A Precongruence Format For Should Testing Preorder

November 14, 2009

Abstract

Should testing preorder was proposed as a liveness-preserving precongruence for a process algebra except for nondeterministic choice operator [22]. However, this precongruce result cannot be generalized to other languages before carefully proved. In this paper, we show that τDes format is a precongruence format for should testing preorder. A precongruence format guarantees the precongruence of given preorder by imposing syntactic restrictions on languages. The τDes format [24] was suggested to be a precongruence format for testing preorder, which is incomparable with should testing preorder on discriminative power. Also, we look into its applications on ACP language.

1 Introduction

When using process algebraic languages to specify distributed systems, a suitable semantic equivalence/preorder is usually necessary for reasoning and analyzing. Various semantic equivalences/preorders have been proposed to be useful in different situations. An equivalence relation is reflexive, symmetric, and transitive, and a preorder relation is reflexive and transitive.

A natural and simple classification is that a given equivalence/preorder may be strong or weak. Their differences mostly exist in the ways of dealing with the internal transitions, which are generally denoted as τ transitions. Strong equivalences/preorders regard τ transitions the same as the observable actions. Weak equivalences/preorders, on the other hand, suppose them unobserved by the outer-world. In this sense, when the given distributed systems are further reactive systems, the weak equivalences/preorders are more suitable than the strong equivalences/preorders. A reactive systems can be seen as a black box, which computes by reacting to the stimuli, e.g., input and output, from its environments, and thus no internal transitions can be witnessed from the outside.

Should testing preorder [22, 7], which is a weak preorder, was proposed to be a solution to the long-standing problem of characterising the coarsest liveness-preserving precongruence with respect to a full (TCSP-inspired) process algebra except for the nondeterministic choice operator [22]. The main difference between should testing preorder and the testing preorder [20] exists in the way dealing with the divergences (τ-loops), which drives them incomparable on ability to differentiate processes. Testing preorder, more specifically must testing preorder, requires that each maximal

run of a testing scenario has a success symbol; On the other hand, should testing preorder requires that at any state of a testing scenario, there exists a potential success symbol in the future. The way should testing preorder dealing with divergences (indeed kind of fairness) resembles with the observation congruence, and thus should testing preorder is strictly coarser than observation congruence.

On the other hand, should testing preorder has also been presented to be practical semantic preorder when specifying the communication protocols [30], since they own a kind of fairness which requires that the internal transitions in a τ-loop may be executed an arbitrary but only finite number of times and the actions after the τ-loop will eventually be enabled [30]. It is trivial that this kind of fairness is important for the communication protocols: no matter how many times transmission of a message may fail, this message can eventually delivered as planned successfully [30].

Though should testing preorder is theoretically important and practically useful as stated above, one thing is not sure that whether it is precongruent in some prescribed process algebraic languages, e.g., CCS [18, 17], CSP [14] or ACP [2], though several operators have been proved precongruent on it in [22]. A preorder $\sqsubseteq$ is further a precongruence in some language $\mathcal{L}$ if and only if, for any context C of n holes in that language, if two set of subprocesses $\{p_1, \ldots, p_n\}$ and $\{q_1, \ldots, q_n\}$ are in preorder relation correspondingly, i.e., $p_i \sqsubseteq q_i$ for all $1 \leq i \leq n$, then their composite processes will also be in preorder relation, i.e., $C(p_1, \ldots, p_n) \sqsubseteq C(q_1, \ldots, q_n)$.

Generally, two ways are possible in dealing with this problem.

The first one is that we verify the precongruence for the operators in that language one by one. This way is adopted by most textbooks introducing some process algebraic languages like CCS and ACP. For example, in CCS, the weak bisimulation is verified to be congruent on parallel composition operator $|$ and prefixing operator $a.$, but not to be congruent on nondeterministic choice $+$. However, making such a verification on all operators is clumsy and tedious. What is worse is that, when the preorder is to be used in other languages, it should be verified from scratch.

The second one is to seek for a precongruence format for the preorders. Precongruence format guarantees the precongruence of its corresponding preorder by imposing syntactic restrictions on the languages. Therefore, the preorder is a precongruence in all languages satisfying its corresponding rule format. Up to now, some rule formats have been presented to meet the equivalences/preorders, for examples, GSOS format [3] and ntyft/nxyft format [12] have been proved to be congruent on strong bisimulation, de Simone [8] format was proved to be congruent on failure equivalence, and so on. The readers are referred to Mousavi, Reniers and Groote [19] for the latest review on the rule formats. The main topic of this paper is to pursue a rule format for the should testing preorder.

Rule format comes from the structural operational semantics (SOS). SOS [21] has been widely used in defining the meanings of the operators in various process algebraic languages. Transition system specifications (TSSs) [13], which borrowed from logic programming, form a theoretical basis for SOS. By imposing some syntactic restrictions on TSSs, one can retrieve so-called rule formats. From a specified rule format, one may deduce some interesting properties. Among these properties, one of the most important ones is whether a preorder/equivalence is a precongruence/congruence for a TSS in this rule format. In the rest of the paper, a TSS is always called a language. Moreover,

a rule format guaranteeing the precongruence of some preorder is also called a precongruence format of that preorder.

In this paper, we will show that τDes-format [24] is a precongruence format for should testing preorder. In [24], τDes-format was proposed as a precongruence format for testing preorder. However, the result of this paper is still interesting enough from the following several aspects. The first is that, though our result implies that they share a same precongruence format, should testing preorder and testing preorder are incomparable in discriminative powers on processes. [[an example?]]

The second is that, our result is based on a broader study on precongruence formats for weak docorated trace preorders [15, 16]. Weak failure preorder and weak readiness preorder, coarser than should testing preorder, are assigned with a different format in which rules with τ-conclusion are not permitted. Weak impossible future preorder and weak possible future preorder, finer than should testing preorder, are assigned with another format in which patience rules for receiving arguments are necessary. The result in this paper falls inbetween them, but with a different proof, because no easy generalization on the previous proof methodology can be made for should testing preorder. We will give a simple review on our main results in Section 7.

τDes format is a subformat of de Simone format [8], whose rules are required to be in the form

$$\frac{\{x_i \xrightarrow{a_i} y_i\}_{i\in I}}{f(x_1,\ldots,x_{ar(f)}) \xrightarrow{a} t}. \tag{1}$$

where a_i and a are all observable actions, ar is a function mapping the operator symbol into its arity, $I \subseteq \{1,\ldots,ar(f)\}$ and the variables x_i and y_i are all distinct and the only variables that occur in the rule. Moreover, the target $t \in \mathbb{T}(\Sigma)$ does not contain variable x_i for $i \in I$ and has no multiple occurrence of variables.

In fact, taking τDes format as a precongruence format for should testing preorder is based on the following three investigations. First, after introducing the τ transitions, we need rules to implement their evolvements. Therefore, the patience rules are added. Patience rules [5] are rules like

$$\frac{x_i \xrightarrow{\tau} x_i'}{f(x_1,\ldots,x_i,\ldots,x_n) \xrightarrow{\tau} f(x_1,\ldots,x_i',\ldots,x_n)} \tag{2}$$

where $1 \le i \le n$. For example, assume process $a|\tau b$ and transition rules of parallel composition operator as follows.

$$\frac{p \xrightarrow{a} p'}{p|q \xrightarrow{a} p'|q}, \frac{q \xrightarrow{a} q'}{p|q \xrightarrow{a} p|q'}, \frac{x \xrightarrow{a} x', y \xrightarrow{b} y'}{x|y \xrightarrow{\tau} x'|y'}(a,b) \in f$$

Then, patience rule $\frac{q \xrightarrow{\tau} q'}{p|q \xrightarrow{\tau} p|q'}$ will be applied before the communication between subprocesses a and b.

Second, not all arguments of a given operator f need patience rules. This arouses a division on the arguments, i.e., active arguments, receiving arguments and other arguments. We will show that rule formats for should testing preorder only needs the patience rules for active arguments. However, it is not a general requirement, since as we stated before, weak impossible future preorder and weak possible future preorder, need also patience rules for receiving arguments.

Third, should testing preorder is invariant under hiding operator, or more generally rules with τ-conclusion. Though this is the same story with testing preorder, it is still not a general case for weak decorated trace preorders, since weak readiness preorder and weak failure preorder might not be preserved.

The structure of this paper is: in Section 2, we will introduce some necessary knowledge for process algebraic languages and SOS. Then in Section 3, the formal definition of the should testing preorder is presented. An intuitive motivation on our opinions will be exhibited with examples in Section 4. Section 5 is devoted to prove the precongruence theorem which serves as the main result of this paper. In Section 6, its application on the ACP language will be shown. Section 7 will compare the result in the paper with our previous works. And then, in Section 8, we will conclude the paper.

2 Preliminaries

For clarity, we divide the preliminaries into several subsections. The first three subsections introduce some general knowledge about process algebraic languages, i.e., syntax, semantic model and semantic equivalences. The fourth subsection defines, in a given language, what makes a preorder be precongruent. Finally, we will define a division on the arguments of an operator, i.e., active arguments, receiving arguments and other arguments.

2.1 Syntax

Let Act denote a set of names which will be used to label on events and Act^* be the set of all action sequences. We use $a, b, \ldots$ to range over actions in Act, and use $A, B, \ldots$ to range over sets of actions in Act. τ denotes the internal actions which can not be observed by the outer world. $\alpha, \beta, \ldots$ range over actions in $Act \cup \{\tau\}$, $\delta, \mu, \sigma, \ldots$ range over sequences of actions, and $\Phi, \Psi, \ldots$ range over sets of sequences. Besides, $\varepsilon = \tau^*$.

Basically, presenting a set of syntactic constructions is the first step to define a process algebraic language, e.g., CCS, CSP and ACP.

Definition 2.1.1 *[1] Let $V = \{x_1, x_2, \ldots\}$ be a set of variables. A signature Σ is a collection of function symbols $f \notin V$ equipped with a function $ar : \Sigma \to N$. The set $\mathbb{T}(\Sigma)$ of terms over a signature Σ is defined recursively by: 1) $V \subseteq \mathbb{T}(\Sigma)$; 2) if $f \in \Sigma$ and $t_1, \ldots, t_{ar(f)} \in \mathbb{T}(\Sigma)$, then $f(t_1, \ldots, t_{ar(f)}) \in \mathbb{T}(\Sigma)$.*

A term $c()$ is abbreviated as c. For $t \in \mathbb{T}(\Sigma)$, $var(t)$ denotes the set of variables that occur in t. $\mathrm{T}(\Sigma)$ is the set of closed terms over Σ, i.e., the terms $p \in \mathbb{T}(\Sigma)$ with $var(p) = \emptyset$. A Σ-substitution ζ is a mapping from V to $\mathbb{T}(\Sigma)$, and a closed Σ-substitution ζ is a mapping from V to $\mathrm{T}(\Sigma)$.

In the paper, we will use $p, q, \ldots$ to range over the closed terms, and call them processes. Here, we give the syntax of a simple process algebraic language **B** as follows

$$p ::= \surd \mid a \cdot p \mid p \boxplus p \mid p \oplus p \mid p \rhd p$$

where the operators $\boxplus$ and $\oplus$ are exactly the internal and external choices of CSP and $\surd$ denotes the successful termination.

2.2 Structural Operational Semantics and Labeled Transition Systems

After composing a process using syntactic constructions, we need to define its semantics for analyzing and reasoning. It is well known that several kinds of semantics are possible, e.g., operational semantics, denotational semantics and axiomatic semantics.

SOS has been widely accepted as a tool to define operational semantics of processes. TSSs are a formalization of SOS [21]. The readers are referred to [1] for a comprehensive review on SOS.

Definition 2.2.1 *A positive Σ-literal is an expression $t \xrightarrow{\alpha} t'$ and a negative Σ-literal is an expression $t \xrightarrow{\alpha} \!\!\!\!\!\rightarrow$, where $t, t' \in \mathbb{T}(\Sigma)$ and $\alpha \in Act \cup \{\tau\}$. A transition rule over Σ is an expression of the form $\frac{H}{C}$, where H a set of Σ-literals (the premises of the rule) and C a positive Σ-literal (the conclusion). The left- and right-hand side of C are called the source and the target of the rule, respectively. Moreover, if $r = \frac{H}{t \xrightarrow{\alpha} t'}$ then define $ante(r) = H$, $cons(r) = \{t \xrightarrow{\alpha} t'\}$, and the output of r as α.*

A TSS, written as (Σ, Ψ), consists of a signature Σ and a set Ψ of transition rules over Σ. A TSS is positive if the premises of its rules are positive.

Following it, the labeled transition systems (LTSs) are to be defined. LTSs are standard semantic models for various process algebraic languages, and in fact, each process has an equivalent LTS by the help of the transition rules.

Definition 2.2.2 *Let Σ be a signature. A transition relation over Σ is a relation $Tr \subseteq T(\Sigma) \times Act \cup \{\tau\} \times T(\Sigma)$. Element (p, α, p') of a transition relation is written as $p \xrightarrow{\alpha} p'$.*

Thus a transition relation over Σ can be regarded as a set of closed positive Σ-literals (transitions).

Definition 2.2.3 *A labeled transition system (LTS) is a triple $(T, Tr, Act \cup \{\tau\})$, where T is the set of processes, i.e., the set of closed terms, and Tr is the transition relation defined as above.*

As an example, we give the transition rules of the language **B** in Table 1.

2.3 Semantic Equivalences and Semantic Preorders

After providing processes with LTSs as their semantics, a natural topic is to decide whether two processes with different syntactic expressions are equivalent. Certainly, two processes with isomorphic LTSs should be deemed to be equivalent since it is virtually never needed to distinguish between isomorphic graphs [27]. However, equivalence with isomorphic LTSs is generally too finer to be practically useful. Therefore, various semantic equivalences coarser

Table 1: Transition rule of language **B**

$$\frac{}{a \xrightarrow{a} \surd} \qquad \frac{}{a \cdot p \xrightarrow{a} p}$$

$$\frac{p \xrightarrow{a} p'}{p \boxplus q \xrightarrow{a} p'} \quad \frac{q \xrightarrow{a} q'}{p \boxplus q \xrightarrow{a} q'} \quad \frac{p \xrightarrow{\tau} p'}{p \boxplus q \xrightarrow{\tau} p' \boxplus q} \quad \frac{q \xrightarrow{\tau} q'}{p \boxplus q \xrightarrow{\tau} p \boxplus q'}$$

$$\frac{}{p \oplus q \xrightarrow{\tau} p} \quad \frac{}{p \oplus q \xrightarrow{\tau} q}$$

$$\frac{p \xrightarrow{a} p'}{p \rhd q \xrightarrow{a} p'} \quad \frac{p \xrightarrow{\tau} p'}{p \rhd q \xrightarrow{\tau} p' \rhd q} \quad \frac{}{p \rhd q \xrightarrow{\tau} q}$$

than tree equivalence are presented for different aims. The readers are referred to [27, 28] for comprehensive reviews on semantic equivalences.

