summand aq' of q such that $T_S^* \vdash ap' \sqsubseteq aq'$. So let ap' be a summand of p. Then $p \xrightarrow{a} p'$, so $\exists q' : q \xrightarrow{a} q'$ and $p' \sqsubseteq_S q'$. Note that aq' is a summand of q. By induction $T_S^* \vdash p' \sqsubseteq q'$, so $T_S^* \vdash ap' \sqsubseteq aq'$.

Take $\mathcal{O} = CS$, so $p \sqsubseteq_{CS} q$. Using the axiom $ax \sqsubseteq ax + y$ one finds that $T_{CS}^* \vdash p \sqsubseteq q$ if $I(p) \neq \emptyset$ and for every summand ap' of p there is a summand aq' of q such that $T_{CS}^* \vdash ap' \sqsubseteq aq'$. In case $I(p) = \emptyset$ it must be that $I(q) = \emptyset$ as well, and hence $T_{CS}^* \vdash p = q = 0$. Otherwise, let ap' be a summand of p. Then $p \xrightarrow{a} p'$, so $\exists q' : q \xrightarrow{a} q'$ and $p' \sqsubseteq_{CS} q'$. Note that aq' is a summand of q. By induction $T_{CS}^* \vdash p' \sqsubseteq q'$, so $T_{CS}^* \vdash ap' \sqsubseteq aq'$.

Take $\mathcal{O} = RS$, so $p \sqsubseteq_{RS} q$. Using the first five axioms of Table 3 one finds that $T_{RS}^* \vdash p \sqsubseteq q$ if $I(p) = I(q)$ and for every summand ap' of p there is a summand aq' of q such that $T_{RS}^* \vdash ap' \sqsubseteq aq'$. As $p \sqsubseteq_{RS} q$ one has $I(p) = I(q)$. Let ap' be a summand of p. Then $p \xrightarrow{a} p'$, so $\exists q' : q \xrightarrow{a} q'$ and $p' \sqsubseteq_{RS} q'$. Note that aq' is a summand of q. By induction $T_{RS}^* \vdash p' \sqsubseteq q'$, so $T_{RS}^* \vdash ap' \sqsubseteq aq'$.

Take $\mathcal{O} = PW$. Suppose $p \sqsubseteq_{PW} q$. The axiom $a(bx + by + z) = a(bx + z) + a(by + z)$ allows to rewrite p and q to BCCSP expressions $p' = \Sigma_{i \in I} a_i p_i$ and $q' = \Sigma_{j \in J} a_j q_j$ with p_i and q_j deterministic. For expressions of this form it is easy to establish that $p' \sqsubseteq_{PW} q' \Leftrightarrow p' \sqsubseteq_{RS} q'$. Using the soundness of the axiom employed, and the completeness of $T_{RS} \subseteq T_{PW}$ for $\sqsubseteq_{RS}$, it follows that $T_{PW} \vdash p = p' \sqsubseteq q' = q$.

The remaining completeness proofs go by a variant of the method of graph transformations, where requirement 3 is replaced by

$$\text{if } g \text{ and } h \text{ are normal forms, then } g \sqsubseteq_{\mathcal{O}} h \Leftrightarrow g \sqsubseteq_{\mathcal{N}} h.$$

Here $\mathcal{N}$ should be a semantics finer than $\mathcal{O}$, for which the completeness theorem has already been established, and for which $T^*_{\mathcal{N}} \subseteq T^*_{\mathcal{O}}$. The reasoning now goes exactly as in the proof of Theorem 8: Suppose $p \sqsubseteq_{\mathcal{O}} q$. Rewrite $U(G(p))$ and $U(G(q))$ to normal forms g and h. Then

$$g =_{\mathcal{O}} U(G(p)) \leftrightarroweq G(p) \sqsubseteq_{\mathcal{O}} G(q) \leftrightarroweq U(G(q)) =_{\mathcal{O}} h.$$

Thus, with requirement 3, $g \sqsubseteq_{\mathcal{N}} h$. Corollary 17.2 yields $\langle\!\langle g \rangle\!\rangle \sqsubseteq_{\mathcal{N}} \langle\!\langle h \rangle\!\rangle$, and one obtains

$$T^*_{\mathcal{O}} \vdash p = \langle\!\langle U(G(p)) \rangle\!\rangle = \langle\!\langle g \rangle\!\rangle \sqsubseteq \langle\!\langle h \rangle\!\rangle = \langle\!\langle U(G(q)) \rangle\!\rangle = q.$$

For each of the six remaining completeness proofs, the class $\mathbb{H}^*$ and the graph transformations are the same as in the proof of Theorem 8. Thus requirements 1 and 2(a) are fulfilled. As (the closed instances of) the axioms for the respective equivalences from Table 2 are easily derivable from the ones for the corresponding preorders from Table 3, requirement 2(b) is fulfilled as well. Requirement 3, which used to follow from Theorem 6 and Propositions 16.3, 16.5, 16.7 and 16.8, now follows from Propositions 16.4, 16.6, 16.7 and 16.8. □

17.4 A language for finite, concrete, sequential processes with internal choice

Let BCSP be the language that extends BCCSP with a binary operator $\oplus$, modelling *internal choice*. Like $p+q$, the expression $p \oplus q$ represents a process, first being involved in a choice between its summands p and q, and then proceeding as the chosen process. However, whereas $+$ represents a choice that can be influenced by the environment of the process (an *external choice*), $\oplus$ represents one that is due to internal nondeterminism of the specified system. BCSP can be regarded as a basic fragment of the language CSP of Hoare [31].

The set T(BCSP) of (closed) terms over BCSP, or (closed) *BCSP-expressions*, and its subset T_1(BCSP) of *initially deterministic BCSP-expressions*, are defined by:

- $0 \in \mathsf{T}_1(\text{BCSP}) \subseteq \mathsf{T}(\text{BCSP})$,
- $aP \in \mathsf{T}_1(\text{BCSP})$ for any $a \in Act$ and $P \in \mathsf{T}(\text{BCSP})$,
- $p + q \in \mathsf{T}_1(\text{BCSP})$ for any $p, q \in \mathsf{T}_1(\text{BCSP})$,
- $P + Q \in \mathsf{T}(\text{BCSP})$ for any $P, Q \in \mathsf{T}(\text{BCSP})$,
- $P \oplus Q \in \mathsf{T}(\text{BCSP})$ for any $P, Q \in \mathsf{T}(\text{BCSP})$.

Again, subterms $a0$ may be abbreviated by a. Brackets are used for disambiguation only, assuming associativity of $+$ and $\oplus$, and letting a bind stronger than $+$ and $\oplus$. Semantically, BCSP-expressions represent nonempty, finite sets of initially deterministic BCSP expressions: for $P, Q \in \mathsf{T}(\text{BCSP})$ let

$$[\![0]\!] := \{0\} \qquad [\![aP]\!] := \{aP\} \qquad [\![P+Q]\!] := \{p+q \mid p \in [\![P]\!],\ q \in [\![Q]\!]\} \qquad [\![P \oplus Q]\!] := [\![P]\!] \cup [\![Q]\!].$$

On T_1(BCSP) action relations $\xrightarrow{a}$ for $a \in Act$ are defined as the predicates on T_1(BCSP) generated by the action rules of Table 4. Here a ranges over Act, P over T(BCSP) and p and q over T_1(BCSP). This makes T_1(BCSP) into a labelled transition system. Hence, in the light of Section 1.5 all semantic equivalences of Sections 2–12 and 14 are well-defined on T(BCSP), and for each of the semantics it is determined when two BCSP-expressions denote the same process.

$\dfrac{p \in [\![P]\!]}{aP \xrightarrow{a} p}$	$\dfrac{p \xrightarrow{a} p'}{p+q \xrightarrow{a} p'}$	$\dfrac{q \xrightarrow{a} q'}{p+q \xrightarrow{a} q'}$

Table 4: Action rules for BCSP

The following theorem says that all these semantic equivalences are congruences for BCSP. Even stronger, all the preorders of this paper are precongruences for BCCSP.

