Keep it Fair: Equivalences*

Tobias Prehn
Modelle und Theorie Verteilter Systeme
TU Berlin, Germany
tobias.prehn@tu-berlin.de

Stephan Mennicke
Institut für Programmierung und Reaktive Systeme
TU Braunschweig, Germany
mennicke@ips.cs.tu-bs.de

For models of concurrent and distributed systems, it is important and also challenging to establish correctness in terms of safety and/or liveness properties. Theories of distributed systems consider equivalences fundamental, since they (1) preserve desirable correctness characteristics and (2) often allow for component substitution making compositional reasoning feasible. Modeling distributed systems often requires abstraction utilizing nondeterminism which induces unintended behaviors in terms of infinite executions with one nondeterministic choice being recurrently resolved, each time neglecting a single alternative. These situations are considered unrealistic or highly improbable. Fairness assumptions are commonly used to filter system behaviors, thereby distinguishing between realistic and unrealistic executions. This allows for key arguments in correctness proofs of distributed systems, which would not be possible otherwise. Our contribution is an equivalence spectrum in which fairness assumptions are preserved. The identified equivalences allow for (compositional) reasoning about correctness incorporating fairness assumptions.

1 Introduction

In theories of concurrent and distributed systems, nondeterminism is central for describing system behavior. Thereby, we highly abstract from implementation details in the decision-making processes to be realized, while focusing on all possible system behaviors. A common abstraction therefore is nondeterminism, especially when it comes to the description of alternative behaviors. Nondeterminism in system modeling falls apart into two categories. *Internal nondeterminism* are those situations in which the observation of an action does not allow for the prediction of the successor state. On the other hand, *external nondeterminism* are circumstances where the successor state may be determined, but it is impossible to predict or influence which one of two or more conflicting actions occurs, e. g., the choice for one specific action may be controlled by an environment the system works in. External nondeterminism is ubiquitous in distributed systems where usually two or more components contribute to the system behavior being modeled by an interleaving of the component actions. For liveness properties ("something good eventually happens"), nondeterminism gives rise to system behaviors considered unrealistic. The "something good" may be in reach infinitely many times but each time, the corresponding action may be evaded. In this paper, we study the implications nondeterminism has on proofs for liveness properties.

In 1965, Edsger W. Dijkstra unleashed a controversial debate on system correctness proofs with his seminal paper on a problem in concurrent programming, today coined to the term *mutual exclusion* [4]. Roughly a year later, Donald E. Knuth noticed and reported an error in Dijkstra's algorithm by showing an erroneous execution sequence, contradicting one aspect of the algorithm, and in conclusion, the proof of correctness as given by Dijkstra [9]. The execution sequence Knuth took constitutes a corner case where one of two agents must be assumed to be indefinitely faster than the other component in order to constantly block the access to the critical section. Let us recapitulate the situation with a less complex

*This work was funded by the DFG (German Research Foundation), grants GO-671/6-2 and NE-1505/2-2.

M. Bartoletti, L. Bocchi, L. Henrio, S. Knight (Eds.):
10th Interaction and Concurrency Experience (ICE 2017)
EPTCS 261, 2017, pp. 5–16, doi:10.4204/EPTCS.261.4

Figure 1: Two processes for which (a) strong fairness of c and (b) weak fairness of c ensures that eventually only actions a are enabled.

example, given by the system depicted in Fig. 1(a). The system behavior of a process q is modeled in a *labeled transition system* (LTS) containing q. The use of LTS gives us a well studied, commonly used formalism and a wide applicability of our results since there exist numerous encodings from other formalisms into LTS and vice versa. When considering the behavior of a process, the respective state may be seen as the initial state of the LTS. One of two things may happen:

1. Either action a occurs, leading to a state where action c is momentarily disabled
2. or action c occurs and afterwards action a may occur infinitely often.

For the sake of illustration, assume that action a is performed by a component K_1 and the observation of action c is due to component K_2. The intuition is that K_1 uses some resource that K_2 needs to perform action c and returns it every even occurrence of action a, as long as c did not occur. Assume that a system correctness property requires us to prove that action c eventually occurs, e. g., in order to ensure that K_2 eventually terminates. Of course, we directly derive a counterexample by an infinite sequence of actions a, i. e., $w = a^\omega$ disproves the desired theorem.

In the example, whenever a recurring decision is independent of outer circumstances, one option may continuously be selected in favor of another, leading to an unfair treatment of the neglected actions, and surely also of the neglected component. This problem is often addressed by posing fairness assumptions, guaranteeing the elimination of the aforementioned unfair treatment of components and/or actions. At this level of abstraction, it is simply unrealistic that one component, here C_2, is infinitely slower than another, when competing for a limited resource. It is sufficient to assume that C_2 eventually gets a grip on the limited resource to perform action c, as long as an even number of actions a occurs infinitely often. With this assumption in mind, it is now possible to rule out the counterexample w. All other infinite behaviors certainly contain action c thus ensuring the required theorem. Thus, a fairness assumption defines a certain balance between the theoretically possible and realistic system behavior.

In practice, process creation involves specification and implementation, allowing for several design decisions of the same effect. To identify systems that are capable of the same behavior and to match implementations against specifications, several equivalence notions are formulated, each meeting different requirements w. r. t. correctness, e. g., deadlock-freedom. Some of those equivalences form a hierarchy [6, 7] and many process logics, as e. g., HML, LTL, or CTL, enjoy the property of *bisimulation invariance* [8, 12]. The latter constitutes an important insight in process theory, stating that whenever two systems are equivalent under bisimulation, then they enjoy the same properties as formulated in a certain logic. Regardless of the strengths of the equivalences in van Glabbeek's spectrum, most of them do not consider system assumptions, such as fairness, stated outside of the used formalism. However, in a system development process, when refining a system specification, such as the one given in Fig. 1(a), to another system that is not bisimilar to the specification, it is unclear to what degree also the assumptions of fairness are carried over. For example, trace equivalence is a candidate that changes the set of fair runs from one process to another equivalent process, being sketched in Sect. 3. This means that, even if we

need to reprove certain properties for trace equivalent system refinements, the same fairness assumptions as posed upon the specification yields new paths to be taken into account. We prove for all equivalences at least as strong as failure trace equivalence, that equivalent systems yield the same fair behavior. On the other hand, equivalences at most as strong as ready equivalence do not preserve fairness in this respect.