As stated in the introduction, semantic equivalences can be classified as strong equivalences and weak equivalences by their different treatments on internal transitions. A common characterization of the equivalences is that, any semantics of a process p can be characterized denotationally by a function $\mathcal{O}(p)$, which constitute the observable behaviors of p [27]. Then the equivalence $\sim_{\mathcal{O}}$ is defined as $p \sim_{\mathcal{O}} q \Longleftrightarrow \mathcal{O}(p) = \mathcal{O}(q)$.

As is well known that, semantic equivalences are generally equivalence relations, i.e., they are reflexive, symmetric and transitive. If the symmetric condition is relaxed, a corresponding preorder relation is obtained. Therefore, for a given semantic equivalence $\sim_{\mathcal{O}}$, there exists its corresponding preorder $\sqsubseteq_{\mathcal{O}}$ such that $p \sim_{\mathcal{O}} q \equiv p \sqsubseteq_{\mathcal{O}} q \wedge q \sqsubseteq_{\mathcal{O}} p$. Using the above function $\mathcal{O}(p)$, the preorder $\sqsubseteq_{\mathcal{O}}$ can be defined by $p \sqsubseteq_{\mathcal{O}} q \Longleftrightarrow \mathcal{O}(p) \subseteq \mathcal{O}(q)$.

Here, as an example, we present the definition of weak trace equivalence/preorder. For an action sequence $\delta = \alpha_1 \ldots \alpha_n$, if there exist $p, p_1, \ldots, p_n \in \mathrm{T}(\Sigma)$ such that $p \xrightarrow{\alpha_1} p_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} p_n$, then we call δ a trace of p, denoted as $p \xrightarrow{\delta}$ or $p \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n}$. As for weak semantics, the weak transition relations and the weak traces are defined as follows. We write $p \overset{a}{\Longrightarrow}$ iff $p \xrightarrow{\tau^*} \xrightarrow{a} \xrightarrow{\tau^*}$, where τ^* denotes an arbitrary number of internal transitions. Hence, for an observable action sequence $\delta = a_1 \ldots a_n$, $p \overset{\delta}{\Longrightarrow}$ iff $p \overset{a_1}{\Longrightarrow} \ldots \overset{a_n}{\Longrightarrow}$.

Definition 2.3.1 *Let p, q be two processes and set $\mathcal{T}(p) = \{\delta \in Act^* | p \overset{\delta}{\Longrightarrow}\}$ be the set of all weak trace of p. Then, p and q are weak trace equivalent, denoted as $p \sim_t q$, iff $\mathcal{T}(p) = \mathcal{T}(q)$. Likewise, p and q are in weak trace preorder, denoted as $p \sqsubseteq_t q$, iff $\mathcal{T}(p) \subseteq \mathcal{T}(q)$.*

Though, in this paper, we focus on the semantic preorder, the result can be safely extended to its corresponding semantic equivalence.

2.4 Precongruence Format

Given a preorder, one of the most frequently-asked questions is whether or not it can be preserved under some frequently-used operators, such as prefixing, choice, parallel composition, etc., in classical process algebraic languages like CCS [18], CSP [14] and ACP [2]. For example, in language **B**, if $p_1 \sqsubseteq_t q_1$ and $p_2 \sqsubseteq_t q_2$, then one might need to know whether $p_1 \boxplus p_2 \sqsubseteq_t q_1 \boxplus q_2$.

Definition 2.4.1 *Let Σ be a signature. A context $C(x_1, \ldots, x_n)$ of n holes over Σ is simply a term in $\mathbb{T}(\Sigma)$ in which n variables occur, each variable only once. If $t_1, \ldots, t_n$ are terms over Σ, then $C(t_1, \ldots, t_n)$ denotes the term obtained by substituting t_1 for the first variable occurring in C, t_2 for the second variable occurring, etc. If $x_1, \ldots, x_n$ are all different variables, then $C(x_1, \ldots, x_n)$ denotes a context of n holes in which x_i is the ith occurring variable.*

In the following, we give the definition on the precongruence of a preorder in a language.

Definition 2.4.2 *Let $\mathcal{L} = (\Sigma, \Psi)$ be a language. A semantic preorder $\sqsubseteq_O$ is precongruent in language $\mathcal{L}$ iff $\forall 1 \leq i \leq n : p_i \sqsubseteq_O q_i \Longrightarrow C(p_1, \ldots, p_n) \sqsubseteq_O C(q_1, \ldots, q_n)$ for any context $C(x_1, \ldots, x_n)$ of n holes in language $\mathcal{L}$, where p_i and q_i are closed terms, i.e., processes, over Σ.*

A precongruence format is to make restrictions on the syntax and transition rules to guarantee that, if a language satisfies this format, then the given preorder is precongruent in it.

2.5 Division on Arguments of an Operator

In a given language $\mathcal{L} = (\Sigma, \Psi)$, the arguments of an operator $f \in \Sigma$ may be divided into three classes as follows.

Definition 2.5.1 *[5, 9, 29] An argument $i \in N$ of an operator f is active if f has a rule in which x_i appears as left-hand side of a premise. A variable x occurring in a term t is receiving in t if t is the target of a rule in which x is the right-hand side of a premise. An argument $i \in N$ of an operator f is receiving if a variable x is receiving in a term t that has a subterm $f(t_1, \ldots, t_n)$ with x occurring in t_i.*

Then, the set of all arguments Arg of an operator can be divided into three classes: active arguments Arg_a, receiving arguments Arg_r and others Arg_o. Therefore, $Arg = Arg_a \cup Arg_r \cup Arg_o$.

Observe that, an argument can be an active argument and a receiving argument at the same time. Here, we make a clear-cut on them and only call those arguments, which are both active arguments and receiving arguments, active arguments. More formally, let $Arg_r := Arg_r \backslash Arg_a$. Therefore, the receiving arguments, from now on, denote only those arguments which are receiving arguments but not active arguments of Definition 2.5.1. Finally, Arg_a, Arg_r and Arg_o will not have intersections.

The reason we make a clear-cut between active arguments and receiving arguments of an operator is that, it will be more clear for our discussion in the following sections when it goes to the unnecessity of introducing patience rules for receiving arguments into a language to make it precongruent under should testing preorder.

Return to the transition rules shown in Table 1. Operator $\boxplus$ has both its two arguments as active arguments. However, the right-hand-side argument of operator $\triangleright$ is neither an active argument nor a receiving argument. It should be classified as other argument.

For the case of receiving arguments, let's see a language by adding two operators f, g and two transition rules r_1, r_2 into language **B**, where $r_1 = \dfrac{x_1 \xrightarrow{c_1} x_1', x_2 \xrightarrow{c_2} x_2'}{f(x_1, x_2) \xrightarrow{a_1} g(x_1', x_2')}$, $r_2 = \dfrac{x_1 \xrightarrow{b_1} x_1'}{g(x_1, x_2) \xrightarrow{a_2} h(x_2)}$. Note that, the second argument of operator g has its second argument as a receiving argument.

Finally, patience rules of an operator, which have been defined in the introduction, can also be divided into three classes, because they are defined in accordance with the arguments of the operator. Therefore, there exist three classes of patience rules, i.e., patience rules for receiving arguments, patience rules for receiving arguments, and patience rules for other arguments. Again, these three classes of patience rules have no intersections.

3 Formal Definitions Of Should Testing Preorder

Should testing preorder was firstly proposed in [7], and then was extensively studied in [22]. It is a semantic preorder based on testing theory [20]. In this paper, we work with its equivalent denotational characterization, since it is more straight to retrieve a precongruence format from its denotational characterization.

First of all, the definition of impossible future pair, F^+ pair in [22], is introduced.

Definition 3.0.2 *$(\sigma, \Phi) \in Act^* \times \mathcal{P}(Act^+)$ is an impossible future pair of process p iff there exists some p' such that $p \overset{\sigma}{\Longrightarrow} p' \wedge \Phi \cap \mathcal{T}(p') = \emptyset$, where $\mathcal{T}(p') = \{\delta \in Act^* \mid p' \overset{\delta}{\Longrightarrow}\}$. The set of all impossible future pairs of process p is called impossible future of p, denoted by $\mathcal{I}(p)$.*

Note that, in the above definition, Φ does not contain ε since $\Phi \in \mathcal{P}(Act^+)$.

Besides, some other definitions are needed.

1. $\delta \preceq \sigma$ iff δ is a prefix of σ.
2. If $\delta \preceq \sigma$, then $\delta^{-1}\sigma = \mu$ such that $\delta\mu = \sigma$. Therefore, $\delta^{-1}\delta = \varepsilon$.
3. $\delta \in\downarrow \Phi$ iff $\exists \sigma \in \Phi : \delta \preceq \sigma$.
4. $\delta^{-1}\Phi = \{\delta^{-1}\sigma | \sigma \in \Phi\}$. Therefore, if $\delta \in \Phi$, then $\varepsilon \in \delta^{-1}\Phi$. To make it complete, we define that $\delta^{-1}\Phi = \{\varepsilon\}$ if $\nexists \sigma \in \Phi : \delta \preceq \sigma$. Note that, in its original definition in [7, 22], δ^{-1} is not defined as a complete function like that. However, it will be useful in the following technical parts to define it as a complete function.

Definition 3.0.3 *Let p and q be two processes. p and q are in should testing preorder relation, denoted as $p \sqsubseteq_{st} q$, iff $\forall (\sigma, \Phi) \in Act^* \times \mathcal{P}(Act^+) : (\sigma, \Phi) \in \mathcal{I}(p) \Longrightarrow \exists \mu \in \{\varepsilon\} \cup \downarrow \Phi - \Phi : (\sigma\mu, \mu^{-1}\Phi) \in \mathcal{I}(q)$.*

In its original definition in [22], it only requires that $\mu \in \{\varepsilon\} \cup \downarrow \Phi$. However, as also stated in [22], if μ is in Φ then $\mu^{-1}\Phi$ will contain ε, which contradicts with the definition of impossible future pair. Therefore, we integrate this implicit requirement, i.e., $\mu \notin \Phi$, into the definition.

In fact, for convenience, this definition can be restated as another one with simpler description as follows.

Definition 3.0.4 *Let p and q be two processes. p and q are in should testing preorder relation, denoted as $p \sqsubseteq_{st} q$, iff* $\forall(\sigma, \Phi) \in Act^* \times \mathcal{P}(Act^+) : (\sigma, \Phi) \in \mathcal{I}(p) \Longrightarrow \exists \mu \in Act^* : (\sigma\mu, \mu^{-1}\Phi) \in \mathcal{I}(q) \wedge \varepsilon \notin \mu^{-1}\Phi$.

Proposition 3.0.1 *Definition 3.0.3 is equivalent to Definition 3.0.4.*

Proof. It is enough to show that $\mu \in \{\varepsilon\} \cup \downarrow \Phi - \Phi$ is equivalent to $\varepsilon \notin \mu^{-1}\Phi$.

($\Longrightarrow$) If $\mu \in \{\varepsilon\} \cup \downarrow \Phi - \Phi$, then $\mu \notin \Phi$. Therefore, it is trivially true that $\varepsilon \notin \mu^{-1}\Phi$.

($\Longleftarrow$) If $\varepsilon \notin \mu^{-1}\Phi$, then μ can not be in Φ and, by the complete definition of $\mu^{-1}\Phi$, there must exist some $\sigma \in \Phi$ such that $\mu \preceq \sigma$. Therefore, $\mu \in \{\varepsilon\} \cup \downarrow \Phi - \Phi$. □

To make clear the above definitions, we present two examples, which will be further discussed in the next section.

1. A simple case. See p_2 and q_2 in Figure 1. It can be verified that $p_2 \sqsubseteq_{st} q_2$. For example, $(c_2, \{b_3\}) \in \mathcal{I}(p_2)$ since, after p_2 evolves into p_2', $\mathcal{T}(p_2') \cap \{b_3\} = \{b_2\} \cap \{b_3\} = \emptyset$. Also, $(c_2, \{b_3\}) \in \mathcal{I}(q_2)$ since after q_2 evolves into q_2', $\mathcal{T}(q_2') \cap \{b_3\} = \{b_2\} \cap \{b_3\} = \emptyset$.

2. A more complex case. Recall the definition of f at the end of Section 2.5. See $f(p_1, p_2)$ and $f(q_1, q_2)$ in the leftmost two graphs of Figure 2. It can be easily verified that $(a_1, \{a_2b_3\}) \in \mathcal{I}(f(p_1, p_2))$. On the other hand, though $(a_1, \{a_2b_3\}) \notin \mathcal{I}(f(q_1, q_2))$, we have $a_2 \in \{\varepsilon\} \cup \downarrow \{a_2b_3\} - \{a_2b_3\}$ and $(a_1a_2, \{b_3\}) \in \mathcal{I}(f(q_1, q_2))$. In fact, $f(p_1, p_2) \sqsubseteq_{st} f(q_1, q_2)$.

4 Intuitive Motivations On Precongruence Formats

This section gives representative examples to show some intuitions on obtaining a precongruence format for should testing preorder. It should be noted that, in this section, we are mainly concerned with the intuitive motivations. The results retrieved in this section will be formally defined and proved in the next section. Also, as the starting point, we assume the basic language **B** which has been introduced as an example in section 2.

4.1 Patience Rules for Non-Active Arguments

Adding rules $r_1 = \dfrac{x_1 \xrightarrow{c_1} x_1', x_2 \xrightarrow{c_2} x_2'}{f(x_1, x_2) \xrightarrow{a_1} g(x_1', x_2')}$, $r_2 = \dfrac{x_1 \xrightarrow{b_1} x_1'}{g(x_1, x_2) \xrightarrow{a_2} h(x_2)}$, $r_3 = \dfrac{x_1 \xrightarrow{b_3} x_1'}{h(x_1) \xrightarrow{b_3} \surd}$, $r_4 = \dfrac{x_1 \xrightarrow{b_2} x_1'}{h(x_1) \xrightarrow{b_2} \surd}$ and their associated patience rules for active arguments into language **B**, we get a new language named $\mathbf{B}_1$.