Theorem 11 Let $P, Q, R, S \in \mathsf{T}(\text{BCSP})$ and let $\mathcal{O}$ be any of the semantics of Sections 2–12, 14. Then

$$\begin{array}{lcl} P =_{\mathcal{O}} Q \;\wedge\; R =_{\mathcal{O}} S & \Rightarrow & aP =_{\mathcal{O}} aQ \;\wedge\; P+R =_{\mathcal{O}} Q+S \;\wedge\; P \oplus R =_{\mathcal{O}} Q \oplus S, \\ P \sqsubseteq_{\mathcal{O}} Q \;\wedge\; R \sqsubseteq_{\mathcal{O}} S & \Rightarrow & aP \sqsubseteq_{\mathcal{O}} aQ \;\wedge\; P+R \sqsubseteq_{\mathcal{O}} Q+S \;\wedge\; P \oplus R \sqsubseteq_{\mathcal{O}} Q \oplus S. \end{array}$$

Proof: Each of the preorders $\mathcal{O}$ has a modal characterization, given by $P \sqsubseteq_{\mathcal{O}} Q \Leftrightarrow \mathcal{O}(P) \subseteq \mathcal{O}(Q)$, where $\mathcal{O}(P) = \bigcup_{p \in [\![P]\!]} \mathcal{O}(p)$ for $P \in \mathsf{T}(\text{BCSP})$ and $\mathcal{O}(p) = \{\varphi \in \mathcal{L}_{\mathcal{O}} \mid p \models \varphi\}$ for $p \in \mathsf{T}_1(\text{BCSP})$. Now $\mathcal{O}(P \oplus Q) = \mathcal{O}(P) \cup \mathcal{O}(Q)$. This immediately yields the compositionality of $\mathcal{O}$ w.r.t. $\oplus$: $P \sqsubseteq_{\mathcal{O}} Q \wedge R \sqsubseteq_{\mathcal{O}} S \Rightarrow P \oplus R \sqsubseteq_{\mathcal{O}} Q \oplus S$, and hence $P =_{\mathcal{O}} Q \wedge R =_{\mathcal{O}} S \Rightarrow P \oplus R =_{\mathcal{O}} Q \oplus S$.

Note that every formula in the infinitary Hennessy-Milner logic is logically equivalent to a disjunction of formulas of the form $\bigwedge_{i \in I} a_i\varphi_i \wedge \bigwedge_{j \in J} \neg a_j\varphi_j$. Let $\mathcal{O}'(P)$ be the class of formulas in $\mathcal{O}(P)$ of that form. It follows that $P \sqsubseteq_{\mathcal{O}} Q \Leftrightarrow \mathcal{O}'(P) \subseteq \mathcal{O}'(Q)$ for $P, Q \in \mathsf{T}(\text{BCSP})$.

For $p, q \in \mathsf{T}_1(\text{BCSP})$ one has $p + q \models \bigwedge_{i \in I} a_i\varphi_i \wedge \bigwedge_{j \in J} \neg a_j\varphi_j$ iff I can be written as $I_1 \cup I_2$ such that $p \models \bigwedge_{i \in I_1} a_i\varphi_i \wedge \bigwedge_{j \in J} \neg a_j\varphi_j$ and $q \models \bigwedge_{i \in I_2} a_i\varphi_i \wedge \bigwedge_{j \in J} \neg a_j\varphi_j$. Moreover, for each semantics $\mathcal{O}$ of this paper, if $\bigwedge_{i \in I} a_i\varphi_i \wedge \bigwedge_{j \in J} \neg a_j\varphi_j \in \mathcal{L}_{\mathcal{O}}$ and $I' \subset I$, then $\bigwedge_{i \in I'} a_i\varphi_i \wedge \bigwedge_{j \in J} \neg a_j\varphi_j \in \mathcal{L}_{\mathcal{O}}$[12]. Thus, for $P, Q \in \mathsf{T}(\text{BCSP})$ and $\bigwedge_{i \in I} a_i\varphi_i \wedge \bigwedge_{j \in J} \neg a_j\varphi_j \in \mathcal{L}_{\mathcal{O}}$, one has $\bigwedge_{i \in I} a_i\varphi_i \wedge \bigwedge_{j \in J} \neg a_j\varphi_j \in \mathcal{O}'(P+Q)$ iff $I = I_1 \cup I_2$ such that $\bigwedge_{i \in I_1} a_i\varphi_i \wedge \bigwedge_{j \in J} \neg a_j\varphi_j \in \mathcal{O}'(P)$ and $\bigwedge_{i \in I_2} a_i\varphi_i \wedge \bigwedge_{j \in J} \neg a_j\varphi_j \in \mathcal{O}'(Q)$. This immediately yields the compositionality of $\mathcal{O}$ w.r.t. $+$.

The compositionality of $\mathcal{O}$ w.r.t. a is straightforward. □

If $P \in \mathsf{T}(\text{BCSP})$, then $G([\![P]\!])$ is a finite process graph with multiple roots. Vice versa, any finite process graph with multiple roots $g \in \mathbb{G}^{mr}$ can be represented by a BCSP-expression $\langle\!\langle g \rangle\!\rangle \in \mathsf{T}(\text{BCSP})$, such that $G(\langle\!\langle g \rangle\!\rangle) \leftrightarrows g$. Just extend Definition 17.1 by $\langle\!\langle g \rangle\!\rangle = \bigoplus_{r \in \text{ROOTS}(g)} \langle\!\langle g_r \rangle\!\rangle$.

Axioms In Table 5, complete axiomatizations in terms of BCSP can be found for the same eleven semantics axiomatized in terms of BCCSP in Tables 2 and 3. The first two sections of the table apply to the equivalences and the first and last section to the preorders. These axioms are mild variations of the ones in Tables 2 and 3, and have been found by exploiting a close correspondence in semantic validity between BCSP and BCCSP expressions. First of all, using the definitions just given, the soundness of the axioms in the first section of Table 5 is easily established. Using these, any closed BCSP expression can be rewritten in the form $\bigoplus_{i=1}^{n} p_i$ with p_i closed BCCSP expressions. Now the following lemma reduces the validity of (in)equations over BCSP to that of (in)equations over BCCSP.

Lemma 17.1 $\displaystyle\bigoplus_{i=1}^{n} p_i \sqsubseteq_{\mathcal{O}} \bigoplus_{j=1}^{m} q_j \;\Leftrightarrow\; \sum_{i=1}^{n} ap_i \sqsubseteq_{\mathcal{O}} \sum_{j=1}^{m} aq_j$ for $p_i, q_j \in \mathsf{T}(\text{BCCSP})$.

Proof: $\varphi \in \mathcal{O}(\bigoplus_{i=1}^{n} p_i) \Leftrightarrow a\varphi \in \mathcal{O}(\sum_{i=1}^{n} ap_i)$. □

[12]At least when replacing the modality X of R, RT, PW and RS by $\bigwedge_{a \in Y} \neg a\top \wedge \bigwedge_{a \in Z} a\top$.

	B	RS	PW	RT	FT	R	F	CS	CT	S	T
$(x \oplus y) \oplus z \ = \ x \oplus (y \oplus z)$	+	+	+	+	+	+	+	+	+	+	+
$x \oplus y \ = \ y \oplus x$	+	+	+	+	+	+	+	+	+	+	+
$x \oplus x \ = \ x$	+	+	+	+	+	+	+	+	+	+	+
$(x + y) + z \ = \ x + (y + z)$	+	+	+	+	+	+	+	+	+	+	+
$x + y \ = \ y + x$	+	+	+	+	+	+	+	+	+	+	+
$x + 0 \ = \ x$	+	+	+	+	+	+	+	+	+	+	+
$(x \oplus y) + z \ = \ (x + z) \oplus (y + z)$	+	+	+	+	+	+	+	+	+	+	+
$a(x \oplus y) \ = \ ax + ay$	+	+	+	+	+	+	+	+	+	+	+
$\sum_{i=1}^n (b_i x_i + b_i y_i) \ = \ \sum_{i=1}^n (b_i x_i + b_i y_i) \oplus \sum_{i=1}^n b_i y_i$		+	v	v	v	v	v	v	v	v	v
$bx + by + z \ = \ (bx + z) \oplus (by + z)$			+	v	v	v	v		v		v
$\sum_{i=1}^n b_i x_i \oplus \sum_{i=1}^n b_i y_i \ = \ \sum_{i=1}^n (b_i x_i + b_i y_i)$				+	+	v	v		v		v
$x + x \ = \ x$					+		v		v		v
$(bx + u) \oplus (by + v) \ = \ (bx + by + u) \oplus (by + v)$						+	+		v		v
$x \oplus (y + z) \ = \ x \oplus (x + y) \oplus (y + z)$							+		v		v
$x + by + z \ = \ (x + by + z) \oplus (by + z)$								+	v	v	v
$(bx + u) \oplus (cy + v) \ = \ bx + cy + u + v$									+		v
$x + y \ = \ (x + y) \oplus y$										+	v
$x \oplus y \ = \ x + y$											+
$x \ \sqsubseteq \ x \oplus y$		+	+	+	+	+	+	+	+	+	+
$bx + by + z \ = \ (bx + z) \oplus (by + z)$			+	v	v	v	v		v		v
$\sum_{i=1}^n b_i x_i \oplus \sum_{i=1}^n b_i y_i \ = \ \sum_{i=1}^n (b_i x_i + b_i y_i)$				+	v	v	v		v		v
$x + x \ = \ x$					+		v		v		v
$(bx + u) \oplus (by + v) \ \sqsupseteq \ bx + by + u$						+	v		v		v
$x \oplus (y + z) \ \sqsupseteq \ x + y$							+		v		v
$ax \ \sqsubseteq \ ax + y$								+	v	v	v
$(bx + u) \oplus (cy + v) \ = \ bx + cy + u + v$									+		v
$x \ \sqsubseteq \ x + y$										+	v
$x \oplus y \ = \ x + y$											+