We consider strong fairness as well as weak fairness, as introduced by Plotkin [13] or Francez [5]. Formal definitions are adapted from notions defined by Reisig [16]. A system run is strong fair w. r. t. some action iff either this action occurs infinitely often or it is enabled only finitely often. An action a is *enabled* in a process if it is possible to execute a, thereby reaching another process. In our example of Fig. 1(a), assuming that action c, and thereby component C_2, is treated strong fair, ensures the success of the theorem. Consider the same theorem for the system in Fig. 1(b). Here, also strong fairness for action c is sufficient for proving the desired property. In this case, strong fairness is not necessary, since also weak fairness suffices. A system run is weak fair w. r. t. some action, here c, iff either it occurs infinitely often or it is not always enabled from some state within the run. Since in Fig. 1(b), process p enables action c and leaves it enabled after an action a occurs, the assumption of weak fairness ensures that c needs to be disabled in a weak fair run, again ruling out the canonical counterexample $w = a^\omega$.

The paper is structured as follows. Sect. 2 introduces the notion of labeled transition systems and a selection of semantic equivalences relevant to this work. The notions of fairness are defined in Sect. 3 and Sect. 4, for each of which an area within the linear-time branching-time spectrum is identified for which fairness is preserved. We conclude the paper by a discussion on related work and plans for future work by Sect. 5.

2 Preliminaries

Here, we cover the definitions of labeled transition systems (LTS) as well as a selection of equivalences on LTS related and classified by van Glabbeek in [6]. LTS is a formalism to describe abstract behaviors while LTS equivalences relate such behaviors to one another. An LTS is a state-transition graph where each transition is labeled by a letter from an alphabet Σ, the set of all (abstract) actions.

Definition 1 (Labeled Transition System)**.** Let Σ be an alphabet. A *labeled transition system* (LTS) is a triple $A = (Q, \Sigma, \longrightarrow)$ with a set of *processes* Q and a labeled transition relation $\longrightarrow \subseteq Q \times \Sigma \times Q$.

Throughout the paper, processes range over p, q, r, s, p', q' as well as p_i and q_i for $i \in \mathbb{N}$. We abbreviate $(p, a, p') \in \longrightarrow$ to $p \xrightarrow{a} p'$. $p \xrightarrow{a}$ denotes enabledness of action a in process p, i. e., there exists $p' \in Q$ such that $p \xrightarrow{a} p'$. For a transition $t = (p, a, p') \in \longrightarrow$, $l(t) := a$ defines its label projection. Fairness notions are based on *runs* of processes. A run of a process p_0 is an alternating sequence of processes and actions, initiated by p_0. Runs come in finite or infinite length. Traces of infinite runs are elements of Σ^ω, i. e., all infinite words over Σ.

Definition 2 (Process Run)**.** Let $A = (Q, \Sigma, \longrightarrow)$ be an LTS. A *finite run of* $p_0 \in Q$ is a sequence of the form $\rho = p_0 a_1 \dots p_{n-1} a_n p_n$ with $p_i \in Q$ and $a_i \in \Sigma$ such that $p_i \xrightarrow{a_{i+1}} p_{i+1}$ $(0 \leq i < n)$. An *infinite run of* $p_0 \in Q$ is an infinite sequence of the form $\rho = p_0 a_1 p_1 a_2 \dots$ with $p_i \in Q$ and $a_i \in \Sigma$ such that $p_i \xrightarrow{a_{i+1}} p_{i+1}$ $(0 \leq i)$. The set of all finite and infinite runs of p_0 is denoted by $Rn(p_0)$.

Subsequently, we introduce eight different semantics on processes, formally. Therefore, we assume a single LTS $A = (Q, \Sigma, \longrightarrow)$ containing all processes mentioned throughout the following paragraphs. All definitions carry over to a setting where each process is identified by a single LTS with distinct initial state. As fairness assumptions deal with infinite runs only, we use the infinitary versions of trace-based semantics. Trace semantics simply enumerates all finite and infinite action sequences of a process. Two processes are equivalent if their sets of action sequences are equal.

Definition 3 (Trace Semantics [6])**.** Let $A = (Q, \Sigma, \longrightarrow)$ be an LTS and $\rho = p_0 a_1 \dots p_{n-1} a_n p_n$ be a run of p_0. The *trace of* ρ is defined by $tr(\rho) := a_1 a_2 \dots a_n \in \Sigma^*$. The transition relation of LTS naturally extends to traces $\sigma = a_1 a_2 \dots a_n \in \Sigma^*$ as $p_0 \xrightarrow{\sigma} p_n$. The set of all finite traces of p_0 is denoted by $Traces(p_0)$.

Let $\rho' = p_0 a_1 p_1 a_2 \dots$ be an infinite run of p_0. The *trace of* ρ' is defined by $tr(\rho') := a_1 a_2 \dots \in \Sigma^\omega$. The set of all infinite traces is defined by $Traces^\infty(p_0)$. Two processes $p_0, q_0 \in Q$ are *infinitary trace equivalent*, denoted by $p_0 \equiv_T^\infty q_0$ iff $Traces(p_0) = Traces(q_0)$ and $Traces^\infty(p_0) = Traces^\infty(q_0)$.

Trace semantics is a linear-time semantics not distinguishing between systems producing the same traces via intermediate processes capable of different actions. Ready semantics respects the branching structure by combining traces of a process with all actions enabled by a process reached by this trace, the so-called ready set. Therefore, ready sets can be seen as system logs for a specific point in time, stating the actions that have been taken so far and the actions being possible at this point.

Definition 4 (Ready Semantics [6])**.** Let $A = (Q, \Sigma, \longrightarrow)$ be an LTS. A *ready pair of* $p_0 \in Q$ is a pair $(\sigma, X) \in \Sigma^* \times 2^\Sigma$ such that there exists a $p \in Q$ with $p_0 \xrightarrow{\sigma} p$ and $X = \{a \in \Sigma \mid p \xrightarrow{a}\}$. The set of all ready pairs of p_0 is denoted by $R(p_0)$. Two processes $p_0, q_0 \in Q$ are *infinitary ready equivalent*, denoted $p_0 \equiv_R^\infty q_0$ iff $R(p_0) = R(q_0)$ and $p_0 \equiv_T^\infty q_0$.