By Definition 2.5.1, to decide the division of some argument of a given operator, we need to take into consideration all transition rules in which the operator appears. Operator f only appears in rule r_1, where both of its arguments occur

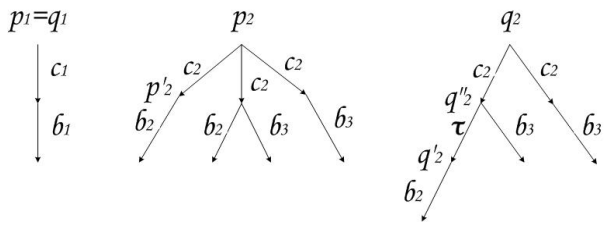

Figure 1: Before composition using the operator f, p_1 and q_1, p_2 and q_2 are in should testing preorder, respectively.

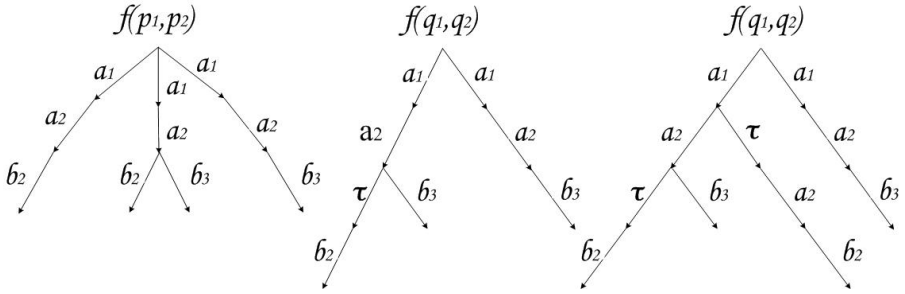

Figure 2: $f(p_1, q_1)$ and $f(p_2, q_2)$ are in should testing preorder even no patience rules for receiving arguments are in the language.

in the left-hand side of the premises and therefore are active arguments. Operator g appears in both rule r_1 and r_2. Its first argument x_1 is an active argument by rule r_2 and its second argument x_2 is a receiving argument by rule r_1.

In Figure 1, it can be easily verified that $p_1 \sqsubseteq_{st} q_1$ and $p_2 \sqsubseteq_{st} q_2$. Now, if further add patience rules for the second argument of $g(x_1, x_2)$ into language $\mathbf{B}_1$ and obtain another language $\mathbf{B}_2$, then we have $f(p_1, p_2) \sqsubseteq_{st} f(q_1, q_2)$ as shown by the leftmost and the rightmost graphs in Figure 2. However, $f(p_1, p_2)$ has already been in should testing preorder with $f(q_1, q_2)$ even in language $\mathbf{B}_1$ as shown by the two leftmost graphs in Figure 2.

Look into their differences in the two graphs for $f(q_1, q_2)$ under languages $\mathbf{B}_1$ and $\mathbf{B}_2$, we find that, the function of the patience rules for receiving arguments is that, without them, the τ transitions may be postponed for a bounded number of observable actions, and this will make some branches of the labeled transition systems merge together. Besides the showcase by example as above, we will prove in the next section that, this function does not break the should testing preorder, though it might break some other preorders, e.g., impossible future preorder and possible future preorder.

4.2 Negative Premises are Excluded

In fact, negative premises are hard to present in the preconguence formats for weak preorders. This can be witnessed by means of a simple example shown in Figure 3: p_2 and q_2 satisfy almost all weak equivalences, including weak

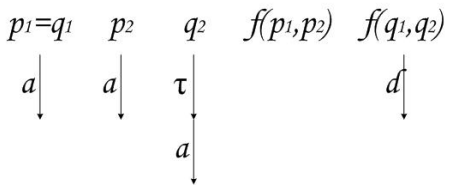

Figure 3: negative premises are not allowed in congruence formats for weak equivalences

bisimulation and testing equivalence. However, after adding rule $r = \dfrac{x_1 \xrightarrow{a} x_1', x_2 \overset{a}{\nrightarrow}}{f(x_1, x_2) \xrightarrow{d} g(x_1', x_2)}$ into language **B**, $f(p_1, p_2)$ and $f(q_1, q_2)$ are even not in weak trace equivalence.

Based on this observation, it is reasonable to consider about the subformats of de Simone format, in which negative premises are excluded by definition.

5 Rule Formats

After intuitive examinations on the necessity or un-necessity of introducing some ingredients into a precongruence format for should testing preorder, we will, in this section, prove that τDes-format of [24] is good candidate for it.

5.1 Definition of τDes-Format

As stated in the introduction, the de Simone language is employed as the starting point for a rule format for should testing preorder.

Definition 5.1.1 *[24] A de Simone language $\mathcal{L}$ is in τDes-format if*

1. *patience rules are the only rules with τ-premises, and*
2. *patience rules for active arguments are all necessary.*

In the following, a language is called τDes-language if it is in τDes-format. As an example, observe that language **B** is a τDes-language since all transition rules in Table 1 are de Simone rules, all rules with τ-premises are patience rules, and each active argument of the operators has a corresponding patience rule.

5.2 Ruloids And Ruloid Theorem On τDes-Format

Ruloids and the ruloid theorem originated from the works of Bloom [5, 4] for GSOS format. In this subsection, we will introduce the ruloids and the ruloid theorem for τDes-format. The ruloid theorem will be useful for proving the precongruence theorem in the following subsections.

For a τDes-language $\mathcal{L} = (\Sigma, \Psi)$, the ruloids $\mathscr{R}(C, \alpha)$, for a context C of n holes and an action α, are a set of expressions like the transition rules:

$$\frac{\{x_i \xrightarrow{\alpha_i} x_i'\}_{i\in I}}{C(x_1, \ldots, x_n) \xrightarrow{\alpha} D(y_1, \ldots, y_n)} \tag{3}$$

such that $y_i \equiv x_i'$ for $i \in I$ and $y_i \equiv x_i$ for $i \notin I$, where $I \subseteq \{1, 2, \ldots, n\}$. These expressions characterize all possible behaviors of the context C in the language. Besides, let $\mathscr{R}(C)$ denote the set of all ruloids of the context C of n holes, i.e., $\mathscr{R}(C) = \bigcup\{\mathscr{R}(C, \alpha) | \alpha \in Act \cup \{\tau\}\}$. It should be noted that context D does not need to have exactly n holes. In fact, after leaving out the copying operation in the de Simone format (the τDes-format is a subformat of the de Simone format), the number of the holes of D should be less than or equivalent to n. But for convenience, in form (3), we still write it as $D(y_1, \ldots, y_n)$.

Furthermore, two properties should be imposed on $\mathscr{R}(C, \alpha)$. We call them soundness property and completeness property, by a bit of abusing the terminologies.

Definition 5.2.1 *Let $\mathcal{L} = (\Sigma, \Psi)$ be a τDes-language, and $C(x_1, \ldots, x_n)$ be any context of n holes in $\mathcal{L}$. A set $\mathscr{R}(C, \alpha)$ of ruloids of form (3) are ruloids of context C and action α, with $\alpha \in Act \cup \{\tau\}$, iff*

- *Soundness. Let $r \in \mathscr{R}(C, \alpha)$ be a ruloid of form (3). If ζ is a closed Σ-substitution such that $\zeta(x_i) \xrightarrow{\alpha_i} \zeta(x_i')$ for all $i \in I$, then there must exist a context D such that $\zeta(C(x_1, \ldots, x_n)) \xrightarrow{\alpha} \zeta(D(y_1, \ldots, y_n))$.*
- *Completeness. Let ζ be any closed Σ-substitution. If $\zeta(C(x_1, \ldots, x_n)) \xrightarrow{\alpha}$, then there must exist a ruloid r of form (3) in ruloids $\mathscr{R}(C, \alpha)$, and $\zeta(x_i) \xrightarrow{\alpha_i}$ for all $i \in I$.*

Below, we will present a strategy to retrieve the ruloids of context C and action α, and then prove that the obtained ruloids satisfy the above two properties, which form the ruloid theorem.

Strategy 5.2.1 *Let $\mathcal{L} = (\Sigma, \Psi)$ be a τDes-language. $C(x_1, \ldots, x_n)$ is any context of n holes in $\mathcal{L}$ and $\alpha \in Act \cup \{\tau\}$ is an action.*

1. *If $C \in V$, i.e., C is a variable, then let $\mathscr{R}(C, \alpha) = \{\frac{x \xrightarrow{\alpha} x'}{x \xrightarrow{\alpha} x'}\}$.*
2. *If $C = f(x_1, \ldots, x_n)$ with $f \in \Sigma$ and $ar(f) = n$, then let $\mathscr{R}(C, \alpha) = (f, \alpha)$, where (f, α) denotes the set of all rules in Ψ whose sources are $f(x_1, \ldots, x_n)$ and outputs are α.*
3. *If C is any context. We can rewrite $C(x_1, \ldots, x_n)$ as $f(C_1(X_1), \ldots, C_m(X_m))$, where $f \in \Sigma$ and $ar(f) = m$. Note that $X_i \cap X_j = \emptyset$ with $1 \leq i, j \leq m$ and $i \neq j$. Without loss of generality, we may suppose that $X_i = x_{i1}x_{i2}\ldots x_{im_i}$ for C_i is a context of m_i holes. Now, let r be any ruloid of form (3) in (f, α) and $\mathscr{R}(C_i, \alpha_i)$ be ruloids of context C_i and action α_i retrieved by induction on this strategy. Then, let $\mathscr{R}(C, \alpha)$ contain all possible ruloids which can be obtained by the following steps:*
   (a) *randomly pick out from $\mathscr{R}(C_i, \alpha_i)$ a rule r_i, for all $i \in I$;*

(b) substitute the variables x_j in r_i with x_{ij}, for all $1 \leq j \leq m_i$;

(c) substitute $x_i \xrightarrow{\alpha_i} x_i'$ in the premise of r with $ante(r_i)$, for all $i \in I$.

Theorem 5.2.1 *Let $\mathcal{L} = (\Sigma, \Psi)$ be a τDes-language, and $C(x_1, \ldots, x_n)$ be any context of n holes in $\mathcal{L}$. The set of ruloids $\mathscr{R}(C, \alpha)$ obtained from the Strategy 5.2.1 is the ruloids of context C and action α with $\alpha \in Act \cup \{\tau\}$.*

Proof. Firstly, the obtained ruloids $\mathscr{R}(C, \alpha)$ of context C and action α are all in form (3). This can be easily obtained by the construction procedure in the Strategy 5.2.1.

Secondly, the obtained ruloids $\mathscr{R}(C, \alpha)$ of context C and action α satisfy the soundness property. Let $r \in \mathscr{R}(C, \alpha)$ be a ruloid of form (3), where C is a context of n holes and $\alpha \in Act \cup \{\tau\}$ is an action. ζ is a closed Σ-substitution such that $\zeta(x_i) \xrightarrow{\alpha_i} \zeta(x_i')$ for all $i \in I$.

i) if $C \in V$, then, without loss of generality, suppose $C = x$. The soundness property is trivially true from $\mathscr{R}(C, \alpha) = \{\frac{x \xrightarrow{\alpha} x'}{x \xrightarrow{\alpha} x'}\}$;

ii) if $C = f(x_1, \ldots, x_n)$, then $\mathscr{R}(C, \alpha) = (f, \alpha)$. Therefore, the soundness property is guaranteed by the transition rules;

iii) if C is any context of n holes, then by Strategy 5.2.1, $C(x_1, \ldots, x_n)$ can be rewritten as $f(C_1(X_1), \ldots, C_m(X_m))$ for some operator $f \in \Sigma$ and $ar(f) = m$, and $ante(r)$ consist of $ante(r_1), \ldots, ante(r_m)$, where $r_i \in \mathscr{R}(C_i, \alpha_i)$ for all $1 \leq i \leq m$. By the assumption of the soundness property that $ante(r)$ is enabled in closed Σ-substitution ζ. Therefore, by the induction hypothesis, $cons(r_1), \ldots, cons(r_m)$ are all enabled in ζ. This means that $\zeta(C_i(X_i)) \xrightarrow{\alpha_i} \zeta(D_i(Y_i))$ for all $1 \leq i \leq m$. In fact, $cons(r_1), \ldots, cons(r_m)$ constitute $ante(f)$. Still by the induction hypothesis on operator f, the transition rules in (f, α) guarantee the enableness of $C(x_1, \ldots, x_n) \xrightarrow{\alpha}$.

Finally, the obtained ruloids $\mathscr{R}(C, \alpha)$ of context C and action α satisfy the completeness property, which can also be easily obtained from the construction procedure of Strategy 5.2.1. □

As we can see that, for a ruloid of form (3), its premises need not include all x_i for $1 \leq i \leq n$. However, we can add $x_i \xrightarrow{\epsilon} x_i'$, for all $i \in \{1, \ldots, n\} \backslash I$, into the premises, as shown in the form (4).

$$\frac{\{x_i \xrightarrow{\alpha_i} x_i'\}_{i \in I} \{x_i \xrightarrow{\epsilon} x_i'\}_{i \in \{1, \ldots, n\} - I}}{C(x_1, \ldots, x_n) \xrightarrow{\alpha} D(y_1, \ldots, y_n)} \tag{4}$$

In this case, $\zeta(x_i) \xrightarrow{\epsilon} \zeta(x_i')$ denotes that subprocess $\zeta(x_i)$ executes no transition. In other word, if $x_i \xrightarrow{\epsilon} x_i'$ is an ϵ transition in a ruloid r then when r is applied with some closed Σ-substitution ζ, subprocess $\zeta(x_i)$ is not fired at all. In this sense, form (4) and form (3) provide same function when applying any closed Σ-substitution ζ.

Note that, the ϵ transitions will not be added to the TSS. In fact, ruloids are not introducing any new elements into the TSS, since by definition, a TSS is a pair (Σ, Ψ) where Σ is a set of function symbols and Ψ is a set of transition rules assigned to the function symbols.