Table 5: Complete axiomatizations in terms of BCSP

Most of the axioms in the last two sections of Table 5 can be recognized as restatements of the axioms of Tables 2 and 3, using the insight of Lemma 17.1. However, in BCSP it is not so clear how the set of initial actions of a process should be defined, and the obvious adaptations of the axioms involving the operator I would not be sound. Therefore the alternatives to those axioms discussed near the end of Section 17.2 are used. Moreover, in BCSP the axiom $x + x = x$ is not sound for readiness semantics. Substituting $a \oplus b$ for x, one derives $a \oplus (a + b) \oplus b = a \oplus b$, of which only the left-hand side has a ready pair $\langle \varepsilon, \{a, b\} \rangle$. However, in the setting of BCCSP all closed instances of $x + x = x$ are derivable from the law $ax + ax = ax$, which corresponds with the BCSP axiom $x \oplus x = x$. Following Lemma 17.1, the characteristic axiom for failure trace equivalence should be $x \oplus y = x \oplus y \oplus (x + y)$. This axiom is derivable from $x + x = x$, and all closed instances of $x + x = x$ are derivable from $x \oplus y = x \oplus y \oplus (x + y)$ and the axioms in the first section of Table 5.

Let $U_{\mathcal{O}}$ be the set of axioms marked with "+" in the column for $\mathcal{O}$ in the first two sections of Table 5, and $U^*_{\mathcal{O}}$ be the set of axioms marked with "+" in the column for $\mathcal{O}$ in the first and last section of Table 5. Write $S \vdash \Phi$ if the formula Φ is provable from the set of axioms S.

Theorem 12 For $\mathcal{O} \in \{T,\ S,\ CT,\ CS,\ F,\ R,\ FT,\ RT,\ PW,\ RS,\ B\}$ and $P, Q \in \mathsf{T}(\text{BCSP})$ one has $P =_{\mathcal{O}} Q \Leftrightarrow U_{\mathcal{O}} \vdash P = Q$ and $P \sqsubseteq_{\mathcal{O}} Q \Leftrightarrow U^*_{\mathcal{O}} \vdash P \sqsubseteq Q$.

Proof: "$\Leftarrow$" (*soundness*): In the light of Theorem 11 it suffices to show that the closed instances of the indicated axioms are valid in the corresponding semantics. In fact, one may restrict attention to the instances where expressions $\bigoplus_{i=1}^n p_i$ with p_i closed BCCSP expressions are substituted for the variables. It is not difficult to check, for each of these axioms, that such instances of it are derivable from the instances of it where simple closed BCCSP expressions are substituted for the variables (but taking $x \oplus y = x \oplus y \oplus (x+y)$ instead of $x + x = x$ to be the characteristic axiom for failure trace semantics). That the instances of the latter kind are valid in the corresponding semantics follows immediately from Lemma 17.1 and the soundness of the axioms for BCCSP.

"First $\Rightarrow$" (*completeness of the axioms for the equivalences*): Let $T'_{\mathcal{O}}$ be the set of axioms marked with "+" in the column for $\mathcal{O}$ in Table 2, but using $a\sum_{i=1}^n b_i x_i + a\sum_{i=1}^n b_i y_i = a\sum_{i=1}^n (b_i x_i + b_i y_i)$ and $a\sum_{i=1}^n (b_i x_i + b_i y_i) = a\sum_{i=1}^n (b_i x_i + b_i y_i) + a\sum_{i=1}^n b_i y_i$ instead of the axioms involving the operator I. As Theorem 8 establishes completeness for closed terms only, it holds for $T'_{\mathcal{O}}$ as well.

Claim: If $T'_{\mathcal{O}} \vdash p = \sum_{j=1}^m a q_j$ for $p, q_j \in \mathsf{T}(\text{BCCSP})$, then, modulo applications of the first three axioms of Table 2, p has the form $p = \sum_{i=1}^n a p_i$.

Proof of the claim: As all axioms in $T'_{\mathcal{O}}$ are equations, I may use induction on the proof of $p = \sum_{j=1}^m a q_j$ in equational logic. The case that $p = \sum_{j=1}^m a q_j$ is a closed instance of an axiom of $T'_{\mathcal{O}}$ proceeds by inspection of those axioms. The cases of placing an equation in a context, as well as reflexivity, symmetry and transitivity, are trivial.

Claim: $T'_{\mathcal{O}} \vdash \displaystyle\sum_{i=1}^n a p_i = \sum_{j=1}^m a q_j \;\Rightarrow\; U_{\mathcal{O}} \vdash \bigoplus_{i=1}^n p_i = \bigoplus_{j=1}^m q_j$ for any $p_i, q_j \in \mathsf{T}(\text{BCCSP})$.

Proof of the claim: I use induction on the proof of $\sum_{i=1}^n a p_i = \sum_{j=1}^m a q_j$ from $T'_{\mathcal{O}}$ in equational logic. The case that $\sum_{i=1}^n a p_i = \sum_{j=1}^m a q_j$ is a closed instance of an axiom of $T'_{\mathcal{O}}$ proceeds by inspection of those axioms, taking into account the remark about $x \oplus y = x \oplus y \oplus (x+y)$ right before this theorem. The case of a closed instance of an axiom of $T'_{\mathcal{O}}$ in a context is straightforward, also using that all closed instances of axioms of $T'_{\mathcal{O}}$ are derivable from the ones of $U_{\mathcal{O}}$, taking into account the remark about $x + x = x$ right before this theorem. The cases of reflexivity and symmetry are trivial. Transitivity follows from the previous claim.

Completeness proof: Suppose $P =_{\mathcal{O}} Q$ for certain $P, Q \in \mathsf{T}(\text{BCSP})$. Using the axioms in the first section of Table 5 one obtains $U_{\mathcal{O}} \vdash P = \bigoplus_{i=1}^n p_i$ and $U_{\mathcal{O}} \vdash Q = \bigoplus_{j=1}^m q_j$ with $p_i, q_j \in \mathsf{T}(\text{BCCSP})$. By the soundness of these axioms one has $\bigoplus_{i=1}^n p_i =_{\mathcal{O}} \bigoplus_{j=1}^m q_j$. Therefore $\sum_{i=1}^n a p_i =_{\mathcal{O}} a\bigoplus_{i=1}^n p_i =_{\mathcal{O}} a\bigoplus_{j=1}^m q_j =_{\mathcal{O}} \sum_{j=1}^m a q_j$ by the soundness of $a(x \oplus y) = ax + ay$ and Theorem 11, and hence $T'_{\mathcal{O}} \vdash \sum_{i=1}^n a p_i = \sum_{j=1}^m a q_j$ by the completeness of $T'_{\mathcal{O}}$. Now $U_{\mathcal{O}} \vdash P = Q$ follows by the claim above.