Although the infinitary versions of ready semantics respects infinite runs to some degree, branching is only considered to a finite extent. Ready trace semantics, on the other hand, removes this restriction by integrating ready sets directly into finite and infinite traces. In comparison to ready pairs, the ready traces can be seen as system logs where not only the taken actions are represented but also the alternative, enabled actions that have been neglected. This is achieved by allowing for (but not enforcing) intermediate ready sets.

Definition 5 (Ready Trace Semantics [6])**.** A word $\sigma_1 \sigma_2 \dots \sigma_n \in (\Sigma \cup 2^\Sigma)^*$ is a *ready trace of* $p_0 \in Q$ iff there are states $p_1, p_2, \dots, p_n \in Q$ such that for each $0 \leq i < n$ either

(a) $\sigma_{i+1} \in \Sigma$ and $p_i \xrightarrow{\sigma_{i+1}} p_{i+1}$ or

(b) $\sigma_{i+1} \in 2^\Sigma$ and $p_i = p_{i+1}$ and $\sigma_{i+1} = \{a \in \Sigma \mid p_i \xrightarrow{a}\}$.

The set of all ready traces of p_0 is denoted by $RT(p_0)$. An *infinite ready trace of* $p_0 \in Q$ is a word $\sigma_1 \sigma_2 \dots \in (\Sigma \cup 2^\Sigma)^\omega$ iff there are states $p_1, p_2, \dots \in Q$ such that for each $0 \leq i$ either (a) or (b) holds. By $RT^\infty(p_0)$ we denote the set of all infinite ready traces of p_0. Two processes $p_0, q_0 \in Q$ are *infinitary ready trace equivalent*, denoted $p_0 \equiv_{RT}^\infty q_0$ iff $RT(p_0) = RT(q_0)$ and $RT^\infty(p_0) = RT^\infty(q_0)$.

A natural counterpart of ready semantics is failures semantics. Its central idea relies on *failure pairs*, each consisting of a finite trace and a (not necessarily maximal) set of actions refused by a process reachable via the trace. Failures semantics can be seen as system logs similar to ready semantics. They also state the actions taken up to some point in time but close with a set of impossible actions.

Definition 6 (Failures Semantics [6])**.** Let $A = (Q, \Sigma, \longrightarrow)$ be an LTS. A *failure pair of* $p_0 \in Q$ is a pair $(\sigma, X) \in \Sigma^* \times 2^\Sigma$ such that there exists a $p \in Q$ with $p_0 \xrightarrow{\sigma} p$ and $X \subseteq \{a \in \Sigma \mid p \not\xrightarrow{a}\}$. The set of all failure pairs of p_0 is denoted by $F(p_0)$. Two processes $p_0, q_0 \in Q$ are *infinitary failures equivalent*, denoted $p_0 \equiv_F^\infty q_0$ iff $F(p_0) = F(q_0)$ and $p_0 \equiv_T^\infty q_0$.

In contrast to the sets of all enabled actions that are integrated into the ready traces, the sets integrated into failure traces contain some of the refused actions.

Definition 7 (Failure Trace Semantics [6])**.** A word $\sigma_1 \sigma_2 \dots \sigma_n \in (\Sigma \cup 2^\Sigma)^*$ is a *failure trace of* $p_0 \in Q$ iff there are states $p_1, p_2, \dots, p_n \in Q$ such that for each $0 \leq i < n$ either

(a) $\sigma_{i+1} \in \Sigma$ and $p_i \xrightarrow{\sigma_{i+1}} p_{i+1}$ or

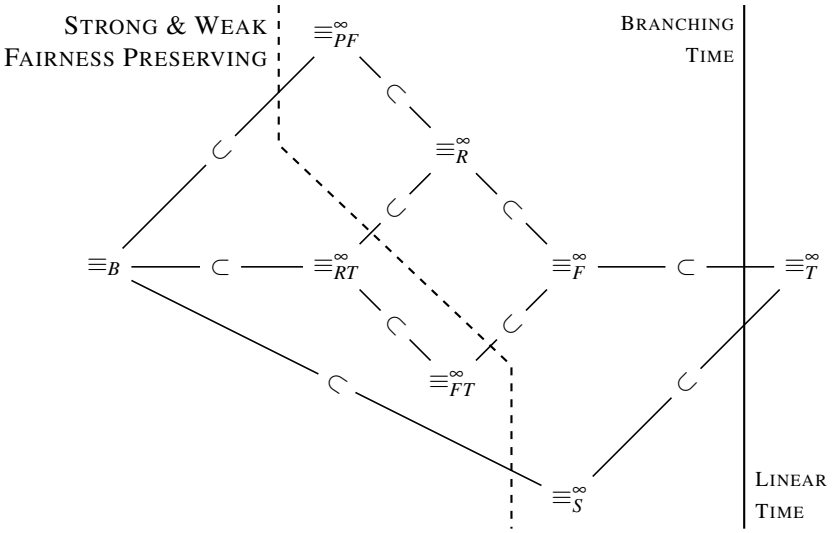

Figure 2: How the semantics relate to each other, adapted from [6].

(b) $\sigma_{i+1} \in 2^{\Sigma}$ and $p_i = p_{i+1}$ and $\sigma_{i+1} \subseteq \{a \in \Sigma \mid p_i \not\xrightarrow{a}\}$.

The set of all failure traces of p_0 is denoted by $FT(p_0)$. An *infinite failure trace of* p_0 is a word $\sigma_1\sigma_2\ldots \in (\Sigma \cup 2^{\Sigma})^{\omega}$ iff there are states $p_1, p_2, \ldots \in Q$ such that for each $0 \leq i$ either (a) or (b) holds. By $FT^{\infty}(p_0)$ we denote the set of all infinite failure traces of p_0. Two processes $p_0, q_0 \in Q$ are *infinitary failure trace equivalent*, denoted $p_0 \equiv^{\infty}_{FT} q_0$ iff $FT(p_0) = FT(q_0)$ and $FT^{\infty}(p_0) = FT^{\infty}(q_0)$.

While ready semantics takes into account the set of enabled actions and, by this, the possible next step of the reached process, possible futures semantics considers all possible next sequences of actions i. e., traces the reached process is capable of.