The introduction of ϵ transitions and thus form (4) will make Proposition 5.5.1 and its proof easier to be comprehended. In Proposition 5.5.1, we will show that, in τDes-languages, when process $C(p_1, \ldots, p_n)$ evolves into

$C'(p'_1, \ldots, p'_n)$, by applying a ruloid, and produces a transition (observable action or τ transition), each of its subprocesses p_i also evolves into p'_i and produces a transition (observable action, τ transition or ϵ transition).

Based on ruloids and ruloid theorem, we may restate patience rules with patience ruloids.

Definition 5.2.2 *A ruloid of the form* $\dfrac{x_i \xrightarrow{\tau} x'_i}{C(x_1, \ldots, x_i, \ldots, x_n) \xrightarrow{\tau} C(x_1, \ldots, x'_i, \ldots, x_n)}$ *with* $1 \leq i \leq n$ *is called a patience ruloid of the ith argument of the context* C.

A ruloid is called a plain ruloid if it is not a patience ruloid. Similar with the division in the patience rules, we can also divide the patience ruloids into three classes, i.e., patience ruloids for active arguments, patience ruloids for receiving arguments and patience ruloids for other arguments.

In fact, Strategy 5.2.1 has already provided a canonical way to retrieve this division. Let $\mathcal{L}$ be a de Simone language and C be any context of n holes in it.

1. If adding patience rules for active arguments into the language, then after applying Strategy 5.2.1, patience ruloids for active arguments are those patience ruloids in $\mathscr{R}(C, \tau)$.

2. If further adding patience rules for receiving arguments into the language, then, after applying Strategy 5.2.1, $\mathscr{R}(C, \tau)$ contains both patience ruloids for active arguments and patience ruloids for receiving arguments. By getting rid of those patience ruloids for active arguments obtained in the first point, we can obtain the patience ruloids for receiving arguments.

Because this division is obtained indirectly from Strategy 5.2.1 and patience rules, it is hard to be used in the following. Here, we propose another equivalent division which is directly based on the arguments of a context.

Definition 5.2.3 *Let* $\mathcal{L} = (\Sigma, \Psi)$ *be a* τ*Des-language, and* C *be any context of* n *holes. The* i*th argument of the context* C *is active if there exists a plain ruloid* r *of form (3) in* $\mathscr{R}(C, \tau)$ *such that* x_i *appears as left-hand side of a premise. The* i*th argument of the context* C *is receiving if it is not active and there exists another context* D *and a plain ruloid* r *of form (3) in* $\mathscr{R}(D)$ *such that* $C(x'_1, \ldots, x'_n)$ *appears as the target of* r *and* x'_i *appears as right-hand side of a premise.*

Proposition 5.2.1 *The division defined by Definition 5.2.3 is equivalent to the division obtained from Strategy 5.2.1 and patience rules.*

Proof. ($\Longleftarrow$) Let $\mathcal{L} = (\Sigma, \Psi)$ be a de Simone language, and C be any context of n holes. If only adding patience rules for active arguments into the language, we need to show that each active argument of the context C defined by Definition 5.2.3 has a patience ruloid. We will prove by making an induction on the context C and Strategy 5.2.1.

1. If $C \in V$ or $C \in \Sigma$, then it can be easily obtained from Strategy 5.2.1 and Definition 2.5.1.

2. If C is any context, then it can be rewritten as $f(C_1(X_1), \ldots, C_m(X_m))$. Assume that contexts $C_1, \ldots, C_m$ satisfy that each active argument has a patience ruloid.

3. We need to prove that each active argument of C defined by Definition 5.2.3 has a patience ruloid. Suppose that the ith argument of C is an active argument. Then, by Definition 5.2.3, there exists a plain ruloid r of form (3) in $\mathscr{R}(C,\tau)$ such that x_i appears as left-hand side of a premise. By Strategy 5.2.1, x_i must appear as left-hand side of a premise of some context. Without loss of generality, assume that x_i is the kth argument of the C_j. By the induction hypothesis, the kth argument of C_j is active and thus has a patience ruloid. Also by Strategy 5.2.1, the jth argument of functor f is active and thus has a patience ruloid. Therefore, we have that the ith argument of C has a patience ruloid by Strategy 5.2.1 and the above two patience ruloids for C_j and f, respectively.

If further adding the patience rules for receiving arguments into the language, we need to prove that each receiving argument of the context C defined by Definition 5.2.3 has a patience ruloid. Assume that the ith argument of context C is receiving. Then, by Definition 5.2.3, there exist another context D and a plain ruloid r of form (3) in $\mathscr{R}(D)$ such that $C(x'_1,\ldots,x'_n)$ appears as the target of r and x'_i appears as right-hand side of a premise. We will prove by making an induction on context C and Strategy 5.2.1.

1. If $C \in V$ or $C \in \Sigma$, then, by Definition 2.5.1, the ith argument of C is receiving. Therefore, it should have a patience rule by the hypothesis. By Strategy 5.2.1, each patience rule is also a patience ruloid.

2. If C is any context, then it can be rewritten as $f(C_1(X_1),\ldots,C_m(X_m))$. Assume that contexts $C_1,\ldots,C_m$ satisfy that each receiving argument has a patience ruloid.

3. We need to prove that the ith argument of C defined by Definition 5.2.3 has a patience ruloid. By Strategy 5.2.1, x_i must appear as right-hand side of a premise of some context. Without loss of generality, assume that x_i is the kth argument of the C_j. By the induction hypothesis, the kth argument of C_j is receiving or active and thus has a patience ruloid. Also by Strategy 5.2.1, the jth argument of functor f is receiving or active and thus has a patience ruloid. Therefore, we have that the ith argument of C has a patience ruloid by Strategy 5.2.1 and the above two patience ruloids for C_j and f, respectively.

($\Longrightarrow$) It is trivially true since, according to Strategy 5.2.1, each rule is also a ruloid. That is to say, we may first obtain the division on patience rules from the division on patience ruloids in Definition 5.2.3, and then using Strategy 5.2.1 to obtain the division from Strategy 5.2.1 and patience rules. □

5.3 Statement of Meta Theorem

First of all, we present the statement of meta theorem.

Theorem 5.3.1 *Let $\mathcal{L}$ be a τDes-language and C be any context of n holes. ζ and ξ are two closed Σ-substitutions mapping x_i into p_i and q_i, respectively. If $\forall 1 \leq i \leq n : p_i \sqsubseteq_{st} q_i$, then $C(p_1,\ldots,p_n) \sqsubseteq_{st} C(q_1,\ldots,q_n)$.*

In order to prove $C(p_1,\ldots,p_n) \sqsubseteq_{st} C(q_1,\ldots,q_n)$ for some context C of n hole for the case of $\forall 1 \leq i \leq n : p_i \sqsubseteq_{st} q_i$, it is enough to prove it for the case of $p_1 \sqsubseteq_{st} q_1$ and $\forall 2 \leq i \leq n : p_i \equiv q_i$. In fact, if the later is true, then we can

deduce $C(p_1,\ldots,p_n) \sqsubseteq_{st} C(q_1,\ldots,q_n)$ from $C(p_1,\ldots,p_n) \sqsubseteq_{st} C(q_1,p_2,\ldots,p_n) \sqsubseteq_{st} C(q_1,q_2,p_3,\ldots,p_n) \sqsubseteq_{st} \ldots \sqsubseteq_{st} C(q_1,q_2,\ldots,q_n)$ and the transitivity of $\sqsubseteq_{st}$, which is a preorder relation. Therefore, the meta theorem is equivalent to the following theorem.

Theorem 5.3.2 *Let $\mathcal{L}$ be a τDes-language and C be any context of n holes. ζ and ξ are two closed Σ-substitutions mapping x_i into p_i and q_i, respectively. If $p_1 \sqsubseteq_{st} q_1$ and $\forall 2 \leq i \leq n : p_i \equiv q_i$, then $C(p_1,\ldots,p_n) \sqsubseteq_{st} C(q_1,\ldots,q_n)$.*

Furthermore, by the definition of should testing preorder in Definition 3.0.4, to prove the above theorem, it is equivalent to prove that, if $p_1 \sqsubseteq_{st} q_1$ and $\forall 2 \leq i \leq n : p_i \equiv q_i$, then, for any $(\sigma,\Phi) \in Act^* \times \mathcal{P}(Act^+)$, if $(\sigma,\Phi) \in \mathcal{I}(C(p_1,\ldots,p_n))$ then, there exists some $\mu \in Act^*$ such that $(\sigma\mu,\mu^{-1}\Phi) \in \mathcal{I}(C(q_1,\ldots,q_n))$ and $\varepsilon \notin \mu^{-1}\Phi$.

We will prove it by the steps showing in the following several subsections from Section 5.4 to Section 5.9. Finally, in Section 5.10, a review on these steps will be made via a proof sketch.

5.4 A Stronger Theorem

In this subsection, we will show that, for any $(\sigma,\Phi) \in \mathcal{I}(C(p_1,\ldots,p_n))$, there exists some Φ' such that

1. $(\sigma,\Phi') \in \mathcal{I}(C(p_1,\ldots,p_n))$,
2. if there exists some $\mu \in Act^*$ such that $(\sigma\mu,\mu^{-1}\Phi') \in \mathcal{I}(C(q_1,\ldots,q_n))$, then $(\sigma\mu,\mu^{-1}\Phi) \in \mathcal{I}(C(q_1,\ldots,q_n))$, and
3. if $\varepsilon \notin \mu^{-1}\Phi'$ then $\varepsilon \notin \mu^{-1}\Phi$.

With this Φ', to prove Theorem 5.3.2, it is enough to prove the following theorem:

Theorem 5.4.1 *Let $\mathcal{L}$ be a τDes-language and C be any context of n holes. ζ and ξ are two closed Σ-substitutions mapping x_i into p_i and q_i, respectively. If $p_1 \sqsubseteq_{st} q_1$ and $\forall 2 \leq i \leq n : p_i \equiv q_i$, then, for any $(\sigma,\Phi) \in Act^* \times \mathcal{P}(Act^+)$, if $(\sigma,\Phi) \in \mathcal{I}(C(p_1,\ldots,p_n))$ then, there exists some $\mu \in Act^*$ such that $(\sigma\mu,\mu^{-1}\Phi') \in \mathcal{I}(C(q_1,\ldots,q_n))$ and $\varepsilon \notin \mu^{-1}\Phi'$, where Φ' is the one introduced above.*

Proposition 5.4.1 *Theorem 5.4.1 implies Theorem 5.3.2.*

Proof. Suppose that $(\sigma,\Phi) \in Act^* \times \mathcal{P}(Act^+)$ is any impossible future pair such that $(\sigma,\Phi) \in \mathcal{I}(C(p_1,\ldots,p_n))$. By the requirements of Φ' in the above, we have

1. $(\sigma,\Phi') \in \mathcal{I}(C(p_1,\ldots,p_n))$,
2. if there exists some $\mu \in Act^*$ such that $(\sigma\mu,\mu^{-1}\Phi') \in \mathcal{I}(C(q_1,\ldots,q_n))$, then $(\sigma\mu,\mu^{-1}\Phi) \in \mathcal{I}(C(q_1,\ldots,q_n))$, and
3. if $\varepsilon \notin \mu^{-1}\Phi'$ then $\varepsilon \notin \mu^{-1}\Phi$.

By Theorem 5.4.1 and $(\sigma, \Phi) \in \mathcal{I}(C(p_1, \ldots, p_n))$, there exists some $\mu \in Act^*$ such that $(\sigma\mu, \mu^{-1}\Phi') \in \mathcal{I}(C(q_1, \ldots, q_n))$ and $\varepsilon \notin \mu^{-1}\Phi'$. Therefore, we have, by the second point above, $(\sigma\mu, \mu^{-1}\Phi) \in \mathcal{I}(C(q_1, \ldots, q_n))$, and, by the third point above, $\varepsilon \notin \mu^{-1}\Phi$. □

The remaining part of this subsection is devoted to show a strategy to construct Φ' from Φ and then prove that the obtained Φ' satisfies the above-mentioned three requirements.

Strategy 5.4.1 *By the definition of impossible future pair and $(\sigma, \Phi) \in \mathcal{I}(C(p_1, \ldots, p_n))$, there exists some $C'(p'_1, \ldots, p'_n)$ such that $C(p_1, \ldots, p_n) \stackrel{\sigma}{\Longrightarrow} C'(p'_1, \ldots, p'_n)$ and $\mathcal{T}(C'(p'_1, \ldots, p'_n)) \cap \Phi = \emptyset$. Therefore, for all $\delta \in \Phi$, we have $\delta \notin \mathcal{T}(C'(p'_1, \ldots, p'_n))$. Then, Φ' is constructed by the following steps:*

1. *let $\Phi' = \emptyset$.*
2. *for all $\delta \in \Phi$,*
   (a) *there must exist its maximal prefix $\delta' \in Act^*$ such that $C'(p'_1, \ldots, p'_n) \stackrel{\delta'}{\Longrightarrow}$ and $\delta = \delta' a \delta''$, where $a \in Act$ denotes the observable action after δ', and then,*
   (b) *add $\delta' a$ into set Φ'.*

Note that, the maximal prefix δ' of some action sequence δ satisfying some given conditions, implies that no other prefix δ''' of δ satisfies both $|\delta'''| > |\delta'|$ and the given conditions. Besides, $\Phi \in Act^+$ implies that each $\delta \in \Phi$ has a corresponding $\delta' a \in \Phi'$.

Below, we will show, by the following three propositions, that the obtained Φ' satisfies the above-mentioned three conditions.