The second "$\Rightarrow$" (*completeness of the axioms for the preorders*) goes likewise, except that in the proof of the second claim, in order to handle the axioms $ax \sqsubseteq ax + y$ and $x \sqsubseteq x + y$, one uses the axiom $x \sqsubseteq x \oplus y$ of $U^*_{\mathcal{O}}$. Furthermore, $ax \sqsubseteq ax + y$ is derivable from U^*_{CT}, and $x \sqsubseteq x + y$ from U^*_T. □

18 Criteria for selecting a semantics for particular applications

Must testing Assume the testing scenario of trace semantics: we are unable to influence the behaviour of an investigated system in any way and can observe the performed actions only. Not even deadlock is observable. In this case there appears to be no reason to distinguish the two processes of Counterexample 3, $ab + a(b + c)$ and $a(b + c)$. They have the same traces, and consequently allow the same observations. Likewise, one might see no reason to distinguish between the two processes of Counterexample 2, $ab + a$ and ab; also these have the same traces. However, when buying process ab, it may come with the guarantee that, in every run of the system, sooner or later it will perform the action b, at least if the action a is known to terminate. Such a guarantee cannot be given for $ab+a$. The distinction between ab and $ab+a$ alluded to here can be formalized with the concept of *must testing*, originally due to De Nicola & Hennessy [17]: ab *must* do a b, whereas $ab + a$ must not.

For finite processes, must testing could be formalized as follows. For $t \subseteq Act^*$ we say that a finite process $p \in \mathbb{P}$ *must* pass the test t if $CT(p) \subseteq t$. To test whether a process will sooner or later perform a b-action take t to be all sequences of actions containing a b. To test whether a process will always perform a b immediately after it does an a, take t to be all traces in which any a is immediately followed by a b. Now write $p \sqsubseteq_T^{\text{must}} q$ if for all tests $t \subseteq Act^*$ such that p must pass t, q must pass t as well. It is easy to see that, for finite processes p and q, $p \sqsubseteq_T^{\text{must}} q$ iff $q \sqsubseteq_{CT} p$.

All testing scenarios $\mathcal{O}$ sketched earlier in this paper can be regarded as forms of *may testing*: it is recorded whether an observation $\varphi \in \mathcal{L}_\mathcal{O}$ *may* be made for a process p, and one writes $p \sqsubseteq_\mathcal{O} q$ if any observation that may be made for p, may also be made for q.

In the context of a testing scenario $\mathcal{O}$ with $\mathcal{O} \succeq CT$, a plausible form of must testing can be defined as well, and for finite processes plausible formalizations yield that $p \sqsubseteq_\mathcal{O}^{\text{must}} q$ iff $q \sqsubseteq_\mathcal{O} p$.

For infinite processes there are several ways to formalize must testing, and analyzing the resulting preorders falls outside of the scope if this paper.

Deadlock behaviour A process is said to reach a state of *deadlock* if it can do no further actions.[13] The process $ab + a$ for instance may deadlock right after performing an a-action, whereas the process ab may not. One could say that a semantics $\mathcal{O}$ *respects deadlock behaviour* iff $\mathcal{O} \succeq CT$. Counterexample 4 then shows that none of the semantics on the left in Figure 9 respects deadlock behaviour; only the left-hand process of Counterexample 4 can deadlock after an a-move. Respecting deadlock behaviour may be a requirement on semantics in applications where either deadlock is important in its own right, or where (implicitly) a form of must-testing is considered.

Full abstraction Many testing scenarios mentioned in this paper employ the notion that an action can happen only if it is not blocked by the environment, that is, only if both the investigated process *and* the environment are ready to participate in it. Modelling both the investigated process and the responsible part of the environment as process graphs gives rise to the following binary *intersection operator* that allows an action to happen only if it can happen in both of its arguments.

Definition 18.1 Let $\cap$ be the binary operator on process graphs defined by

- $\text{NODES}(g \cap h) = \text{NODES}(g) \times \text{NODES}(h)$,

[13] In settings were successful termination is modelled (cf. Section 19) a state of deadlock is only reached if moreover the process cannot terminate successfully.

- $\text{ROOTS}(g \cap h) = \text{ROOTS}(g) \times \text{ROOTS}(h)$,
- $((s,t), a, (s',t')) \in \text{EDGES}(g)$ iff $(s,a,s') \in \text{EDGES}(g) \wedge (t,a,t') \in \text{EDGES}(h)$.

In order to obtain a connected process graph, unreachable parts need to be removed.

This operator is also called *synchronous parallel composition* and is denoted $\|$ in HOARE [31]. It can be added to BCCSP or BCSP by employing the action rule $\dfrac{p \xrightarrow{a} p',\ q \xrightarrow{a} q'}{p \cap q \xrightarrow{a} p' \cap q'}$.

Trace semantics turns out to compositional for the intersection operator, i.e. if $g =_T g'$ and $h =_T h'$ then $g \cap h =_T g' \cap h'$. For $T(g \cap h) = T(g) \cap T(h)$. So are failures and readiness semantics:

$$\langle \sigma, X \rangle \in F(g \cap h) \ \Leftrightarrow\ \exists \langle \sigma, Y \rangle \in F(g),\ \langle \sigma, Z \rangle \in F(h) : X = Y \cup Z$$

$$\langle \sigma, X \rangle \in R(g \cap h) \ \Leftrightarrow\ \exists \langle \sigma, Y \rangle \in R(g),\ \langle \sigma, Z \rangle \in R(h) : X = Y \cap Z.$$

In fact, it is not hard to see that all semantics of this paper are compositional for $\cap$, except for CT and CS, and their (in)finitary versions. The two processes of Counterexample 3, $ab + a(b+c)$ and $a(b+c)$, are completed trace equivalent, even completed simulation equivalent, yet after intersecting them with ac only the first one has a completed trace a.

In applications where the intersection operator is used, one may require a suitable semantics to be compositional for it. This rules out CT and CS. If also deadlock behaviour is of importance, F appears to be the coarsest semantics to be considered, as least among the ones reviewed in this paper. As a matter of fact, it is the coarsest semantics even among the ones not reviewed here.

Definition 18.2 An equivalence relation is called *fully abstract* w.r.t. a property if it is the coarsest equivalence with that property, i.e. if it has the property, and any other equivalence having that property is finer.

An equivalence is said to *fully abstract* w.r.t. another equivalence $\sim$ and some operators, if it is the coarsest equivalence finer than $\sim$ that is compositional w.r.t. those operators.

An equivalence $\approx$ on $\mathbb{G}$ is fully abstract w.r.t. an equivalence $\sim$ and a set L of operators on $\mathbb{G}$ iff
(1) it is compositional w.r.t. the operators in L, and
(2) for any two process graphs $g, h \in \mathbb{G}$ with $g \not\approx h$ there exists a context $C[\cdot]$ of operators from L such that $C[g] \not\sim C[h]$.

In fact, for every equivalence relation $\sim$ on $\mathbb{G}$ and every set L of operators on $\mathbb{G}$ there exists a unique equivalence relation $\approx$ that is fully abstract w.r.t. $\sim$ and the operators in L, namely the one defined by $g \approx h$ iff $C[g] \sim C[h]$ for every context $C[\cdot]$ of operators from L.

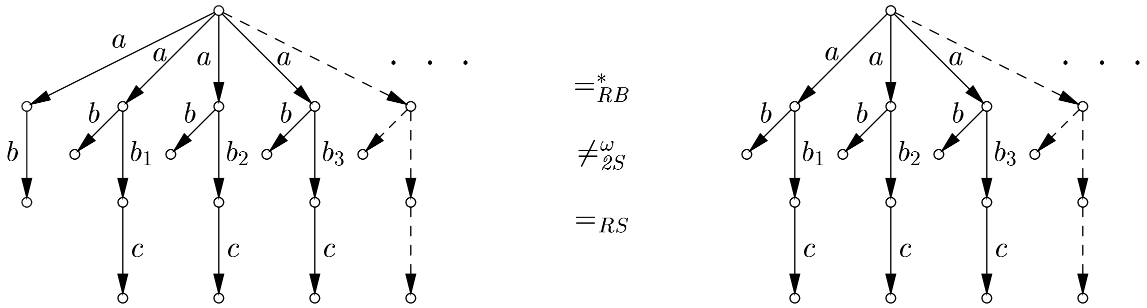
Theorem 13 Failures equivalence is fully abstract w.r.t. $=_{CT}$ and $\cap$, i.e. w.r.t. deadlock behaviour and intersection.

Proof: (1) has already been established. For (2), let $g \neq_F h$. W.l.o.g. let $\langle \sigma, X \rangle \in F(g) - F(h)$. Let k be the process graph that is shaped like the failure pair $\langle \sigma, X \rangle$, i.e. the process that performs the actions of σ in succession, after which it offers a choice between the actions of X, and nothing else. Then $\sigma \in CT(g \cap k) - CT(h \cap k)$. $\square$

Variants of Theorem 13 are abundant in the literature. See e.g. [11].