Definition 8 (Possible Futures Semantics [6])**.** Let $A = (Q, \Sigma, \longrightarrow)$ be an LTS. A *possible future of* $p_0 \in Q$ is a pair $(\sigma, X) \in \Sigma^* \times 2^{\Sigma^*}$ such that there exists a $p \in Q$ with $p_0 \xrightarrow{\sigma} p$ and $X = \mathsf{Traces}(p)$. The set of all possible futures of p_0 is denoted by $PF(p_0)$. Two processes $p_0, q_0 \in Q$ are *infinitary possible futures equivalent*, denoted $p_0 \equiv^{\infty}_{PF} q_0$ iff $PF(p_0) = PF(q_0)$ and $p_0 \equiv^{\infty}_{T} q_0$.

A process simulates another one when it is capable of mimicking every step of the simulated process such that the reached process again simulates the result of the original step. Two processes are simulation equivalent if one simulates the other and vice versa.

Definition 9 (Simulation Semantics)**.** Let $A = (Q, \Sigma, \longrightarrow)$ be an LTS. A *simulation* between p_0 and $q_0 \in Q$ is a binary relation $R \subseteq Q \times Q$ such that (1) $(p_0, q_0) \in R$ and (2) if $(p, q) \in R$, it holds that

- $p \xrightarrow{a} p'$ implies that $\exists q' \in Q : q \xrightarrow{a} q'$ and $(p', q') \in R$

p_0 and q_0 are *similar*, denoted $p_0 \equiv^{\infty}_{S} q_0$ iff there exists a simulation relation between p_0 and q_0 and a simulation relation between q_0 and p_0.

A bisimulation relation between to processes is a single relation that is a simulation for both directions simultaneously.

Definition 10 (Bisimulation Semantics)**.** Let $A = (Q, \Sigma, \longrightarrow)$ be an LTS. A *bisimulation* between p_0 and $q_0 \in Q$ is a binary relation $R \subseteq Q \times Q$ such that (1) $(p_0, q_0) \in R$ and (2) if $(p, q) \in R$, it holds that

- $p \xrightarrow{a} p'$ implies that $\exists q' \in Q : q \xrightarrow{a} q'$ and $(p', q') \in R$ and

- $q \xrightarrow{a} q'$ implies that $\exists p' \in P : p \xrightarrow{a} p'$ and $(p',q') \in R$.

p_0 and q_0 are *bisimilar*, denoted $p_0 \equiv_B q_0$ iff there exists a bisimulation relation between p_0 and q_0.

Corresponding processes have to be capable of the same actions leading again to equivalent processes. All the equivalences we defined form an equivalence spectrum, having bisimilarity referring to the finest equivalence and trace equivalence being the coarsest one. All interrelations are depicted in Fig. 2. For a comprehensive overview we refer to van Glabbeek [6]. For common LTS operators, all the branching time equivalences described in this section are congruences, i. e., equivalence of components implies equivalence of composition. In the following sections, we analyze the notions of weak and strong fairness to define notions of *fair language equivalence*. In both cases, we look at the question which of the equivalences, mentioned in this section, also implies fair language equivalence.

3 Strong Fairness

In concurrent and distributed systems, several choices between actions take place with some of them turning up recurrently. As those choices are resolved independently of the current execution, the system may always choose the same action while neglecting the alternatives. Fairness assumptions are incorporated to overcome such an unfair treatment of actions. Thereupon, according notions of fair system languages are defined. The following definition is inspired by the notion of *fairness* as introduced by Reisig [16].

Definition 11 (Strong Fairness)**.** Let $A = (Q, \Sigma, \longrightarrow)$ be an LTS and $\mathscr{F} \subseteq \Sigma$. A run $\rho = p_0 a_1 p_1 a_2 \ldots$ of $p_0 \in Q$ is *strong fair w. r. t.* $\mathscr{F}$ iff for all $a \in \mathscr{F}$, the existence of infinitely many p_i in ρ with $p_i \xrightarrow{a}$ implies that a occurs infinitely often in ρ. The set of all strong fair runs of p_0 w. r. t. $\mathscr{F}$ is denoted by $\Phi_{\mathscr{F}}(p_0)$. The *strong fair language of* p_0 *w. r. t.* $\mathscr{F}$ is defined as $\mathscr{L}^{\Phi}_{\mathscr{F}}(p_0) := \{tr(\rho) \mid \rho \in \Phi_{\mathscr{F}}(p_0)\}$. We call a trace σ strong fair in p_0 w. r. t. $\mathscr{F}$ iff $\sigma \in \mathscr{L}^{\Phi}_{\mathscr{F}}(p_0)$ i. e., there is a run ρ with $tr(\rho) = \sigma$ that is strong fair in p_0 w. r. t. $\mathscr{F}$.

By this definition, finite runs are always strong fair. Please note that the superscript Φ is just an identifier for strong fairness. In the following section, φ will likewise denote weak fairness. Fair languages describe the actually executable system behaviors, as the possibility of unfair treatment of recurrent choices vanishes in case of infinite executions. Subsequently, we identify equivalences within the spectrum presented in Sect. 2 for which equivalence of two systems implies equal strong fair languages. We thereby find an equivalence spectrum for which strong fairness assumptions, as reflected by the choice of $\mathscr{F}$, leave the set of fair runs, and thereby the fair language, of a system invariant. In a system development process, stepwise refinement w. r. t. this spectrum may be used to substitute components by equivalent subsystems without altering the fair behaviors of the subsystems.

As strong fairness considers the resolution of (nondeterministic) choices, a linear time equivalence such as trace equivalence should not suffice to preserve strong fairness. This claim is supported by the example LTS depicted in Fig. 3. The traces of processes p and q are equal, i. e., $Traces(p) = \{a,b\}^* = Traces(q)$ and $Traces^{\infty}(p) = \{a,b\}^{\omega} = Traces^{\infty}(q)$. However, the strong fair language w. r. t. $\mathscr{F} = \{a\}$ of both processes differ since b^{ω} is not fair w. r. t. action a in p. In contrast, q may take the transition leading away from q at some point disabling action a forever and resulting in a fair trace b^{ω}.