Proposition 5.4.2 *If $(\sigma, \Phi) \in \mathcal{I}(C(p_1, \ldots, p_n))$, then $(\sigma, \Phi') \in \mathcal{I}(C(p_1, \ldots, p_n))$, where Φ' is the one obtained by Strategy 5.4.1.*

Proof. By Strategy 5.4.1, each element in Φ' is in the form $\delta' a$, where δ' is a maximal prefix of some $\delta \in \Phi$ satisfying that $C'(p'_1, \ldots, p'_n) \stackrel{\delta'}{\Longrightarrow}$. Therefore, by the maximality of δ', we have that $C'(p'_1, \ldots, p'_n) \stackrel{\delta' a}{\Rightarrow}$, i.e., $\delta' a \notin \mathcal{T}(C'(p'_1, \ldots, p'_n))$. Then, by the randomicity of $\delta' a$ in Φ', we have $\Phi' \cap \mathcal{T}(C'(p'_1, \ldots, p'_n)) = \emptyset$. Therefore, $(\sigma, \Phi') \in \mathcal{I}(C(p_1, \ldots, p_n))$. □

Proposition 5.4.3 *Let $(\sigma, \Phi) \in \mathcal{I}(C(p_1, \ldots, p_n))$ and Φ' be the one constructed by Strategy 5.4.1. If there exists some $\mu \in Act^*$ such that $(\sigma\mu, \mu^{-1}\Phi') \in \mathcal{I}(C(q_1, \ldots, q_n))$, then $(\sigma\mu, \mu^{-1}\Phi) \in \mathcal{I}(C(q_1, \ldots, q_n))$.*

Proof. Suppose that there exists some $\mu \in Act^*$ such that $(\sigma\mu, \mu^{-1}\Phi') \in \mathcal{I}(C(q_1, \ldots, q_n))$. By the definition of impossible future pair, there exists some $C''(q''_1, \ldots, q''_n)$ such that $C(q_1, \ldots, q_n) \stackrel{\sigma\mu}{\Longrightarrow} C''(q''_1, \ldots, q''_n)$ and $\mathcal{T}(C''(q''_1, \ldots, q''_n)) \cap \mu^{-1}\Phi' = \emptyset$. Therefore, for all $\delta''' \in \mu^{-1}\Phi'$, we have $\delta''' \notin \mathcal{T}(C''(q''_1, \ldots, q''_n))$.

Now, by Strategy 5.4.1, each element in Φ' is in form $\delta' a$, where $\delta' a \delta'' = \delta$ and $\delta \in \Phi$. Without loss of generality, we suppose that $\delta''' = \mu^{-1}\delta' a$. By $\delta''' \notin \mathcal{T}(C''(q_1'', \ldots, q_n''))$, $\mu^{-1}\delta' a \notin \mathcal{T}(C''(q_1'', \ldots, q_n''))$ and thus $\mu^{-1}\delta \notin \mathcal{T}(C''(q_1'', \ldots, q_n''))$.

Finally, by the 1-1 correspondence between δ and $\delta' a$, we have that $\mu^{-1}\delta \notin \mathcal{T}(C''(q_1'', \ldots, q_n''))$ for all $\delta \in \Phi$, i.e., $\mu^{-1}\Phi \cap \mathcal{T}(C''(q_1'', \ldots, q_n'')) = \emptyset$.

Then, we can obtain $(\sigma\mu, \mu^{-1}\Phi) \in \mathcal{I}(C(q_1, \ldots, q_n))$ by the definition of impossible future pair. □

Proposition 5.4.4 *If $\varepsilon \notin \mu^{-1}\Phi'$ then $\varepsilon \notin \mu^{-1}\Phi$.*

Proof. By Proposition 3.0.1, $\mu \in \{\varepsilon\} \cup \downarrow \Phi - \Phi$ is equivalent to $\varepsilon \notin \mu^{-1}\Phi$. Therefore, if $\varepsilon \notin \mu^{-1}\Phi'$, then $\mu \in \{\varepsilon\} \cup \downarrow \Phi' - \Phi'$.

By the Strategy 5.4.1, each element in Φ' is a prefix of some element in Φ. Therefore, if $\mu \in \{\varepsilon\} \cup \downarrow \Phi' - \Phi'$ then there should be $\mu \in \{\varepsilon\} \cup \downarrow \Phi - \Phi$. Finally, by $\mu \in \{\varepsilon\} \cup \downarrow \Phi - \Phi$ is equivalent to $\varepsilon \notin \mu^{-1}\Phi$, we have $\varepsilon \notin \mu^{-1}\Phi$. □

5.5 Decomposing (σ, Φ')

The objective of this subsection is to decompose $(\sigma, \Phi') \in \mathcal{I}(C(p_1, \ldots, p_n))$, which is obtained in the previous subsection, into its counterparts for subprocesses p_i, i.e., to obtain $(\sigma_i, \Phi_i') \in \mathcal{I}(p_i)$.

First of all, we need a proposition to show the decomposability of weak traces in τDes-languages.

Proposition 5.5.1 *Let $\mathcal{L} = (\Sigma, \Psi)$ be an τDes-language, and $C(x_1, \ldots, x_n)$ be any context of n holes. Suppose that ζ is any closed Σ-substitution mapping x_i into p_i. If σ is a trace in $\mathcal{T}(C(p_1, \ldots, p_n))$, then, for all $1 \le i \le n$, there is a trace σ_i in $\mathcal{T}(p_i)$ such that, when $C(p_1, \ldots, p_n) \stackrel{\sigma}{\Longrightarrow} C'(p_1', \ldots, p_n')$, we have $p_i \stackrel{\sigma_i}{\Longrightarrow} p_i'$.*

Proof. Since $C(p_1, \ldots, p_n) \stackrel{\sigma}{\Longrightarrow} C'(p_1', \ldots, p_n')$, we have $C(p_1, \ldots, p_n) = C_0(p_{10}, \ldots, p_{n0}) \stackrel{\alpha_1}{\longrightarrow} C_1(p_{11}, \ldots, p_{n1}) \stackrel{\alpha_2}{\longrightarrow} \ldots \stackrel{\alpha_m}{\longrightarrow} C_m(p_{1m}, \ldots, p_{nm}) = C'(p_1', \ldots, p_n')$, where $\forall 1 \le j \le m : \alpha_j \in Act \cup \{\tau\}$ and $\sigma' = \alpha_1 \ldots \alpha_m$ is equivalent to σ if all its τ transitions are omitted.

We will prove this proposition by making an induction on the length of σ'.

1) $|\sigma'| = 1$. Let $\sigma' = \alpha$. By the completeness property of the ruloids, there should be a ruloid of form (3) in $\mathscr{R}(C, \alpha)$, and $p_i \stackrel{\alpha_i}{\longrightarrow}$ for all $i \in I$. As shown before that, we have a ruloid of form (4) corresponding with form (3). Therefore, there exist $p_i \stackrel{\alpha_i}{\longrightarrow} p_i'$ for all $i \in I$ and $p_i \stackrel{\epsilon}{\longrightarrow} p_i'$ for all $i \in \{1, \ldots, n\} - I$, i.e., when $C(p_1, \ldots, p_n) \stackrel{\sigma}{\Longrightarrow} C'(p_1', \ldots, p_n')$, we have $p_i \stackrel{\alpha_i}{\Longrightarrow} p_i'$ for all $i \in I$ and $p_i \stackrel{\tau^*}{\longrightarrow} p_i'$ for all $i \in \{1, \ldots, n\} - I$.

2) Assume that, when $|\sigma'| = m - 1$ with $m \ge 1$, if σ is a trace in $\mathcal{T}(C(p_1, \ldots, p_n))$ then, for all $1 \le i \le n$, there should be a trace σ_i in $\mathcal{T}(p_i)$ such that, when $C(p_1, \ldots, p_n) \stackrel{\sigma}{\Longrightarrow} C'(p_1', \ldots, p_n')$, we have $p_i \stackrel{\sigma_i}{\Longrightarrow} p_i'$.

3) For $|\sigma'| = m$, suppose that $C(p_1, \ldots, p_n) = C_0(p_{10}, \ldots, p_{n0}) \stackrel{\alpha_1}{\longrightarrow} C_1(p_{11}, \ldots, p_{n1}) \stackrel{\alpha_2}{\longrightarrow} \ldots \stackrel{\alpha_m}{\longrightarrow} C_m(p_{1m}, \ldots, p_{nm}) = C'(p_1', \ldots, p_n')$. By the induction hypothesis, when $C(p_1, \ldots, p_n) = C_0(p_{10}, \ldots, p_{n0}) \stackrel{\alpha_1}{\longrightarrow} C_1(p_{11}, \ldots, p_{n1}) \stackrel{\alpha_2}{\longrightarrow} \ldots \stackrel{\alpha_{m-1}}{\longrightarrow} C_{m-1}(p_{1(m-1)}, \ldots, p_{n(m-1)}) = C''(p_1'', \ldots, p_n'')$, we have $p_i \stackrel{\sigma_i''}{\Longrightarrow} p_i''$. Now, when $C_{m-1}(p_{1(m-1)}, \ldots, p_{n(m-1)}) = C''(p_1'', \ldots, p_n'') \stackrel{\alpha_m}{\longrightarrow}$

$C_m(p_{1m}, \ldots, p_{nm}) = C'(p'_1, \ldots, p'_n)$, we have $p''_i \stackrel{\alpha'_m}{\Longrightarrow} p'_i$. Therefore, when $C(p_1, \ldots, p_n) \stackrel{\sigma}{\Longrightarrow} C'(p'_1, \ldots, p'_n)$, we have $p_i \stackrel{\sigma''_i \alpha'_m}{\Longrightarrow} p'_i$. □

By $(\sigma, \Phi') \in \mathcal{I}(C(p_1, \ldots, p_n))$ and the definition of impossible future pair, there exists some $C'(p'_1, \ldots, p'_n)$ such that $C(p_1, \ldots, p_n) \stackrel{\sigma}{\Longrightarrow} C'(p'_1, \ldots, p'_n)$ and $\mathcal{T}(C'(p'_1, \ldots, p'_n)) \cap \Phi' = \emptyset$.

Now, by Proposition 5.5.1, the procedure $C(p_1, \ldots, p_n) \stackrel{\sigma}{\Longrightarrow} C'(p'_1, \ldots, p'_n)$ can be decomposed into $p_i \stackrel{\sigma_i}{\Longrightarrow} p'_i$ for all $1 \leq i \leq n$.

For the case of Φ', we will propose a strategy to construct Φ'_i as follows.

Strategy 5.5.1 *By Strategy 5.4.1, all the elements of Φ' are in form $\delta' a$ such that $C'(p'_1, \ldots, p'_n) \stackrel{\delta'}{\Longrightarrow}$. Now, for some subprocess p_i,*

1. *for all action sequences $\nu \in Act^*$, let set $A^i_\nu = \emptyset$.*
2. *for all $\delta' a \in \Phi'$ and for all $C''(p''_1, \ldots, p''_n)$ with $C'(p'_1, \ldots, p'_n) \stackrel{\delta'}{\Longrightarrow} C''(p''_1, \ldots, p''_n)$,*
   (a) *by Proposition 5.5.1, decompose δ' into δ'_i, and*
   (b) *add all elements of set $\{a \in Act | p''_i \stackrel{a}{\Longrightarrow}\}$ into set $A^i_{\delta'_i}$.*
3. *let Φ'_i be the set $\bigcup\{\delta'_i b | A^i_{\delta'_i} \neq \emptyset \wedge b \in Act \wedge b \notin A^i_{\delta'_i}\}$. Note: if some set $A^i_{\delta'_i} = Act$ then let $b = \beta$ for some $\beta \notin Act$.*

In fact, set $A^i_{\delta'_i}$ is the set of all possible next actions of p'_i after it executes a weak trace δ'_i. And thus, b in the third point of the above strategy is used to denote that, for any p''_i with $p'_i \stackrel{\delta'_i}{\Longrightarrow} p''_i$, $p''_i \stackrel{b}{\nRightarrow}$, though there exists some p''_i such that $p'_i \stackrel{\delta'_i}{\Longrightarrow} p''_i$. Furthermore, Φ'_i contains all $\delta'_i b$ like that. Therefore, for some δ'_i with $A^i_{\delta'_i} \neq \emptyset$, if $\delta'_i a \notin \Phi'_i$ then there must exist some p''_i such that $p'_i \stackrel{\delta'_i}{\Longrightarrow} p''_i$ and $p''_i \stackrel{a}{\Longrightarrow}$.

Below, we will prove that $(\sigma_i, \Phi'_i) \in \mathcal{I}(p_i)$.

Proposition 5.5.2 $(\sigma_i, \Phi'_i) \in \mathcal{I}(p_i)$.

Proof. First, by $(\sigma, \Phi') \in \mathcal{I}(C(p_1, \ldots, p_n))$, we have $C(p_1, \ldots, p_n) \stackrel{\sigma}{\Longrightarrow} C'(p'_1, \ldots, p'_n)$, Then by Proposition 5.5.1, $p_i \stackrel{\sigma_i}{\Longrightarrow} p'_i$ when $C(p_1, \ldots, p_n) \stackrel{\sigma}{\Longrightarrow} C'(p'_1, \ldots, p'_n)$.

Now, it is enough to prove that, for all elements δ''_i in Φ'_i, there is $p'_i \stackrel{\delta''_i}{\nRightarrow}$.

Observe that, by Strategy 5.5.1, δ''_i must be in the form $\delta'_i b$ such that $p'_i \stackrel{\delta'_i}{\Longrightarrow}$. Moreover, though $p'_i \stackrel{\delta'_i}{\Longrightarrow}$, b should satisfy that, for all p''_i with $p'_i \stackrel{\delta'_i}{\Longrightarrow} p''_i$, $p''_i \stackrel{b}{\nRightarrow}$. Therefore, there should be $p'_i \stackrel{\delta'_i b}{\nRightarrow}$.

By the randomicity of δ''_i, and thus $\delta'_i b$, we have that, for all elements δ''_i in Φ'_i, there is $p'_i \stackrel{\delta''_i}{\nRightarrow}$. □

5.6 Application of Should Testing Preorder in Subprocesses

By now, we have an impossible future pair for each subprocess p_i, i.e., $(\sigma_i, \Phi'_i) \in \mathcal{I}(p_i)$. Therefore, by the hypothesis that $p_i \sqsubseteq_{st} q_i$ and the definition of should testing preorder, there exist some μ_i and q'_i such that $(\sigma_i \mu_i, \mu_i^{-1} \Phi'_i) \in \mathcal{I}(q_i)$, i.e., $q_i \stackrel{\sigma_i \mu_i}{\Longrightarrow} q'_i$ and $\mu_i^{-1} \Phi'_i \cap \mathcal{T}(q'_i) = \emptyset$.