Renaming For every function $f : Act \to Act$ one can define a unary renaming operator on $\mathbb{G}$ that renames the labels of all transitions in its argument according to f. In case f is injective, all semantics of this paper are compositional for the associated renaming operator, as is trivial to check. Non-injective renaming operators are useful to express a degree of abstraction. Imagine a process that can do, among others, actions a_1 and a_2. At some level of abstraction, the difference between a_1 and a_2 may be considered irrelevant. This can be expressed by applying a renaming that relabels both a_1 and a_2 into the same action a. Naturally, if two processes are equivalent before applying such a renaming operator, one would expect them to still be equivalent afterwards, i.e. after abstracting from the difference between a_1 and a_2. It is for this reason that one might require semantics to be compositional for (non-injective) renaming. As it happens, all semantics between F^1 and B^- fail this requirement. For the two processes of Counterexample 4 are HML-equivalent ($=_B^-$), but after renaming all actions b_i into b (for $i = 1, 2, \ldots$) the resulting processes are not even singleton-failures equivalent ($=_F^1$). For only the first one has a singleton-failure pair $\langle a, b\rangle$. This can be considered an argument against the semantics on the left of Figure 9.

Counterexample 20 shows that also $F2S^*$, $R2S^*$, FB^* and RB^* are not compositional for renaming. In this counterexample b is a shorthand for $\Sigma_{i=1}^{\infty} b_i$, in the sense that whenever a transition $p \xrightarrow{b} q$ is displayed, all the transitions $p \xrightarrow{b_i} q$ for $i \geq 1$ are meant to be present. With some effort



Counterexample 20: $F2S^*$, $R2S^*$, FB^* and RB^* are not compositional for renaming

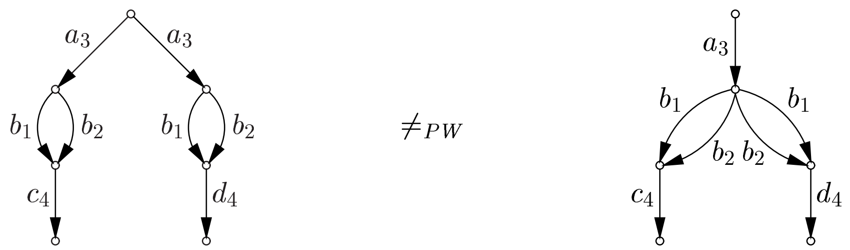
one checks that both processes satisfy the same formulas in $\mathcal{L}^*_{RB}$. However, after renaming all actions b_i into b they are no longer $2S^-$-equivalent: only the first process satisfies $a\neg(bc\top)$. For all other semantics of Figure 9 it is rather easy to establish that they are compositional for renaming.

Other compositionality requirements Many formal languages for the description of concurrent systems, including CCS [37], SCCS [39], CSP [31] and ACP [7], are *De Simone languages* (cf. [3]). This means that their operators (the *De Simone operators*) can be defined with action rules of a particular form (the *De Simone format*). Because De Simone languages are used heavily in algebraic system verification, semantic equivalences that are compositional for such languages are often desirable.

Theorem 14 The semantics T, T^∞, F, F^∞, R, R^∞, FT, FT^∞, RT, RT^∞, PF, PF^∞, S^*, S^ω, S, FS^*, RS^*, RS^ω, RS, $2S^\omega$, $2S$, B^ω and B are compositional w.r.t. all De Simone languages.
Proof: Omitted. □

For all the other semantics of Figure 9, which are displayed there in red (or shaded), there are counterexamples against such a result. Tree semantics fails to be compositional w.r.t. the + of

BCCSP, unless the action relations are upgraded with multiplicities, but that takes us outside of De Simone format. The semantics F^1, F^-, R^-, FT^-, RT^-, RS^-, $2S^-$, B^-, $F2S^*$, $R2S^*$, FB^* and RB^* fail to be compositional w.r.t. renaming, and CT, CT^∞, CS, CS^ω, CS fail to be compositional w.r.t. intersection. These are all De Simone operators. Finally, Counterexample 21 shows that PW is not compositional for the synchronization operator $\times$ of SCCS [39]—also a De Simone operator. This operator can be used to create a context, in which the two possible worlds equivalent processes of Counterexample 8 are converted into the two processes below. These are no longer possible



$(abc + abd) \times 3(1.4 + 2.4)$ $\qquad$ $a(bc + bd) \times 3(1.4 + 2.4)$

Counterexample 21: Possible worlds semantics is not compositional for synchronization

worlds equivalent, for only the one on the right has a possible world $a_3(b_1c_4 + b_2d_4)$. The same counterexample can also be created with the inverse image operator of CSP [31].

In Baeten, Bergstra & Klop [6] a unary priority operator was defined on process graphs. This operator, which is not a De Simone operator, assumes a partial ordering $<$ on Act, i.e. there is one priority operator for each such ordering. The operator acts on graphs by removing all transitions (s, a, t) for which there is a transition (s, b, u) with $b > a$ (and unreachable parts are removed as well). Thus, in a choice between several actions, only the actions with maximal priority may be executed. It is known that RT, RS, B and U are compositional for the priority operators. I think that RT^∞, PW, RS^*, RS^ω, RB^* and B^ω are too. However, none of the other semantics of Figure 9 is. Thus, in applications where priority operators are used and algebraic reasoning makes compositionality essential, only semantics like RT, RS and B are recommendable.

Depending on the application, compositionality for other operators may be required as well, leading to various restrictions on the array of suitable semantics. More on which semantics are compositional for which operators can be found in Aceto, Fokkink & Verhoef [3] and the references therein.

The Recursive Specification Principle A *recursive specification* is an equation of the form $X = t$ with X a variable and t a term (in a language such as BCCSP) containing no other variables than X. (In the literature often recursive specifications are allowed to involve more variables and more such equations, but I do not need those here.) A recursive specification $X = t$ over BCCSP is *guarded* if every occurrence of X in t occurs in a subterm at' of t with $a \in Act$. Recursive specifications are meant to specify processes. A process p is said to be a *solution* of the recursive specification $X = t$, using the semantics $\mathcal{O}$, if the equation evaluates to a true statement when substituting p for X and interpreting $=$ as $=_\mathcal{O}$. The *recursive specification principle (RSP)* says that guarded recursive specifications have unique solutions. It has been established for bisimulation semantics by Milner [39] (using the language SCCS), and holds in fact for most semantics encountered in this paper. In process algebra, two processes are often proven semantically equivalent by showing that they are solutions of the same recursive specification (cf. [5]). For this purpose it is important to work with a semantics in which RSP holds. In the infinitary semantics between T^∞ and PW this is in fact not the case. For in those semantics the two different processes of Counterexample 1 are both solutions of the guarded recursive specification $X = aX + a$. For the finitary semantics this counterexample does not apply, because the two processes are identified, whereas in simulation semantics (of finer) these two processes fail to be solutions of the same recursive specification.

Other considerations In general it depends on the kind of interactions that are permitted between a process and its environment (i.e. the testing scenario) which semantics is sufficiently discriminating for a particular application. When a range of appropriate semantics is found, also considering the criteria discussed earlier in the section, the question rises which of these semantics to actually use (e.g. in making a formal verification). A natural choice is the *coarsest* of the appropriate semantics, i.e. the one which is fully abstract w.r.t. the requirements it has to meet in order to be adequate in the context in which the investigated processes will be operating. In this semantics more equations are valid than in any other. If the goal is to prove that two processes are equivalent, this may succeed when using the fully abstract semantics, whereas it may not even be true in a finer one. Sometimes it is argued that the complexity of deciding equivalence between processes is too high for certain semantics; using them would give rise to too hard verifications. However, this can not be an argument for rejecting a semantics in favour of a finer one. For doing the verification in the finer semantics is actually a *method* of establishing equivalence in the coarser semantics. In other words, when $\mathcal{O} \prec \mathcal{N}$, establishing $p =_\mathcal{O} q$ cannot be harder than establishing $p =_\mathcal{N} q$, as establishing $p =_\mathcal{N} q$ is one of the ways of establishing $p =_\mathcal{O} q$. If deciding $\mathcal{O}$-equivalence has a higher complexity than deciding $\mathcal{N}$-equivalence, the hard cases to decide must be the equations $p =_\mathcal{O} q$ for which $p =_\mathcal{N} q$ is not even true. It is especially for those applications that $\mathcal{O}$-semantics has a distinct advantage over $\mathcal{N}$-semantics. This argument has been made forcefully in VALMARI [48].