Ready equivalence and failures equivalence are among the weakest branching time equivalences. They compare prefixes of possibly infinite runs and their capabilities, i. e., a set of actions being enabled (readies) or disabled (failures) after the respective prefix. But fairness also considers information on actions being enabled during the run corresponding to the mentioned prefix. Furthermore, considering a

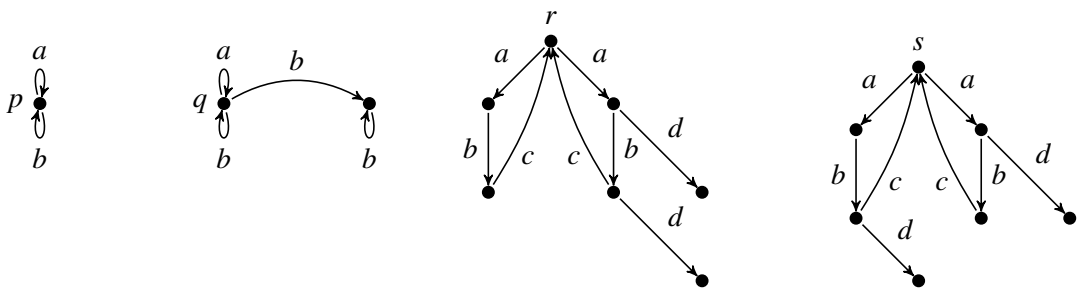

Figure 3: A finite LTS, where $p \equiv_T^\infty q$, but $\mathscr{L}^\Phi_{\{a\}}(p) \neq \mathscr{L}^\Phi_{\{a\}}(q)$ and $r \equiv_R^\infty s$, but $\mathscr{L}^\Phi_{\{d\}}(r) \neq \mathscr{L}^\Phi_{\{d\}}(s)$.

run of a process with two prefixes w_1 and w_2 where w_1 is a prefix of w_2 does not necessarily imply the existence of a unique run with the same two prefixes in a failures/ready equivalent process. For example, processes r and s in Fig. 3 are ready equivalent thus also failures equivalent. Both processes contain two different abc-loops such that action d can be but does not have to be enabled after any a or b throughout the infinite trace $(abc)^\omega$. Furthermore s is forced to enable d once in every abc subtrace while r is not. Therefore d is enabled infinitely often in any run of s corresponding to $(abc)^\omega$ whereas r can use the left loop solemnly, never enabling d. As d is not recurrent in $(abc)^\omega$, this trace is strong fair w. r. t. $\mathscr{F} = \{d\}$ in r but not in s resulting in different fair languages. Thus, respecting branching time does not necessarily imply strong fair language equivalence.

However, failure trace equivalence and ready trace equivalence preserve information on disabledness or enabledness of actions on finite and infinite runs. In contrast to failures and ready equivalence, not only capabilities at the end of prefixes are considered but also in every intermediate step. Having this information also included in infinite runs suffices to restrict equivalent processes to those having the same strong fair language w. r. t. some $\mathscr{F} \subseteq \Sigma$. Our proof incorporates failure trace equivalence, immediately implying the result for stronger equivalences, e. g., ready trace equivalence.

Theorem 12. *Let $A = (Q, \Sigma, \longrightarrow)$ be an LTS and $p_0, q_0 \in Q$ such that $p_0 \equiv_{FT}^\infty q_0$. For any $\mathscr{F} \subseteq \Sigma$, $\mathscr{L}^\Phi_{\mathscr{F}}(p_0) = \mathscr{L}^\Phi_{\mathscr{F}}(q_0)$.*

Proof. Assume, there is some trace $\sigma \in \mathscr{L}^\Phi_{\mathscr{F}}(p_0) \setminus \mathscr{L}^\Phi_{\mathscr{F}}(q_0)$. It holds that $\sigma \in \mathit{Traces}^\infty(q_0)$, as (a) if σ was finite, it would be in $\mathscr{L}^\Phi_{\mathscr{F}}(q_0)$ and (b) from $p_0 \equiv_{FT}^\infty q_0$ we deduce $p_0 \equiv_T^\infty q_0$, implying $\mathit{Traces}^\infty(p_0) = \mathit{Traces}^\infty(q_0)$. As $\sigma \in \mathscr{L}^\Phi_{\mathscr{F}}(p_0)$, there is a strong fair run $\rho_p = p_0 a_1 p_1 a_2 \ldots$ w. r. t. $\mathscr{F}$ such that $tr(\rho_p) = \sigma$. Let σ_p be the maximal failure trace corresponding to ρ_p i. e., the trace that alternates between the action labels of ρ_p and the maximal refusal sets in each state. This σ_p is defined as $\sigma_p = X_0 a_1 X_1 a_2 \ldots$ where $X_i = \{a' \in \Sigma \mid p_i \stackrel{a'}{\not\longrightarrow}\}$. As σ_p corresponds to a strong fair run of p_0, it holds that for each $a \in \mathscr{F}$, (i) label a occurs infinitely often in ρ_p, i. e., $\forall i \in \mathbb{N} : \exists j > i : a_j = a$ or (ii) the number of refusal sets that the label a does not occur in is finite, i. e., $\exists n \in \mathbb{N} : \forall j \geq n : a \in X_j$. From $p_0 \equiv_{FT}^\infty q_0$ we know that σ_p is also a failure trace of q_0. Though σ_p is a maximal failure trace for p_0 it does not have to be maximal for q_0. A corresponding maximal failure trace for q_0 would be $\sigma_q = Y_0 a_1 Y_1 a_2 \ldots$ with refusal sets $Y_i = \{a' \in \Sigma \mid q_i \stackrel{a'}{\not\longrightarrow}\}$ and we get that $X_i \subseteq Y_i$. As $\sigma \notin \mathscr{L}^\Phi_{\mathscr{F}}(q_0)$, every run $\rho_q = q_0 a_1 q_1 a_2 \ldots$ that corresponds to σ_q has to neglect strong fairness for some $a \in \mathscr{F}$. If case (i) holds, ρ_q contains a infinitely often and, by this, respects strong fairness of a. If case (ii) holds, there is an $n \in \mathbb{N}$ such that every refusal set $Y_j = \{a' \in \Sigma \mid q_j \stackrel{a'}{\not\longrightarrow}\}$ with $j > n$ contains a. Otherwise ρ_q would not be a run that corresponds to σ_q. Therefore ρ_q does not neglect fairness of a which contradicts $\sigma \notin \mathscr{L}^\Phi_{\mathscr{F}}(q_0)$. The case of $\sigma \in \mathscr{L}^\Phi_{\mathscr{F}}(q_0) \setminus \mathscr{L}^\Phi_{\mathscr{F}}(p_0)$ is analogous. □

Our result immediately implies that all equivalences $\equiv \subseteq \equiv^{\infty}_{FT}$ preserve strong fairness. This result together with the following counterexamples, concerning ready equivalence, possible futures equivalence and simulation equivalence, indicate a spectrum of strong fairness preserving equivalences (cf. Fig. 2).