Furthermore, as stated in Section 5.3, we can safely suppose that $p_1 \sqsubseteq_{st} q_1$ and, for all $2 \leq i \leq n$, $p_i \equiv q_i$. Therefore, we will have

1. there exists some $\mu_1 \in Act^*$ such that $(\sigma_1\mu_1, \mu_1^{-1}\Phi_1') \in \mathcal{I}(q_1)$ and $\varepsilon \notin \mu_1^{-1}\Phi_1'$, and
2. when $2 \leq i \leq n$, for all μ_i such that $\varepsilon \notin \mu_i^{-1}\Phi_i'$ and $\mu_i \in \mathcal{T}(p_i')$, there is $(\sigma_i\mu_i, \mu_i^{-1}\Phi_i') \in \mathcal{I}(q_i)$, where, by definition of impossible future pair and $(\sigma_i, \Phi_i') \in \mathcal{I}(p_i)$, $p_i \overset{\sigma_i}{\Longrightarrow} p_i'$ and $\mathcal{T}(p_i') \cap \Phi_i' = \emptyset$. Note that, there always exists a such μ_i since, by $p_i \equiv q_i$ and $(\sigma_i, \Phi_i') \in \mathcal{I}(p_i)$, we can simply let $\mu_i = \tau^*$ and thus $(\sigma_i\mu_i, \mu_i^{-1}\Phi_i') = (\sigma_i, \Phi_i') \in \mathcal{I}(p_i) = \mathcal{I}(q_i)$.

The first clause is trivial from the definition of should testing preorder, we will prove the second clause as follows.

Proposition 5.6.1 *Let $(\sigma_i, \Phi_i') \in \mathcal{I}(p_i)$. Then, there exists some p_i' such that $p_i \overset{\sigma_i}{\Longrightarrow} p_i'$ and $\mathcal{T}(p_i') \cap \Phi_i' = \emptyset$. For any $\nu_i \in \mathcal{T}(p_i')$, if $\varepsilon \notin \nu_i^{-1}\Phi_i'$ then $(\sigma_i\nu_i, \nu_i^{-1}\Phi_i') \in \mathcal{I}(p_i)$.*

Proof. By the fact that $\nu_i \in \mathcal{T}(p_i')$, to prove that $(\sigma_i\nu_i, \nu_i^{-1}\Phi_i') \in \mathcal{I}(p_i)$, it is enough to show that there exists some p_i'' such that $p_i \overset{\sigma_i}{\Longrightarrow} p_i' \overset{\nu_i}{\Longrightarrow} p_i''$ and $\mathcal{T}(p_i'') \cap \nu_i^{-1}\Phi_i' = \emptyset$.

By $\nu_i \in \mathcal{T}(p_i')$, there must exist some p_i'' such that $p_i' \overset{\nu_i}{\Longrightarrow} p_i''$. We need to show that $\mathcal{T}(p_i'') \cap \nu_i^{-1}\Phi_i' = \emptyset$, i.e., $\forall \mu_i \in \nu_i^{-1}\Phi_i' : \mu_i \notin \mathcal{T}(p_i'')$.

To show $\forall \mu_i \in \nu_i^{-1}\Phi_i' : \mu_i \notin \mathcal{T}(p_i'')$, it is enough to show that $\forall \mu_i\nu_i \in \Phi_i' : \mu_i\nu_i \notin \mathcal{T}(p_i')$ since $p_i' \overset{\nu_i}{\Longrightarrow} p_i''$.

Finally, it is guaranteed by $\mathcal{T}(p_i') \cap \Phi_i' = \emptyset$. □

In the remaining of this subsection, we will prove several propositions which will be used in the following subsections. All these propositions are based on the assumptions that $(\sigma_i, \Phi_i') \in \mathcal{I}(p_i)$, $p_i \sqsubseteq_{st} q_i$ and thus there exists some $\mu_i \in Act^*$ such that $(\sigma_i\mu_i, \mu_i^{-1}\Phi_i') \in \mathcal{I}(q_i)$ and $\varepsilon \notin \mu_i^{-1}\Phi_i'$.

The first proposition states that μ_i should be in $\mathcal{T}(p_i')$, where, by definition of impossible future pair and $(\sigma_i, \Phi_i') \in \mathcal{I}(p_i)$, $p_i \overset{\sigma_i}{\Longrightarrow} p_i'$ and $\mathcal{T}(p_i') \cap \Phi_i' = \emptyset$.

Proposition 5.6.2 $\mu_i \in \mathcal{T}(p_i')$.

Proof. It can be easily obtained by Strategy 5.5.1 and the requirement that $\varepsilon \notin \mu_i^{-1}\Phi_i'$.

By Strategy 5.5.1, each elements in Φ_i' are in the form $\delta_i'b$ such that $p_i' \overset{\delta_i'}{\Longrightarrow}$ but $p_i' \overset{\delta_i'b}{\nRightarrow}$.

Therefore, μ_i can only be a prefix of some $\delta_i'b$ in Φ_i'. Moreover, μ_i cannot be $\delta_i'b$ since $\varepsilon \notin \mu_i^{-1}\Phi_i'$.

Finally, μ_i must be a proper prefix of $\delta_i'b$. Then, by $p_i' \overset{\delta_i'b}{\nRightarrow}$, we have that $\mu_i \in \mathcal{T}(p_i')$. □

The second proposition states that we can extend $\sigma_i\mu_i$ randomly along the weak traces of p_i', and obtain another pair $(\sigma_i\mu_i\nu_i, (\mu_i\nu_i)^{-1}\Phi_i')$, which is also an impossible future pair of p_i. It is much similar with Proposition 5.6.1, we treat it as an individual proposition just for the convenience of using it.

Proposition 5.6.3 *Let $(\sigma_i\mu_i, \mu_i^{-1}\Phi_i') \in \mathcal{I}(p_i)$. Then, there exists some p_i' such that $p_i \overset{\sigma_i\mu_i}{\Longrightarrow} p_i'$ and $\mathcal{T}(p_i') \cap \mu_i^{-1}\Phi_i' = \emptyset$. For any $\nu_i \in \mathcal{T}(p_i')$, if $\varepsilon \notin (\mu_i\nu_i)^{-1}\Phi_i'$ then $(\sigma_i\mu_i\nu_i, (\mu_i\nu_i)^{-1}\Phi_i') \in \mathcal{I}(p_i)$.*

Proof. It is trivially true from Proposition 5.6.1 by substituting σ_i with $\sigma_i\mu_i$ and Φ'_i with $\mu_i^{-1}\Phi'_i$. □

Moreover, as a corollary of the above two propositions, $\mu_i\nu_i$, which is stated in the above proposition for the case of $(\sigma_i\mu_i, \mu_i^{-1}\Phi'_i) \in \mathcal{I}(p_i)$, should be also in $\mathcal{T}(p'_i)$.

Proposition 5.6.4 *$\mu_i\nu_i \in \mathcal{T}(p'_i)$.*

Proof. It is trivially true from Proposition 5.6.2 and Proposition 5.6.3. □

5.7 Seeking for $\sigma_i\mu_i$ in $C(p_1, \ldots, p_n)$

Though, in the preceding subsection, we have shown that there exist some impossible future pairs $(\sigma_i\mu_i, \mu_i^{-1}\Phi'_i)$ for all subprocesses q_i, we cannot expect that any $(\sigma_i\mu_i, \mu_i^{-1}\Phi'_i) \in \mathcal{I}(q_i)$ for all $1 \leq i \leq n$ can directly compose $(\sigma\mu, \mu^{-1}\Phi') \in \mathcal{I}(C(q_1, \ldots, q_n))$. In fact, to show that there exists some μ such that $(\sigma\mu, \mu^{-1}\Phi') \in \mathcal{I}(C(q_1, \ldots, q_n))$ and $\varepsilon \notin \mu^{-1}\Phi'$, we need to make more restricts on μ_i.

To this end, we are aiming at seeking some specific $\sigma_i\mu_i$ for all $1 \leq i \leq n$ in this subsection. After retrieving these $\sigma_i\mu_i$, we will, in Section 5.8, show that they can compose a weak trace $\sigma\mu$ of $C(q_1, \ldots, q_n)$, and then, in Section 5.9, show that a variant of μ, called μ', will make $(\sigma\mu', \mu'^{-1}\Phi') \in \mathcal{I}(C(q_1, \ldots, q_n))$ hold, which will finish our proof.

Before that, we conclude the results in Section 5.6 as follows.

1. For the case of p_i with $2 \leq i \leq n$, by the hypothesis that $\forall 2 \leq i \leq n : p_i \equiv q_i$ and the fact that $(\sigma_i, \Phi'_i) \in \mathcal{I}(p_i)$, we have $(\sigma_i, \Phi'_i) \in \mathcal{I}(q_i)$. Then, by Proposition 5.6.1, to keep $(\sigma_i\mu_i, \mu_i^{-1}\Phi'_i) \in \mathcal{I}(q_i)$, μ_i can be any action sequence such that $\varepsilon \notin \mu_i^{-1}\Phi'_i$. Observe that, by Proposition 5.6.2, the requirement $\mu_i \in \mathcal{T}(p'_i)$, in the beginning of Section 5.6 for the case of p_i with $2 \leq i \leq n$, has been implied in $(\sigma_i\mu_i, \mu_i^{-1}\Phi'_i) \in \mathcal{I}(q_i)$.

2. For the case of p_1, since $p_1 \sqsubseteq_{st} q_1$ and $(\sigma_1, \Phi'_1) \in \mathcal{I}(p_1)$, we have that, there exists some μ_1 such that $(\sigma_1\mu_1, \mu_1^{-1}\Phi'_1) \in \mathcal{I}(q_1)$ and $\varepsilon \notin \mu_1^{-1}\Phi'_1$. Furthermore, by Proposition 5.6.2, $\mu_1 \in \mathcal{T}(p'_1)$, where p'_1 satisfies that $p_1 \stackrel{\sigma_1}{\Longrightarrow} p'_1$ and $\mathcal{T}(p'_1) \cap \Phi'_1 = \emptyset$.

The following proposition provides a way to obtain those specific $\sigma_i\mu_i$ for all $1 \leq i \leq n$. Suppose that $(\sigma, \Phi') \in \mathcal{I}(C(p_1, \ldots, p_n))$, and thus, by the definition of impossible future pair, there exists $C'(p'_1, \ldots, p'_n)$ such that $C(p_1, \ldots, p_n) \stackrel{\sigma}{\Longrightarrow} C'(p'_1, \ldots, p'_n)$ and $\mathcal{T}(C'(p'_1, \ldots, p'_n)) \cap \Phi' = \emptyset$.

Proposition 5.7.1 *If some $\mu \in Act^*$ and process $C''(p''_1, \ldots, p''_n)$ satisfy that $C'(p'_1, \ldots, p'_n) \stackrel{\mu}{\Longrightarrow} C''(p''_1, \ldots, p''_n)$ and, after decomposing μ using Proposition 5.5.1, μ_1 is equivalent to the μ_1 in $(\sigma_1\mu_1, \mu_1^{-1}\Phi'_1) \in \mathcal{I}(q_1)$, then the obtained μ_i will satisfy that $(\sigma_i\mu_i, \mu_i^{-1}\Phi'_i) \in \mathcal{I}(q_i)$ for all $1 \leq i \leq n$.*

Proof. We divide it into two cases:

If $i = 1$, then $p_1 \sqsubseteq q_1$. $(\sigma_1\mu_1, \mu_1^{-1}\Phi'_i) \in \mathcal{I}(q_1)$ comes directly from the hypothesis.

If $2 \leq i \leq n$, then $p_i \equiv q_i$. By the fact that $(\sigma_i, \Phi_i') \in \mathcal{I}(p_i)$ and $p_i \stackrel{\sigma_i}{\Longrightarrow} p_i'$ when $C(p_1, \ldots, p_n) \stackrel{\sigma}{\Longrightarrow} C'(p_1', \ldots, p_n')$, we have that $p_i \stackrel{\sigma_i\mu_i}{\Longrightarrow} p_i''$ since $C'(p_1', \ldots, p_n') \stackrel{\mu}{\Longrightarrow} C''(p_1'', \ldots, p_n'')$. Then, by Proposition 5.6.1 and $(\sigma_i, \Phi_i') \in \mathcal{I}(p_i)$, we have $(\sigma_i\mu_i, \mu_i^{-1}\Phi_i') \in \mathcal{I}(p_i)$ since $\mu_i \in \mathcal{T}(p_i')$. Finally, by $p_i \equiv q_i$, we have $(\sigma_i\mu_i, \mu_i^{-1}\Phi_i') \in \mathcal{I}(q_i)$. □

Proposition 5.7.2 *There always exists a such $\mu \in Act^*$ satisfying the conditions of Proposition 5.7.1.*

Proof. This result can be easily obtained from the way constructing Φ_i' from Φ' in Strategy 5.5.1. To make $\varepsilon \notin \mu_1^{-1}\Phi_1'$, μ_1 should be a proper prefix of some $\delta_1' b$ in Φ_1'. Therefore, there should be some $\delta' a$ in Φ' corresponding with $\delta_1' b$ in Φ_1', which means that there must exist a proper prefix of $\delta' a$ satisfying the condition of μ in Proposition 5.7.1. □

5.8 Composing $\sigma\mu$ in $C(q_1, \ldots, q_n)$ from $\sigma_i\mu_i$

Continuing with the preceding subsection, we will show that, the obtained $\sigma_i\mu_i$ can also compose into $\sigma\mu$ in $C(q_1, \ldots, q_n)$, i.e., there exists some $C''(q_1'', \ldots, q_n'')$ such that $C(q_1, \ldots, q_n) \stackrel{\sigma\mu}{\Longrightarrow} C''(q_1'', \ldots, q_n'')$ and, at the same time, $q_i \stackrel{\sigma_i\mu_i}{\Longrightarrow} q_i''$.

Observe that, here, q_i'' does not need to be the q_i' obtained from $(\sigma_i\mu_i, \mu_i^{-1}\Phi_i') \in \mathcal{I}(q_i)$. In fact, we will show, in the following proposition, that, only when the ith argument of C'' is an active argument, can q_i'' be safely equivalent to q_i'.

To make it more clear, the following proposition will substitute $\sigma\mu$ with σ. Certainly, the obtained result will be also suitable for the above requirement.

Before that, we need one more definition on delay processes. Suppose that $\sigma \in \mathcal{T}(p)$ for some process p, then delay processes of $p \stackrel{\sigma}{\Longrightarrow}$ are those satisfying that 1) if $|\sigma| = 0$, then p itself is the delay process, and 2) if $|\sigma| \geq 1$, then let $\sigma = \sigma' a$ and delay processes are those processes p' such that $p \stackrel{\sigma}{\Longrightarrow} \stackrel{a}{\longrightarrow} p'$.