In practice, it may not always be certain in what ways the environment can interact with investigated processes, and hence what constitutes their observable behaviour. Moreover, the processes under investigation may be transferred to more powerful environments long after their initial use. One of the ways this could happen is through the introduction of more operators for which the underlying semantics has to be compositional. A big disadvantage of semantics that are fully abstract with respect to non-stable notions of observability (or non-stable sets of operators) is that whenever a verification is carried out in a such a semantics, and one decides that the context in which the verified system will be working is such that actually a little bit more can be observed that what was originally accounted for, the verification has to be completely redone. Moreover, the correctness of the investigated systems keeps depending on the completeness of the underlying testing scenario. In such cases it is preferable to carry out verifications in the finest semantics for which this is convenient. This gives stronger equivalence results, which have a greater change of surviving in conditions where the environment gets more powerful than originally anticipated. Especially using bisimulation is safe bet, as it respects the internal structure of processes to such a degree that it is hard to imagine ever running into an environment that distinguishes bisimilar processes. In BLOOM, ISTRAIL & MEYER [12] it is argued that ready simulation semantics already respects the limits of observable behaviour, so this may be a good alternative. It should be pointed out, however, that most applications involve abstraction from internal actions (not treated in this paper), and hence require variants of the semantics treated here that accommodate such abstractions. In this setting, the question of which semantics represents the limit of observable behaviour is much harder.

19 Distinguishing deadlock and successful termination

Often researchers feel the need to distinguish two ways in which a process can end: successfully (by completing its mission) or unsuccessfully (for instance because its waits for an input from the environment that will never arrive). This distinction can be formally modelled in the context of labelled transition systems by considering triples $(\mathbb{P}, \rightarrow, \surd)$ in which $(\mathbb{P}, \rightarrow)$ is a labelled transition system as in Definition 1.1 and $\surd \subseteq \mathbb{P}$ is a predicate on processes expressing which ones can terminate successfully in their current state. It may or may not be required that the processes $p \in \mathbb{P}$ with $\surd(p)$ have no outgoing transitions. Likewise, in the setting of process graphs, one studies tuples $(\textsc{nodes}(g), \textsc{root}(g), \textsc{edges}(g), \surd(g))$ with $\surd(g) \subseteq \textsc{nodes}(g)$. Now any labelled transition system over an alphabet *Act* equipped with such a successful termination predicate, can be encoded as an ordinary labelled transition system over an alphabet $Act \cup \{\surd\}$ with $\surd \notin Act$. Namely, instead of labelling the processes/states where successful termination occurs with $\surd$, one can view successful termination as a kind of action, and add $\surd$-labelled transitions from those processes/states to fresh endstates. Now any semantic equivalence defined on ordinary labelled transition systems extends to labelled transition systems with a successful termination predicate by declaring two processes equivalent iff they are equivalent in the encoded transition system. In fact, in the same way all equivalences and preorders of this paper extend to labelled transition systems equipped with arbitrary predicates $P \subseteq \mathbb{P}$. Below, three of the thusly defined equivalences are characterized explicitly in terms of $\surd$.

Definition 19.1 Let $(\mathbb{P}, \rightarrow, \surd)$ be a labelled transition system with successful termination.
$\sigma \in Act^*$ is a *terminating trace* of a process p if there is a process q such that $p \xrightarrow{\sigma} q$ and $\surd(q)$.
Let $L(p)$ denote the set of terminating traces of p (and let $T(p)$ and $CT(p)$ be defined as before).

Now two processes p and q are trace equivalent iff $T(p) = T(q)$ and $L(p) = L(q)$. They are completed trace equivalent iff $T(p) = T(q)$, $CT(p) = CT(q)$ and $L(p) = L(q)$. They are bisimulation equivalent iff there exists a binary relation R on $\mathbb{P}$ with pRq, satisfying, for $a \in Act$:

- if pRq and $p \xrightarrow{a} p'$, then $\exists q' :\ q \xrightarrow{a} q'$ and $p'Rq'$;
- if pRq and $q \xrightarrow{a} q'$, then $\exists p' :\ p \xrightarrow{a} p'$ and $p'Rq'$;
- if pRq, then $\surd(p) \Leftrightarrow \surd(q)$.

Language semantics The nondeterministic *automata* studied in *automata theory* (cf. Hopcroft & Ullman [33]) can be regarded as process graphs with a termination predicate (except that in automata theory the focus is on *finite* automata). The states $s \in \textsc{nodes}(g)$ with $\surd(s)$ are called *accepting* or *final states*, and a string $\sigma \in Act^*$ is said to be *accepted* by the automaton g iff $\sigma \in L(g)$. The set $L(g)$ of all strings accepted by g is called the *language accepted by g*. In automata theory two automata are considered equivalent iff they accept the same language. Therefore *language equivalence* can be defined as follows.

Definition 19.2 Two processes p and q in a labelled transition system with successful termination are *language equivalent*, notation $p =_L q$, if $L(p) = L(q)$. Write $p \subseteq_L q$ iff $L(p) \subseteq L(q)$.

Clearly, language semantics makes more identifications than trace semantics (i.e. $L \prec T$). It could be appended to the bottoms of Figures 1 and 9. The reason for not treating it earlier in this paper is that it cannot be defined uniformly in terms of action relations. For either the definition depends on the predicate $\surd$, which is not a part of ordinary labelled transition systems, or, when encoding the $\surd$-predicate by a transition label $\surd$, the definition treats $\xrightarrow{\surd}$ different from the other action relations.

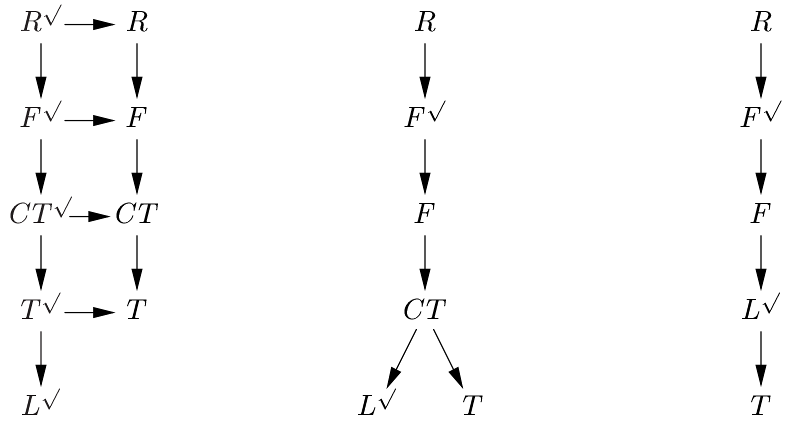
Complete axiomatizations A variant of the language BCCSP of Section 17 that distinguishes between deadlock and successful termination is the language $\mathrm{BCCSP}_{\delta\varepsilon}$, obtained from BCCSP by replacing the constant 0 by two constants δ and ε, representing deadlock and successful termination, respectively. On $\mathsf{T}(\mathrm{BCCSP}_{\delta\varepsilon})$ action relations $\xrightarrow{a}$ for $a \in Act$ are again defined as the predicates on $\mathsf{T}(\mathrm{BCCSP}_{\delta\varepsilon})$ generated by the action rules of Table 1. Furthermore, the predicate $\surd \subseteq \mathsf{T}(\mathrm{BCCSP}_{\delta\varepsilon})$ is generated by the rules of Table 6. Now the complete axiomatizations of Table 2 apply to $\mathrm{BCCSP}_{\delta\varepsilon}$

$$\surd(\varepsilon) \qquad \frac{\surd(p)}{\surd(p+q)} \qquad \frac{\surd(q)}{\surd(p+q)}$$

Table 6: Rules for the termination predicate

as well, provided that the occurrences of 0 are changed into δ, an axiom $I(\varepsilon) = \varepsilon$ is added, and the characteristic axioms for CS and CT also get variants in which $by + z$ resp. $cy + v$ is replaced by ε. Language equivalence can be axiomatized by adding the axiom $a\delta = \delta$ to the axioms for trace equivalence. This axiom corresponds with a transformation on finite process trees that removes states from which it is impossible to reach a state of successful termination. On the normal forms w.r.t. this transformation, language equivalence and trace equivalence coincide.