As pointed out in the discussion about trace equivalence, processes p and q in Fig. 3 do not share the same strong fair language. However, it holds that $p \equiv^{\infty}_{S} q$ which may be seen by simulation relations containing nothing but the pair (p,q) ((q,p), resp.). Every $p \xrightarrow{a} p$ step can be simulated by a corresponding $q \xrightarrow{a} q$ step and vice versa. The same holds for $p \xrightarrow{b} p$ steps. The b step of q leading away from q and all subsequent b steps can also be simulated by a $p \xrightarrow{b} p$ step. This counterexample lets us conclude that simulation semantics does not preserve fairness.

Figure 4: A finite LTS, where $p \equiv^{\infty}_{PF} q$, but $\mathscr{L}^{\Phi}_{\{c\}}(p) \neq \mathscr{L}^{\Phi}_{\{c\}}(q)$.

The LTS shown in Fig. 4 has two possible futures equivalent processes p and q that do have different fair languages w. r. t. $\mathscr{F} = \{c\}$. Process p can take the left a loop to produce an infinite trace a^{ω} whereas q is not capable of doing infinite a actions without enabling c infinitely often and by this treating c unfair.

4 Weak Fairness

The second characteristic of fairness we discuss is weak fairness. In contrast to strong fairness, it does not suffice for an action to be enabled infinitely often to enforce an infinite number of occurrences in a fair run. In weak fair runs, every action that is enabled infinitely long has to be taken infinitely often. The following definition is inspired by the notion of *progress* as introduced by Reisig [16].

Definition 13 (Weak Fairness)**.** Let $A = (Q, \Sigma, \longrightarrow)$ be an LTS and $\mathscr{F} \subseteq \Sigma$. A run $\rho = p_0 a_1 p_1 a_2 \ldots$ of $p_0 \in Q$ is *weak fair w. r. t.* $\mathscr{F}$ iff for all $a \in \mathscr{F}$, the existence of an $i \geq 0$ with $p_j \xrightarrow{a}$ for all $j \geq i$ implies that a occurs infinitely often in ρ. The set of all weak fair runs of p_0 w. r. t. $\mathscr{F}$ is denoted by $\varphi_{\mathscr{F}}(p_0)$. The *weak fair language of* p_0 *w. r. t.* $\mathscr{F}$ is defined as $\mathscr{L}^{\varphi}_{\mathscr{F}}(p_0) := \{tr(\rho) \mid \rho \in \varphi_{\mathscr{F}}(p_0)\}$. We call a trace σ weak fair in p_0 w. r. t. $\mathscr{F}$ iff $\sigma \in \mathscr{L}^{\varphi}_{\mathscr{F}}(p_0)$ i. e., there is a run ρ with $tr(\rho) = \sigma$ that is weak fair in p_0 w. r. t. $\mathscr{F}$.

Note that finite runs are always weak fair. In case of preservation of weak fairness under ready equivalence we again find a counterexample. The processes p and q depicted in Fig. 5 are ready equivalent but their weak fair languages w. r. t. $\mathscr{F} = \{d,e\}$ differ. The trace $a(bc)^{\omega}$ has no according fair run in p as each run producing this trace would either have d or e enabled in each step following the first a step. In contrast, q alternates between enabling d and e throughout each run. Thus, weak fairness of $\{d,e\}$ is not neglected by q and $\mathscr{L}^{\varphi}_{\{d,e\}}(p) \not\ni a(bc)^{\omega} \in \mathscr{L}^{\varphi}_{\{d,e\}}(q)$.

We may show that failure trace equivalence preserves also weak fairness, leaving us with the same spectrum of weak fairness preserving equivalences as in case of strong fairness.

Theorem 14. *Let* $A = (Q, \Sigma, \longrightarrow)$ *be an LTS and* $p_0, q_0 \in Q$ *such that* $p_0 \equiv^{\infty}_{FT} q_0$*. For any* $\mathscr{F} \subseteq \Sigma$*,* $\mathscr{L}^{\varphi}_{\mathscr{F}}(p_0) = \mathscr{L}^{\varphi}_{\mathscr{F}}(q_0)$.

Figure 5: A finite LTS, where $p \equiv_R^\infty q$, but $\mathscr{L}^\varphi_{\{d,e\}}(p) \neq \mathscr{L}^\varphi_{\{d,e\}}(q)$.

Proof. Assume there is a trace $\sigma \in \mathscr{L}^\varphi_{\mathscr{F}}(p_0) \setminus \mathscr{L}^\varphi_{\mathscr{F}}(q_0)$. It holds that $\sigma \in \mathit{Traces}^\infty(q_0)$, since (a) if σ was finite, it would be in $\mathscr{L}^\varphi_{\mathscr{F}}(p_0)$ and (b) $p_0 \equiv_{FT}^\infty q_0$ implies that $p_0 \equiv_T^\infty q_0$, implying $\mathit{Traces}^\infty(p_0) = \mathit{Traces}^\infty(q_0)$. As $\sigma \in \mathscr{L}^\varphi_{\mathscr{F}}(p_0)$, there is a weak fair run $\rho_p = p_0 a_1 p_1 a_2 \ldots$ w. r. t. $\mathscr{F}$ such that $tr(\rho_p) = \sigma$. Let σ_p be the maximal failure trace corresponding to ρ_p i. e., the trace that alternates between the action labels of ρ_p and the maximal refusal sets in each state. This σ_p is defined as $\sigma_p = X_0 a_1 X_1 a_2 \ldots$ where $X_i = \{a' \in \Sigma \mid p_i \overset{a'}{\not\longrightarrow}\}$. As σ_P corresponds to a weak fair run of p_0, it holds that for each $a \in \mathscr{F}$, (i) the label a occurs infinitely often in ρ_p, i. e., $\forall i \in \mathbb{N} : \exists j > i : a_j = a$, or (ii) the label a occurs in infinitely many refusal sets, i. e., for every n there is $j \geq n$ such that X_j contains a. This can be formalized by $\forall n \in \mathbb{N} : \exists j \geq n : a \in X_j$.