Proposition 5.8.1 *Let $\mathcal{L} = (\Sigma, \Psi)$ be a τDes-language, and $C(x_1, \ldots, x_n)$ be any context of n holes. Suppose that ζ and ξ are any two closed Σ-substitution mapping x_i into p_i and q_i, respectively. If for all $1 \leq i \leq n$, $p_i \sqsubseteq_t q_i$, then*

1. *for any trace $\sigma \in \mathcal{T}(C(p_1, \ldots, p_n))$ and some context C' such that $C(p_1, \ldots, p_n) \stackrel{\sigma}{\Longrightarrow} C'(p_1', \ldots, p_n')$, there exist $q_1', \ldots, q_n'$ such that $C(q_1, \ldots, q_n) \stackrel{\sigma}{\Longrightarrow} C'(q_1', \ldots, q_n')$, and*
2. *if there exists a patience ruloid for the ith argument of context C' then q_i' can be any process such that $q_i \stackrel{\sigma_i}{\Longrightarrow} q_i'$, and if there does not exist a patience ruloid for the ith argument of context C' then q_i' can be any delay processes of $q_i \stackrel{\sigma_i}{\Longrightarrow}$, where σ_i is obtained by decomposing σ into the weak traces of subprocess p_i.*

Proof. Suppose that σ is a trace of $C(p_1, \ldots, p_n)$, and $C(p_1, \ldots, p_n) \stackrel{\sigma}{\Longrightarrow} C'(p_1', \ldots, p_n')$. By Proposition 5.5.1, when $C(p_1, \ldots, p_n) \stackrel{\sigma}{\Longrightarrow} C'(p_1', \ldots, p_n')$, we have $p_i \stackrel{\sigma_i}{\Longrightarrow} p_i'$ for all $1 \leq i \leq n$. Then, by $p_i \sqsubseteq_t q_i$, we have $q_i \stackrel{\sigma_i}{\Longrightarrow}$ for all $1 \leq i \leq n$.

For $C(p_1, \ldots, p_n) \stackrel{\sigma}{\Longrightarrow} C'(p_1', \ldots, p_n')$, we have $C(p_1, \ldots, p_n) = C_0(p_{10}, \ldots, p_{n0}) \stackrel{\alpha_1}{\longrightarrow} C_1(p_{11}, \ldots, p_{n1}) \stackrel{\alpha_2}{\longrightarrow} \ldots \stackrel{\alpha_m}{\longrightarrow} C_m(p_{1m}, \ldots, p_{nm}) = C'(p_1', \ldots, p_n')$, where $\forall 1 \leq j \leq m : \alpha_j \in Act \cup \{\tau\}$ and $\sigma' = \alpha_1 \ldots \alpha_m$ is equivalent to σ if all its τ transitions are omitted.

Suppose that the sequence of plain ruloids applied in the above procedure is $r_1 r_2 \dots r_k$. It is enough to show that $C(q_1, \dots, q_n)$ can also apply ruloids $r_1 r_2 \dots r_k$ in the same order, and $C(q_1, \dots, q_n) \stackrel{\sigma}{\Longrightarrow} C'(q'_1, \dots, q'_n)$ for some $q'_1, \dots, q'_n$. Furthermore, if there exists a patience ruloid for the ith argument of context C' then q'_i can be any process such that $q_i \stackrel{\sigma_i}{\Longrightarrow} q'_i$, and if there does not exist a patience ruloid for the ith argument of context C' then q'_i can be any delay processes of $q_i \stackrel{\sigma_i}{\Longrightarrow}$.

We will prove it by making an induction on k.

1) $k = 0$. Then, $C = C'$ and only patience ruloids are applied when $C(p_1, \dots, p_n) \stackrel{\sigma}{\Longrightarrow} C'(p'_1, \dots, p'_n)$ and thus $\sigma = \tau^*$. By Proposition 5.5.1, $\sigma_i = \tau^*$ for all $1 \leq i \leq n$. Therefore, there must exist $q'_i, \dots, q'_n$ such that $C(q_1, \dots, q_n) \stackrel{\tau^*}{\Longrightarrow} C'(q'_1, \dots, q'_n)$ since an extreme possibility is that $q_i \equiv q'_i$ for all $1 \leq i \leq n$. Now, if there exists a patience ruloid for the ith argument of context C' then q'_i can be any process such that $q_i \stackrel{\tau^*}{\longrightarrow} q'_i$ by the soundness property of ruloids and the definition of patience ruloids. On the other hand, if there does not exist a patience ruloid for the ith argument of context C' then q'_i can be q_i.

2) Assume that, when $k = m - 1$ with $m \geq 1$, the above statement holds.

3) For $k = m$, suppose that, $C(p_1, \dots, p_n) \stackrel{\sigma}{\Longrightarrow} C''(p''_1, \dots, p''_n)$ and $C''(p''_1, \dots, p''_n) \stackrel{\delta}{\Longrightarrow} C'(p'_1, \dots, p'_n)$, where the first $k - 1$ plain ruloids of $r_1 r_2 \dots r_k$ are applied when $C(p_1, \dots, p_n) \stackrel{\sigma}{\Longrightarrow} C''(p''_1, \dots, p''_n)$ and the kth plain ruloid is applied when $C''(p''_1, \dots, p''_n) \stackrel{\delta}{\Longrightarrow} C'(p'_1, \dots, p'_n)$.

By Proposition 5.5.1, there exist σ_i, δ_i for all $1 \leq i \leq n$ such that $p_i \stackrel{\sigma_i}{\Longrightarrow} p''_i$ and $p''_i \stackrel{\delta_i}{\Longrightarrow} p'_i$. By $p_i \sqsubseteq_t q_i$, we have $q_i \stackrel{\sigma_i \delta_i}{\Longrightarrow}$.

Then, by the induction hypothesis, $C(q_1, \dots, q_n)$ can also apply the first $k - 1$ ruloids and reaches $C''(q''_1, \dots, q''_n)$. Moreover, if there exists a patience ruloid for the ith argument of context C'' then q''_i can be any process such that $q_i \stackrel{\sigma_i}{\Longrightarrow} q''_i$, and if there does not exist a patience ruloid for the ith argument of context C'' then q''_i can be any delay process of $q_i \stackrel{\sigma_i}{\Longrightarrow}$.

Furthermore, for all $1 \leq i \leq n$, let q''_i be any delay process of $q_i \stackrel{\sigma_i}{\Longrightarrow}$ such that $q_i \stackrel{\sigma_i}{\Longrightarrow} q''_i \stackrel{\delta_i}{\Longrightarrow}$. There always exists a such q''_i since $q_i \stackrel{\sigma_i \delta_i}{\Longrightarrow}$.

Suppose that the kth ruloid r_k is in form (3). Then, by the definition of the τDes-format, all arguments in I have corresponding patience ruloids since they are all active arguments of C'' by Definition 5.2.3. Therefore, by the soundness property of the ruloids, we may apply the patience ruloids for the arguments in I and obtain $C''(q''_1, \dots, q''_n) \stackrel{\tau^*}{\Longrightarrow} C''(q'''_1, \dots, q'''_n)$, such that $q'''_i \equiv q''_i$ if $i \notin I$ and $q'''_i \stackrel{\delta_i}{\longrightarrow}$ if $i \in I$. Then, also by the soundness property of the ruloids, ruloid r_k will be applied and $C''(q'''_1, \dots, q'''_n) \stackrel{\delta}{\longrightarrow} C'(q''''_1, \dots, q''''_n)$, where $q''''_i \equiv q''_i$ if $i \notin I$ and q''''_i is any process satisfying $q'''_i \stackrel{\delta_i}{\longrightarrow} q''''_i$ if $i \in I$.

Now, we can see that q''''_i is indeed a delay process of $q_i \stackrel{\sigma_i \delta_i}{\Longrightarrow}$.

Finally, if there exists a patience ruloid for the ith argument of context C' then q''''_i may evolve into any process q'_i such that $q''''_i \stackrel{\tau^*}{\Longrightarrow} q'_i$ and thus q'_i may be any process such that $q_i \stackrel{\sigma_i \delta_i}{\Longrightarrow} q'_i$. On the other hand, if there does not exist a patience ruloid for the ith argument of context C' then let q'_i be q''''_i, and thus q'_i is any delay process of $q_i \stackrel{\sigma_i \delta_i}{\Longrightarrow}$. □

5.9 Refusal of Φ'

Though, as stated in the preceding subsection, not all q_i'' can be equivalent to q_i', we can find a variant of μ, denoted as μ', such that $(\sigma\mu', \mu'^{-1}\Phi') \in \mathcal{I}(C(q_1, \ldots, q_n))$. By this result, we can claim that, there always be some μ such that $(\sigma\mu, \mu^{-1}\Phi') \in \mathcal{I}(C(q_1, \ldots, q_n))$, which will conclude the proof of Theorem 5.4.1 and thus Theorem 5.3.1, i.e., the meta theorem.

In the following, we will provide a strategy to obtain a such μ'.

Strategy 5.9.1 *Suppose that* $C(q_1, \ldots, q_n) \stackrel{\sigma\mu}{\Longrightarrow}$, *and* $(\sigma_i\mu_i, \mu_i^{-1}\Phi_i') \in \mathcal{I}(q_i)$. *Then, there must exist a* $\delta' a \in \Phi'$ *such that* μ *is a prefix of* δ' *and* δ' *is not a proper prefix of any* $\delta'' b \in \Phi'$. *Let* μ' *be* δ'.

Proposition 5.9.1 *Strategy 5.9.1 will always obtain some* μ'.

Proof. By Section 5.7, we have that $\varepsilon \notin \mu^{-1}\Phi'$. Therefore, μ should be a proper prefix of some $\delta' a$ in Φ', and thus μ should be a prefix of some δ' such that there exists some $\delta' a \in \Phi'$.

Observe that, there may exist several $\delta' a \in \Phi'$ satisfying that μ is a prefix of δ'. Therefore, in these $\delta' a$, there will always exist some $\delta' a$ satisfies δ' is not a proper prefix of any δ'' with $\delta'' b \in \Phi'$ and μ is a prefix of δ''. □

Proposition 5.9.2 *Let* μ' *be the one obtained by Strategy 5.9.1. We can conclude that* $\forall \nu \in \mu'^{-1}\Phi : |\nu| = 1$.

Proof. It can be easily obtained from the fact that $\delta' a \in \Phi'$, μ' is δ' and δ' is not a proper prefix of some $\delta'' b \in \Phi'$. □

The following proposition provides a critical way to show that, if only the set of next observable actions are concerned, then patience rules for non-active arguments are indeed not necessary.

Proposition 5.9.3 *Let* $\mathcal{L} = (\Sigma, \Psi)$ *be an* τ*Des-language, and* $C(x_1, \ldots, x_n)$ *be any context of n holes. Suppose that* ζ *is any closed* Σ*-substitution mapping* x_i *into* p_i. *If the ith argument is not an active argument of* $C(x_1, \ldots, x_n)$ *and* $p_i \stackrel{\tau^*}{\longrightarrow} p_i'$, *then* $\mathcal{T}(C(p_1, \ldots, p_i, \ldots, p_n), 1) = \mathcal{T}(C(p_1, \ldots, p_i', \ldots, p_n), 1)$.

Proof. Without loss of generality, suppose that $p = C(p_1, \ldots p_i, \ldots, p_n)$ and $q = C(p_1, \ldots, p_i', \ldots, p_n)$, where C is any context of n holes in the language $\mathcal{L}$. Let $A_1 = \{a \in Act | p \stackrel{a}{\Longrightarrow}\}$ and $A_2 = \{a \in Act | p \stackrel{a}{\Longrightarrow}\}$. We need to prove $A_1 = A_2$. Consider the next ruloid which will be applied.

If the next ruloid is a patience ruloid, then it should be a patience ruloid for active argument, since $\mathcal{L}$ is an τDes-language. However, applying the patience ruloid will not produce observable actions for $C(p_1, \ldots p_i, \ldots, p_n)$ and $C(p_1, \ldots, p_i', \ldots, p_n)$. Because the ith argument is not an active argument, $C(p_1, \ldots p_i, \ldots, p_j, \ldots, p_n) \stackrel{\tau}{\longrightarrow} C(p_1, \ldots p_i, \ldots, p_j', \ldots, p_n)$ and $C(p_1, \ldots p_i', \ldots, p_j, \ldots, p_n) \stackrel{\tau}{\longrightarrow} C(p_1, \ldots p_i', \ldots, p_j', \ldots, p_n)$ when the jth argument of context C is an active argument and $p_j \stackrel{\tau}{\longrightarrow} p_j'$. Now, it is enough to consider the set of next observable actions of $C(p_1, \ldots p_i, \ldots, p_j', \ldots, p_n)$ and $C(p_1, \ldots p_i', \ldots, p_j', \ldots, p_n)$.

If the next ruloid is a plain ruloid, then it should not be a ruloid with τ conclusion, since $\mathcal{L}$ is an τDes-language. Suppose that the applied ruloid r is in form (3), then the ith argument is not in I since it is not an active argument.

Therefore, by the soundness property of the ruloids, the p_i will not be fired when applying the ruloid r. Furthermore, since p and q are only different in p_i and p'_i, we have $A_1 = A_2$. □

Finally, we will show that, the obtained μ' will make $(\sigma\mu', \mu'^{-1}\Phi') \in \mathcal{I}(C(q_1, \ldots, q_n))$.

Proposition 5.9.4 $(\sigma\mu', \mu'^{-1}\Phi') \in \mathcal{I}(C(q_1, \ldots, q_n))$.

Proof. First, we will show that $\sigma\mu' \in \mathcal{T}(C(q_1, \ldots, q_n))$.

By Strategy 5.4.1, for all $\delta' a \in \Phi'$, $\delta' \in \mathcal{T}(C'(p'_1, \ldots, p'_n))$, where $C(p_1, \ldots, p_n) \stackrel{\sigma}{\Longrightarrow} C'(p'_1, \ldots, p'_n)$. Then, by Strategy 5.9.1, μ' is some δ' with $\delta' a \in \Phi'$. Therefore, $\mu' \in \mathcal{T}(C'(p'_1, \ldots, p'_n))$, and thus $\sigma\mu' \in \mathcal{T}(C(p_1, \ldots, p_n))$.