Successful termination as default Naturally, ordinary transition systems can be regarded as transition systems with successful termination by taking the termination predicate to be empty. On such transition systems, language equivalence turns out to be the universal relation, axiomatized by the equation $x = y$. Alternatively, ordinary transition systems can be regarded as transition systems with successful termination by letting $\surd$ be the set of processes without outgoing transitions, i.e. by regarding all termination to be successful. In this context, on a transition system $(\mathbb{P}, \rightarrow)$ on can define any of the semantics $\mathcal{O}$ of this paper as in Section 15, or by taking successful termination into account as in the present section. Denote the latter version of $\mathcal{O}$ by $\mathcal{O}^\surd$. Then two processes are $\mathcal{O}^\surd$-equivalent iff they are $\mathcal{O}$-equivalent after appending a $\surd$-transition to every endstate. Comparing semantics that take termination into account as well as semantics that abstract from it yields in first approximation a “double” version of Figure 9, of which a tiny fragment is displayed in Figure 12(a). However, for processes p for which all termination is successful one has $CT(p) = L(p)$. Hence the semantics $T^\surd$, $CT^\surd$ and CT coincide. One also verifies easily that R coincides with $R^\surd$, FT with $FT^\surd$, RT with $RT^\surd$, RS with $RS^\surd$, B with $B^\surd$, etc. However, F and $F^\surd$ differ, as demonstrated by Counterexample 22. There $F(\mathit{left}) = F(\mathit{right})$ but $\langle a, \{c, \surd\}\rangle \in F^\surd(\mathit{left}) - F^\surd(\mathit{right})$. Also PF differs from PF^0 and $2S$ from $2S^0$, for in Counterexample 14 one has $\mathit{left} =_{2S} \mathit{right}$ but, after appending a $\surd$-transition to every endnode, $a\neg b\surd 0 \in \mathcal{L}^\surd_{2S}(\mathit{left}) - \mathcal{L}^\surd_{2S}(\mathit{right})$. Thus Figure 12(a) collapses to Figure 12(b). In Groote & Huttel [24] *normed processes* are studied: processes that never loose the possibility to terminate eventually. A process p is normed iff for each process q reachable from p, there is a process r reachable from q that terminates (i.e. has

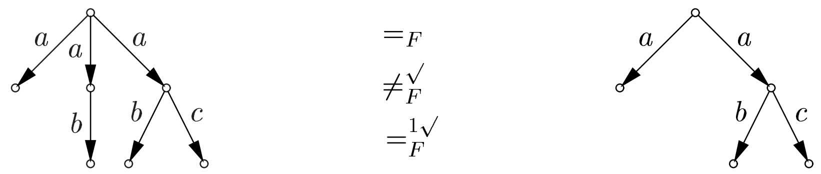


(a) first approximation (b) general case (c) normed processes

Figure 12: The linear time – branching time spectrum for successfully terminating processes

no outgoing transitions) (and all termination is considered successful). For normed processes p, $T(p)$ is completely determined by $L(p)$. Hence Figure 12(b) collapses further to Figure 12(c). This explains why in [24] $L^{\surd}$ coincides with CT and is finer than T.

Sequencing and sequential composition The *sequential composition* of processes p and q (cf. [31, 7]), denoted $p \cdot q$, is the process that first executes p, and upon successful termination of p executes q. This operator is defined only on domains of processes on which successful termination is somehow represented. *Sequencing* on the other hand is defined on domains of processes that do not distinguish between deadlock and successful termination: let $p;q$ denote the process that first executes p until it can do no further actions, and then q [12]. On process graphs, $g;h$ can be constructed by appending (at its root) a disjoint copy of h to every endnode of g. On process graphs with successful termination, $g \cdot h$ on the other hand can be constructed by appending a disjoint copy of h to every node s of g with $\surd(s)$. In case $\surd(s)$ is possible even if s is not an endnode, the graph h needs to be transformed first in such a way that its root has no incoming edges [7].



$a + ab + a(b + c)$ $a + a(b + c)$

Counterexample 22: Failures semantics is not compositional for sequencing

Counterexample 22 shows that failures semantics is not compositional for sequencing. There *left* $=_F$ *right*, but *left*;$c \neq_F$ *right*;c. The same counterexample, with all endnodes successfully terminating, shows that singleton-failures semantics is not compositional for either sequencing or sequential composition. Likewise, Counterexample 14 shows that *PF* and *2S* are not compositional for sequencing, and Counterexample 4 shows that none of the semantics between T and B^- are. All of the semantics studied in this paper, except for F^1, are compositional for sequential composition. As sequencing is the same as sequential composition on processes where all endstates, and only those, are considered to be successfully terminating, this implies that all the semantics $\mathcal{O}^{\surd}$, except for $F^{1\surd}$, are compositional for sequencing. If c is an action that does not occur in p or q, then $p;c =_{\mathcal{O}} q;c \Leftrightarrow p =_{\mathcal{O}}^{\surd} q$. (Think of c as $\surd$.) From this it follows that for all semantics $\mathcal{O}$, except F^1, $\mathcal{O}^{\surd}$ is fully abstract w.r.t. $\mathcal{O}$ and sequencing, at least for processes that can not execute every action in *Act*.

Concluding remarks

In this paper various semantic equivalences for concrete sequential processes are defined, motivated, compared and axiomatized. Of course many more equivalences can be given than the ones presented here. The reason for selecting just these, is that they can be motivated rather nicely and/or play a rôle in the literature on semantic equivalences. In Abramsky & Vickers [2] the observations which underly many of the semantics in this paper are placed in a uniform algebraic framework, and some general completeness criteria are stated and proved. They also introduce *acceptance semantics*, which can be obtained from acceptance-refusal semantics (Section 7) by dropping the refusals, and analogously *acceptance trace semantics*. I am not aware of any reasonable testing scenario for these notions.

In Section 9 I remarked that a testing scenario for simulation and ready simulation semantics can be obtained by adding an *undo*-button to the scenario's for trace and ready trace semantics. Likewise, Schnoebelen [47] investigates the addition of an *undo*-button to the testing scenarios for completed trace, readiness, failures and failure trace semantics, thereby obtaining 3 new equivalences $CT_{\#}$, $R_{\#}$ and $F_{\#}$. *Undo*-failure trace equivalence coincides with finitary failure simulation equivalence, just like *undo*-trace and *undo*-ready trace equivalence coincide with finitary simulation and finitary ready simulation equivalence. For image finite processes $R_{\#}$ coincides with $F_{\#}$. Furthermore $R \preceq R_{\#} \preceq RS^*$, $F \preceq F_{\#} \preceq FS^*$, $CT \preceq CT_{\#} \preceq CS^*$ and $S^* \preceq CT_{\#} \preceq F_{\#} \preceq R_{\#}$.

An interesting topic is the generalization of this work to a setting with silent moves and/or with parallelism. In both cases there turn out to be many interesting variations. The generalization to a setting with invisible actions will be tackled in [21]. Some work towards generalizing the spectrum to a setting with parallelism can be found for instance in [44] and [19].

References

[1] S. Abramsky (1987): *Observation equivalence as a testing equivalence. Theoretical Computer Science* 53, pp. 225–241.

[2] S. Abramsky & S. Vickers (1993): *Quantales, observational logic and process semantics. Mathematical Structures in Computer Science* 3, pp. 161–227.

[3] L. Aceto, W.J. Fokkink & C. Verhoef (2000): *Structural operational semantics.* In J.A. Bergstra, A. Ponse & S.A. Smolka, editors: *Handbook of Process Algebra*, chapter 3. Elsevier.

[4] P. Aczel (1988): *Non-well-founded Sets*, *CSLI Lecture Notes* 14. Stanford University.

[5] J.C.M. Baeten, editor (1990): *Applications of Process Algebra.* Cambridge Tracts in Theoretical Computer Science 17. Cambridge University Press.

[6] J.C.M. Baeten, J.A. Bergstra & J.W. Klop (1987): *Ready-trace semantics for concrete process algebra with the priority operator.* Computer Journal 30(6), pp. 498–506.

[7] J.C.M. Baeten & W.P. Weijland (1990): *Process Algebra.* Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press.

[8] J.W. de Bakker, J.N. Kok, J.-J.Ch. Meyer, E.-R. Olderog & J.I. Zucker (1986): *Contrasting themes in the semantics of imperative concurrency.* In J.W. de Bakker, W.P. de Roever & G. Rozenberg, editors: *Current Trends in Concurrency*, LNCS 224, Springer, pp. 51–121.