From $p_0 \equiv_{FT}^\infty q_0$ we know that σ_p is also a failure trace of q_0. Though σ_p is a maximal failure trace of p_0 it does not have to a be maximal of q_0. A corresponding maximal failure trace of q_0 would be $\sigma_q = Y_0 a_1 Y_1 a_2 \ldots$ with refusal sets $Y_i = \{a' \in \Sigma \mid q_i \overset{a'}{\not\longrightarrow}\}$ and we get that $X_i \subseteq Y_i$. As $\sigma \notin \mathscr{L}^\varphi_{\mathscr{F}}(q_0)$, every run $\rho_q = q_0 a_1 q_1 a_2 \ldots$ that corresponds to σ_q neglects fairness for some $a \in \mathscr{F}$. If case (i) holds, ρ_q contains a infinitely often and, by this, respects weak fairness of a. In case (ii), for every $n \in \mathbb{N}$ there is a $j \geq n$ such that $Y_j = \{a' \in \Sigma \mid q_j \overset{a'}{\not\longrightarrow}\}$ contains a. This can directly be followed from $X_i \subseteq Y_i$. Otherwise ρ_q would not be a run that corresponds to σ_q. Therefore ρ_q does not neglect weak fairness of a, contradicting $\sigma \notin \mathscr{L}^\varphi_{\mathscr{F}}(q_0)$. The case of $\sigma \in \mathscr{L}^\varphi_{\mathscr{F}}(q_0) \setminus \mathscr{L}^\varphi_{\mathscr{F}}(p_0)$ is analogous. □

The crucial point in proving preservation of weak and strong fairness under failure trace equivalence is the construction of a maximal failure trace that corresponds to a given generic trace. This construct contains sufficient information to decide whether the given generic trace is fair regardless of weak or strong fairness. As those traces are preserved for failure trace equivalent systems, fairness is preserved. The proofs of strong and weak fairness only differ in the description of disallowed enabledness for actions in $\mathscr{F} \subseteq \Sigma$ that are not recurrent. While strong fairness requires the actions to be always eventually disabled, weak fairness needs those actions to be eventually always disabled.

As in Sect. 3, we also discuss the weak fair languages of simulation equivalent as well as possible futures equivalent processes. Regarding simulation equivalence, processes p and q of Fig. 3 again serve as a counterexample, since they are simulation equivalent but have different weak fair languages. This counterexample lets us conclude that simulation semantics preserves neither strong nor weak fairness. Also in case of possible futures we subsequently discuss a counterexample justifying the indicated border of fairness preservation in Fig. 2. The LTS shown in Fig. 6 has two possible futures equivalent processes p and q that do show differences in their weak fair languages w. r. t. $\mathscr{F} = \{b\}$. Process p may follow the infinite branch to produce the infinite trace a^ω thereby respecting weak fairness of action b, whereas every trace a^ω of q eventually enables b in every subsequent step. Each trace of p returning to p (always

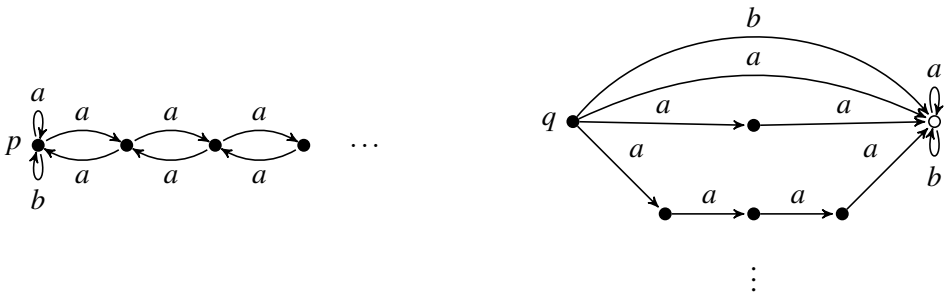

Figure 6: An LTS, where $p \equiv^{\infty}_{PF} q$, but $\mathscr{L}^{\varphi}_{\{b\}}(p) \neq \mathscr{L}^{\varphi}_{\{b\}}(q)$.

after $2n$ steps for some $n \in \mathbb{N}$) is reflected by infinite branching of q into an infinite number of paths of finite lengths of multiples of two, indicated by the three horizontally (process p) or vertically (process q) aligned dots ($\cdots$).

The reason why the processes depicted in Fig. 6 are indistinguishable by possible futures is that the captured futures only account for the traces from a certain state and traces are, by definition, finite. When considering a slightly stronger notion of possible futures, incorporating also infinite traces as possible futures, p and q may be distinguished, possibly altering the decision on weak fairness preservation in general. Further observe that the given processes both have an infinite state space, which is different from any other counterexample given in this paper. We do not give a formal proof but strongly conjecture that for finite-state processes (i. e., for which the set of all reachable processes is finite), possible futures equivalence preserves weak fairness. The idea of the proof is that a weak fair run of a finite-state process p traverses at least one reachable process infinitely many times. Moreover, there must be such a process respecting weak fairness of some action b, i. e., b is disabled and the trace b is no possible future of that process. Such a process must also be reachable from a possible futures equivalent process q, which in turn may be used to construct the same weak fair run as given by p. Since our argument is concerned with an arbitrary infinite weak fair run, the claim holds in general. The counterexample for strong fairness, given by Fig. 4, already employs finite-state processes thus indicating a split of the borders of fairness preservation for finite-state processes.

5 Conclusion

In this paper, we discovered an equivalence spectrum for which fairness assumptions are preserved between equivalent systems. When distinguishing between internal and external moves of a system, as usually done in process-algebraic verification, handling internal actions in a fair way has already been studied for notions of global fairness [1, 15, 14], i. e., where not only a subset of actions and/or components is considered. The main focus of these works is the proper handling of divergent system behavior, manifested in infinite sequences of internal actions, i. e., τ. Thereupon, *Koomen's Fair Abstraction Rule* (KFAR) [10] is taken into account allowing to reduce divergent behavior to a single internal action in process-algebraic settings. The resulting semantic equivalence Bergstra et al. [1] discovered and Puhakka and Valmari extensively studied [15, 14] is called *Chaos-Free Failures Divergences* (CFFD).

There are open questions we plan to address in future work. First, in this work we dealt with equivalence relations not abstracting from internal behavior (i. e., τ-transitions) and, thereupon, also fairness of actions that are not τ. When trying to lift our results to a weak equivalence spectrum [7], it is unclear

what treatment of τ-transitions is preferable. Previous works, as mentioned above, already present a variety of possibilities, each worth investigating. Besides the inclusion of internal actions, as conjectured in the last section, considering only finite state systems may yield even more diversity between the equivalences preserving different styles of fairness.

Furthermore, we would like to explore the conjecture, given in the end of Sect. 4, namely that possible futures equivalence implies weak fair language equivalence in case of finite-state processes. It might be interesting to see how the spectrum of fairness preservation evolves for the various notions of progress in finite-state processes.

Another important aspect for future work is the composition of system behaviors. Some of the equivalences in van Glabbeek's spectrum are congruences for certain composition operators. Congruences allow for a compositional reasoning in the sense that congruent systems show the same overall behavior when plugged into a fixed environment. If for two systems, correctness is proven under the same fairness assumption, it is unclear whether the composed system behavior under fairness only constitutes the fair behaviors of the components.

Figure 7: Two systems p and q and a possible composition (p,q).

Consider for example the systems p and q in Figure 7 under fairness of $\mathscr{F} = \{a\}$. Regardless whether strong or weak fairness is taken into account, the run b^ω is no fair run of process p. But when composed with process q in a manner where transitions with the same action labels have to be done synchronously, the disregard of joined action a becomes a possibility as it is not enabled initially. Therefore, b^ω becomes a fair trace of the composed system whereas it does not occur in any of the fair languages of the components. This example highlights the influence of the used composition mechanism and properties of components on the relation of fair languages of components and composed system.

When considering composition operators like the one of CCS by Milner [11], dealing with internal behavior under fairness is crucial. Puhakka and Valmari [15] studied this subject for CFFD in a general LTS composition framework. Thereby, they limited the notions of fairness in order to obtain a usable abstraction and verification framework. Stronger equivalences as the ones we identified in this paper have not been considered so far. Other works are also concerned directly or indirectly with different fairness notions in process algebra settings [3, 2, 5], e. g., Corradini et al. [2] consider fairness of actions. They obtain a compositional semantics for the process language PAFAS, incorporating a TCSP-parallel operator and ensuring weak fairness by forcing each enabled action to happen eventually.

References

[1] Jan A. Bergstra, Jan W. Klop & Ernst-Rüdiger Olderog (1987): *Failures without Chaos: a Process Semantics for Fair Abstraction*. In M. Wirsing, editor: *Formal Description of Programming Concepts – III*, Lecture Notes in Computer Science, North-Holland, Amsterdam, pp. 77–101.

[2] Flavio Corradini, Maria R. Di Berardini & Walter Vogler (2006): *Fairness of Actions in System Computations*. *Acta Informatica* 43(2), pp. 73–130, doi:10.1007/s00236-006-0011-2.

[3] Gerardo Costa & Colin Stirling (1984): *A fair calculus of communicating systems*. *Acta Informatica* 21(5), pp. 417–441, doi:10.1007/BF00271640.

[4] Edsger W. Dijkstra (1965): *Solution of a problem in concurrent programming control*. *CACM* 8(9), p. 569, doi:10.1145/365559.365617.

[5] Nissim Francez (1986): *Fairness*. Springer-Verlag New York, Inc., New York, NY, USA, doi:10.1007/978-1-4612-4886-6.

[6] Rob J. van Glabbeek (1990): *The linear time - branching time spectrum*. In J. C. M. Baeten & J. W. Klop, editors: *CONCUR '90 Theories of Concurrency: Unification and Extension: Amsterdam, The Netherlands, August 27–30, 1990 Proceedings*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 278–297, doi:10.1007/BFb0039066.

[7] Rob J. van Glabbeek (1993): *The linear time — Branching time spectrum II*. In Eike Best, editor: *CONCUR'93: 4th Intrenational Conference on Concurrency Theory Hildesheim, Germany, August 23–26, 1993 Proceedings*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 66–81, doi:10.1007/3-540-57208-2_6.

[8] Matthew Hennessy & Robin Milner (1985): *Algebraic Laws for Nondeterminism and Concurrency*. *J. ACM* 32(1), pp. 137–161, doi:10.1145/2455.2460.

[9] Donald E. Knuth (1966): *Additional comments on a problem in concurrent programming control*. *Commun. ACM* 9(5), pp. 321–322, doi:10.1145/355592.365595.

[10] Cees J. Koomen (1985): *Algebraic specification and verification of communication protocols*. *Science of Computer Programming* 5, pp. 1 – 36, doi:10.1016/0167-6423(85)90002-4.

[11] Robin Milner (1980): *A Calculus of Communicating Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, doi:10.1007/3-540-10235-3.

[12] Faron Moller & Alexander Rabinovich (1999): *On the expressive power of CTL*. In: *Proceedings. 14th Symposium on Logic in Computer Science (Cat. No. PR00158)*, pp. 360–368, doi:10.1109/LICS.1999.782631.

[13] Gordon David Plotkin (1982): *A powerdomain for countable non-determinism*, pp. 418–428. Springer Berlin Heidelberg, Berlin, Heidelberg, doi:10.1007/BFb0012788.

[14] Antti Puhakka (2005): *Using Fairness Constraints in Process-Algebraic Verification*. In Dang Van Hung & Martin Wirsing, editors: *Theoretical Aspects of Computing – ICTAC 2005: Second International Colloquium, Hanoi, Vietnam, October 17-21, 2005. Proceedings*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 546–561, doi:10.1007/11560647_36.

[15] Antti Puhakka & Antti Valmari (2001): *Liveness and Fairness in Process-Algebraic Verification*. In Kim G. Larsen & Mogens Nielsen, editors: *CONCUR 2001 — Concurrency Theory: 12th International Conference Aalborg, Denmark, August 20–25, 2001 Proceedings*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 202–217, doi:10.1007/3-540-44685-0_14.

[16] Wolfgang Reisig (1998): *Elements of Distributed Algorithms: Modeling and Analysis with Petri Nets*. Springer Berlin Heidelberg, Berlin, Heidelberg, doi:10.1007/978-3-662-03687-7.