[9] J.W. de Bakker & J.I. Zucker (1982): *Processes and the denotational semantics of concurrency.* Information and Control 54(1/2), pp. 70–120.

[10] J.A. Bergstra, J.W. Klop & E.-R. Olderog (1986): *Failure semantics with fair abstraction.* Report CS-R8609, CWI, Amsterdam.

[11] J.A. Bergstra, J.W. Klop & E.-R. Olderog (1988): *Readies and failures in the algebra of communicating processes.* SIAM Journal on Computing 17(6), pp. 1134–1177.

[12] B. Bloom, S. Istrail & A.R. Meyer (1995): *Bisimulation can't be traced.* Journal of the ACM 42(1), pp. 232–268.

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

[14] S.D. Brookes & A.W. Roscoe (1985): *An improved failures model for communicating processes.* In S.D. Brookes, A.W. Roscoe & G. Winskel, editors: *Seminar on Concurrency*, LNCS 197, Springer, pp. 281–305.

[15] P. Darondeau (1982): *An enlarged definition and complete axiomatisation of observational congruence of finite processes.* In M. Dezani-Ciancaglini & U. Montanari, editors: *Proceedings international symposium on programming: 5^{th} colloquium, Aarhus*, LNCS 137, Springer, pp. 47–62.

[16] R. De Nicola (1987): *Extensional equivalences for transition systems.* Acta Informatica 24, pp. 211–237.

[17] R. De Nicola & M. Hennessy (1984): *Testing equivalences for processes.* Theoretical Computer Science 34, pp. 83–133.

[18] J. Engelfriet (1985): *Determinacy $\rightarrow$ (observation equivalence = trace equivalence).* Theoretical Computer Science 36(1), pp. 21–25.

[19] R.J. van Glabbeek (1990): *The refinement theorem for ST-bisimulation semantics.* In M. Broy & C.B. Jones, editors: Proceedings IFIP TC2 Working Conference on *Programming Concepts and Methods*, Sea of Gallilee, Israel, North-Holland, pp. 27–52.

[20] R.J. van Glabbeek (1990): *The linear time – branching time spectrum.* Report CS-R9029, CWI, Amsterdam. Extended abstract in J.C.M. Baeten & J.W. Klop, editors: Proceedings *CONCUR '90, Theories of Concurrency: Unification and Extension*, Amsterdam, August 1990, LNCS 458, Springer-Verlag, 1990, pp. 278–297.

[21] R.J. van Glabbeek (1993): *The linear time – branching time spectrum II; the semantics of sequential systems with silent moves.* Manuscript. Preliminary version available by ftp at `ftp://boole.stanford.edu/pub/spectrum.ps.gz`. Extended abstract in E. Best, editor: Proceedings *CONCUR'93*, 4^{th} International Conference on Concurrency Theory, Hildesheim, Germany, August 1993, LNCS 715, Springer, pp. 66–81.

[22] R.J. van Glabbeek & J.J.M.M. Rutten (1989): *The processes of De Bakker and Zucker represent bisimulation equivalence classes.* In *J.W. de Bakker, 25 jaar semantiek, liber amicorum*, CWI, Amsterdam, pp. 243–246.

[23] J.F. Groote (1990): *A new strategy for proving ω–completeness with applications in process algebra.* In J.C.M. Baeten & J.W. Klop, editors: *Proceedings CONCUR 90,* Amsterdam, LNCS 458, Springer, pp. 314–331.

[24] J.F. Groote & H. Hüttel (1994): *Undecidable equivalences for basic process algebra.* *Information and Control* 115(2), pp. 354–371.

[25] J.F. Groote & F.W. Vaandrager (1992): *Structured operational semantics and bisimulation as a congruence.* *Information and Computation* 100(2), pp. 202–260.

[26] M. Hennessy (1985): *Acceptance trees.* *Journal of the ACM* 32(4), pp. 896–928.

[27] M. Hennessy & R. Milner (1980): *On observing nondeterminism and concurrency.* In J.W. de Bakker & J. van Leeuwen, editors: *Proceedings 7^{th} ICALP, Noorwijkerhout*, LNCS 85, Springer, pp. 299–309. This is a preliminary version of:

[28] M. Hennessy & R. Milner (1985): *Algebraic laws for nondeterminism and concurrency.* *Journal of the ACM* 32(1), pp. 137–161.

[29] C.A.R. Hoare (1978): *Communicating sequential processes.* *Communications of the ACM* 21(8), pp. 666–677.

[30] C.A.R. Hoare (1980): *Communicating sequential processes.* In R.M. McKeag & A.M. Macnaghten, editors: *On the construction of programs – an advanced course*, Cambridge University Press, pp. 229–254.

[31] C.A.R. Hoare (1985): *Communicating Sequential Processes.* Prentice Hall, Englewood Cliffs.

[32] M.J. Hollenberg (1995): *Hennessy-Milner classes and process algebra.* In A. Ponse, M. de Rijke & Y. Venema, editors: *Modal Logic and Process Algebra: a Bisimulation Perspective*, *CSLI Lecture Notes* 53, CSLI Publications, Stanford, California, pp. 187–216.

[33] J.E. Hopcroft & J.D. Ullman (1979): *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley.

[34] J.K. Kennaway (1981): *Formal semantics of nondetermism and parallelism.* PhD thesis, University of Oxford.

[35] K.G. Larsen & A. Skou (1991): *Bisimulation through probabilistic testing.* *Information and Computation*, 94.

[36] A.R. Meyer (1985): *Report on the 5^{th} international workshop on the semantics of programming languages in Bad Honnef.* Bulletin of the European Association for Theoretical Computer Science 27, pp. 83–84.

[37] R. Milner (1980): *A Calculus of Communicating Systems*, LNCS 92. Springer.

[38] R. Milner (1981): *Modal characterisation of observable machine behaviour.* In G. Astesiano & C. Bohm, editors: *Proceedings CAAP 81*, LNCS 112, Springer, pp. 25–34.

[39] R. Milner (1983): *Calculi for synchrony and asynchrony.* Theoretical Computer Science 25, pp. 267–310.

[40] E.-R. Olderog & C.A.R. Hoare (1986): *Specification-oriented semantics for communicating processes.* Acta Informatica 23, pp. 9–66.

[41] D.M.R. Park (1981): *Concurrency and automata on infinite sequences.* In P. Deussen, editor: 5^{th} *GI Conference*, LNCS 104, Springer, pp. 167–183.

[42] I.C.C. Phillips (1987): *Refusal testing.* Theoretical Computer Science 50, pp. 241–284.

[43] A. Pnueli (1985): *Linear and branching structures in the semantics and logics of reactive systems.* In W. Brauer, editor: *Proceedings 12^{th} ICALP, Nafplion*, LNCS 194, Springer, pp. 15–32.

[44] L. Pomello (1986): *Some equivalence notions for concurrent systems – An overview.* In G. Rozenberg, editor: *Advances in Petri Nets 1985*, LNCS 222, Springer, pp. 381–400.

[45] A.W. Roscoe (1993): *Unbounded non-determinism in CSP.* Journal of Logic and Computation 3(2), pp. 131–172.

[46] W.C. Rounds & S.D. Brookes (1981): *Possible futures, acceptances, refusals and communicating processes.* In 22^{th} *Annual Symposium on Foundations of Computer Science,* Nashville, Tennessee, IEEE, New York, pp. 140–149.

[47] Ph. Schnoebelen (1991): *Experiments on processes with backtracking.* In J.C.M. Baeten & J.F. Groote, editors: *Proceedings CONCUR 91,* Amsterdam, LNCS 527, Springer, pp. 80–94.

[48] A. Valmari (1995): *Failure-based equivalences are faster than many believe.* In J. Desel, editor: Proceedings of the International Workshop on *Structures in Concurrency Theory,* Berlin, May 1995, Workshops in Computing, Springer, pp. 326–340.

[49] S. Veglioni & R. De Nicola (1998): *Possible worlds for process algebras.* In D. Sangiorgi & R. de Simone, editors: *Proceedings CONCUR 98,* Nice, France, LNCS 1466, Springer, pp. 179–193.

[50] G. Winskel (1984): *Synchronization trees.* Theoretical Computer Science 34(1/2), pp. 33–82.

Index