Then, by Proposition 5.8.1, $p_1 \sqsubseteq_{st} q_1$ and $p_i \equiv q_i$, we have that $\sigma\mu' \in \mathcal{T}(C(q_1, \ldots, q_n))$.

Second, we need to show that, there exists some $C''(q''_1, \ldots., q''_n)$ such that $C(q_1, \ldots, q_n) \stackrel{\sigma\mu'}{\Longrightarrow} C''(q''_1, \ldots., q''_n)$ and $\mathcal{T}(C''(q''_1, \ldots., q''_n)) \cap \mu'^{-1}\Phi' = \emptyset$.

By Strategy 5.9.1, μ is a prefix of μ'. Therefore, after decomposing $\sigma\mu'$, $\sigma_i\mu_i$ should be also a prefix of $\sigma_i\mu'_i$. Then, by Proposition 5.6.3, $(\sigma_i\mu'_i, \mu'^{-1}_i\Phi'_i)$ is also a impossible future pair of subprocess q_i.

By the definition of impossible future pair, $(\sigma_i\mu'_i, \mu'^{-1}_i\Phi'_i) \in \mathcal{I}(q_i)$ means that, there exists q'_i such that $q_i \stackrel{\sigma_i\mu'_i}{\Longrightarrow} q'_i$ and $\mathcal{T}(q'_i) \cap \mu'^{-1}_i\Phi'_i = \emptyset$.

Composing $\sigma_i\mu'_i$ using Proposition 5.8.1, we have that $C(q_1, \ldots, q_n) \stackrel{\sigma\mu'}{\Longrightarrow} C''(q''_1, \ldots, q''_n)$ for some context C'' and process $C''(q''_1, \ldots, q''_n)$.

Like we have discussed at the beginning of Section 5.8, q''_i cannot always be safely equivalent to q'_i, where $q_i \stackrel{\sigma_i\mu'_i}{\Longrightarrow} q'_i$ and $\mathcal{T}(q'_i) \cap \mu'^{-1}_i\Phi'_i = \emptyset$. By Proposition 5.8.1, only when the ith argument of C'' is active, can q''_i be safely equivalent to q'_i.

Fortunately, by Proposition 5.9.2, $\forall \nu \in \mu'^{-1}\Phi : |\nu| = 1$, we need only consider the set of next observable actions of $C''(q''_1, \ldots., q''_n)$. And by Proposition 5.9.3, the set of next actions of $C''(q''_1, \ldots, q''_n)$ and $C''(q'_1, \ldots, q'_n)$ are equivalent. As a result, the set of next refused actions of them are also equivalent.

Now, by $\mathcal{T}(q'_i) \cap \mu'^{-1}_i\Phi'_i = \emptyset$, we have $\{a \in Act | q'_i \stackrel{a}{\Longrightarrow}\} \subseteq Act - \mu'^{-1}_i\Phi'_i$. On the other hand, by Strategy 5.5.1, $\{a \in Act | \exists p''_i : p'_i \stackrel{\mu'_i}{\Longrightarrow} p''_i \wedge p''_i \stackrel{a}{\Longrightarrow}\} \cup \mu_i^{-1}\Phi'_i = Act$. Therefore, $\{a \in Act | q'_i \stackrel{a}{\Longrightarrow}\} \subseteq \{a \in Act | \exists p''_i : p'_i \stackrel{\mu'_i}{\Longrightarrow} p''_i \wedge p''_i \stackrel{a}{\Longrightarrow}\}$, which will make $C''(q'_1, \ldots, q'_n) \subseteq \mu'^{-1}\mathcal{T}(C(p'_1, \ldots, p'_n))$.

Finally, by $\mathcal{T}(C(p'_1, \ldots, p'_n)) \cap \Phi' = \emptyset$, we have $\mathcal{T}(C''(q'_1, \ldots, q'_n)) \cap \mu'^{-1}\Phi' = \emptyset$ and thus $\mathcal{T}(C''(q''_1, \ldots, q''_n)) \cap \mu'^{-1}\Phi' = \emptyset$. □

5.10 Meta Theorem

In this subsection, we will make a review on the proof of meta theorem by providing a proof sketch. The formal proofs of each steps have been shown in the previous subsections. For clarity, we copy the statement of meta theorem to here.

Theorem 5.10.1 *Let $\mathcal{L}$ be a τDes-language and C be any context of n holes. ζ and ξ are two closed Σ-substitutions mapping x_i into p_i and q_i, respectively. If $\forall 1 \leq i \leq n : p_i \sqsubseteq_{st} q_i$, then $C(p_1, \ldots, p_n) \sqsubseteq_{st} C(q_1, \ldots, q_n)$.*

Proof Sketch.

1. By the definition of should testing preorder, it is equivalent to prove that if $(\sigma, \Phi) \in \mathcal{I}(C(p_1, \ldots, p_n))$ then, there exists some $\mu \in Act^*$ such that $(\sigma\mu, \mu^{-1}\Phi) \in \mathcal{I}(C(q_1, \ldots, q_n))$ and $\varepsilon \notin \mu^{-1}\Phi$. By the transitivity of preorder relation, it is further equivalent to assume that $p_1 \sqsubseteq_{st} q_1$ and $\forall 2 \leq i \leq n : p_i \equiv q_i$, as stated in Theorem 5.3.2.

2. Theorem 5.3.2 holds if Theorem 5.4.1 holds. This is obtained by a strategy, i.e., Strategy 5.4.1, in which a specific Φ' is constructed. We show that if there exists some μ such that $(\sigma\mu, \mu^{-1}\Phi') \in \mathcal{I}(C(q_1, \ldots, q_n))$ then we have $(\sigma\mu, \mu^{-1}\Phi) \in \mathcal{I}(C(q_1, \ldots, q_n))$. With this result, the proof is reduced to find a specific μ satisfying that $(\sigma\mu, \mu^{-1}\Phi') \in \mathcal{I}(C(q_1, \ldots, q_n))$, which is fulfilled in the following steps.

3. By Proposition 5.5.1 and Strategy 5.5.1, (σ, Φ') can be decomposed into (σ_i, Φ'_i). Proposition 5.5.2 shows that $(\sigma_i, \Phi'_i) \in \mathcal{I}(p_i)$.

4. By $(\sigma_i, \Phi'_i) \in \mathcal{I}(p_i)$ and the definition of should testing preorder, there must exist some μ_i such that $(\sigma_i\mu_i, \mu_i^{-1}\Phi'_i) \in \mathcal{I}(p_i)$. For the case of $2 \leq i \leq n$, the hypothesis that $\forall 2 \leq i \leq n : p_i \equiv q_i$ implies that, μ_i can be any action sequence satisfying $\varepsilon \notin \mu_i^{-1}\Phi'_i$.

5. Section 5.7 shows that, we can always find some $\sigma\mu$ such that, after decomposing, we have the same $\sigma_1\mu_1$ as the above. Then, this $\sigma\mu$ is proved in Section 5.8 to be a weak trace of $C(q_1, \ldots, q_n)$. However, we cannot ascertain that this μ is the one we need to make $(\sigma\mu, \mu^{-1}\Phi') \in \mathcal{I}(C(q_1, \ldots, q_n))$, because patience rules for non-active arguments are not necessary in τDes-languages.

6. In Section 5.9, we show that, there exists a μ' such that μ is its prefix and $(\sigma\mu', \mu'^{-1}\Phi') \in \mathcal{I}(C(q_1, \ldots, q_n))$. □

6 Applications

6.1 BPA

First, let's examine BPA language [2], which is a simplified version of ACP language. The syntax of the BPA consists of two operators: alternative composition + and sequential composition $\cdot$. Their transition rules are those in Table 2.

To meet weak equivalences, adding patience rules are rather necessary. An intuitive way is to substitute the $a \in Act$ with $\alpha \in Act \cup \{\tau\}$. However, the rules produced by this way do not satisfy the τDes-format. The main problem is presented in Table 3, which contains some rules for + operator after substituting a with τ. These rules are not patience rules but have τ premises.

On the other hand, a simple counterexample is possible that the modified + operator is not invariant under should testing preorder. Let $p_1 = p_2 = a$, $q_1 = \tau b$ and $q_2 = b$. Then, $p_1 \sqsubseteq_{st} p_2$ and $q_1 \sqsubseteq_{st} q_2$. However, $p_1 + q_1 \not\sqsubseteq_{st} p_2 + q_2$, since $(\emptyset, \{b\}) \in \mathcal{I}(p_1 + q_1)$ but $\nexists \mu \in \{\varepsilon\} \cup \downarrow \{b\} - \{b\} : (\mu, \mu^{-1}\{b\}) \in \mathcal{I}(p_2 + q_2)$.

Generally, there are two ways in dealing with this problem.

Table 2: Operational rules for BPA operators ($a \in Act$) [2]

$$\frac{}{a \xrightarrow{a} \surd} \qquad \frac{x \xrightarrow{a} x'}{x+y \xrightarrow{a} x'} \quad \frac{y \xrightarrow{a} y'}{x+y \xrightarrow{a} y'} \quad \frac{x \xrightarrow{a} \surd}{x+y \xrightarrow{a} \surd} \quad \frac{y \xrightarrow{a} \surd}{x+y \xrightarrow{a} \surd}$$

$$\frac{x \xrightarrow{a} x'}{x \cdot y \xrightarrow{a} x' \cdot y} \quad \frac{x \xrightarrow{a} \surd}{x \cdot y \xrightarrow{a} y}$$

Table 3: Not patience rules after substituting a with τ, but they have τ premises

$$\frac{x \xrightarrow{\tau} x'}{x+y \xrightarrow{\tau} x'} \quad \frac{y \xrightarrow{\tau} y'}{x+y \xrightarrow{\tau} y'} \quad \frac{x \xrightarrow{\tau} \surd}{x+y \xrightarrow{\tau} \surd} \quad \frac{y \xrightarrow{\tau} \surd}{x+y \xrightarrow{\tau} \surd}$$

The first one is to introduce a slightly finer preorder $\sqsubseteq_{st}^{c}$, which strengthen the original preorder $\sqsubseteq_{st}$ with a stable preorder $\sqsubseteq_{stable}$, i.e., $\sqsubseteq_{st}^{c}$=$\sqsubseteq_{st} \cap \sqsubseteq_{stable}$. The stable preorder is defined as: $p \sqsubseteq_{stable} q$ iff $q \overset{\tau}{\nrightarrow} \Longrightarrow p \overset{\tau}{\nrightarrow}$. This way has been adopted by [7] for should testing preorder in a CCS-like language.

The second one is to adopt other operators to substitute the + operator. In fact, the internal choice operator and external choice operator of the CSP language will be sound for this goal. More specifically, operators $\boxplus, \oplus, \triangleright$ of language **B** are also used to substitute the + operator and the prefixing with τ. Using these operators, $ap+bq$, $\tau ap+\tau bp$ and $ap + \tau bq$ can be represented by $ap \boxplus bq$, $ap \oplus bp$ and $ap \triangleright bq$, respectively.

Note that, the right-hand-side argument of sequential composition operator $\cdot$ is neither an active argument nor a receiving argument. Therefore, no patience rules are needed for it.

6.2 ACP

Table 5 lists the additional rules for the ACP language, besides those rules of BPA in Table 3. Fortunately, it can be easily verified that, except for the left merge operator $\between$, by substituting all a, b, c with τ, the added rules are all patience rules.

The τ rule for the left merge operator is in form (5). We can see that, it is even not a patience rule. And we can borrow a counterexample in paper [29] to show that the left merge operator is not compositional under should testing preorder: $p_1 = \tau \cdot (\tau \cdot a + b)$ and $p_2 = \tau \cdot a + b$ are in should testing preorder relation, but $p_1 \between c$ and $p_2 \between c$ are not.

$$\frac{x \xrightarrow{\tau} x'}{x \between y \xrightarrow{\tau} x' \between y} \tag{5}$$

Table 4: Additional rules for ACP operators ($a, b, c \in Act$) [2]

$$\frac{x \xrightarrow{a} x'}{x\|y \xrightarrow{a} x'\|y} \quad \frac{y \xrightarrow{a} y'}{x\|y \xrightarrow{a} x\|y'} \quad \frac{x \xrightarrow{a} x\surd}{x\|y \xrightarrow{a} y} \quad \frac{y \xrightarrow{a} \surd}{x\|y \xrightarrow{a} x}$$

$$\frac{x \xrightarrow{a} x', y \xrightarrow{b} y'}{x\|y \xrightarrow{c} x'\|y'}\ \gamma(a,b) = c \quad \frac{x \xrightarrow{a} x', y \xrightarrow{b} \surd}{x\|y \xrightarrow{c} x'}\ \gamma(a,b) = c$$

$$\frac{x \xrightarrow{a} \surd, y \xrightarrow{b} y'}{x\|y \xrightarrow{c} y'}\ \gamma(a,b) = c \quad \frac{x \xrightarrow{a} \surd, y \xrightarrow{b} \surd}{x\|y \xrightarrow{c} \surd}\ \gamma(a,b) = c$$

$$\frac{x \xrightarrow{a} x'}{x\between y \xrightarrow{c} x'\|y} \quad \frac{x \xrightarrow{a} \surd}{x\between y \xrightarrow{c} y}$$

$$\frac{x \xrightarrow{a} x', y \xrightarrow{b} y'}{x|y \xrightarrow{c} x'\|y'}\ \gamma(a,b) = c \quad \frac{x \xrightarrow{a} x', y \xrightarrow{b} \surd}{x|y \xrightarrow{c} x'}\ \gamma(a,b) = c$$

$$\frac{x \xrightarrow{a} \surd, y \xrightarrow{b} y'}{x|y \xrightarrow{c} y'}\ \gamma(a,b) = c \quad \frac{x \xrightarrow{a} \surd, y \xrightarrow{b} \surd}{x|y \xrightarrow{c} \surd}\ \gamma(a,b) = c$$

$$\frac{x \xrightarrow{a} x'}{\vartheta(x) \xrightarrow{c} \vartheta(x')}\ a \notin H \quad \frac{x \xrightarrow{a} \surd}{\vartheta(x) \xrightarrow{c} \surd}\ a \notin H$$

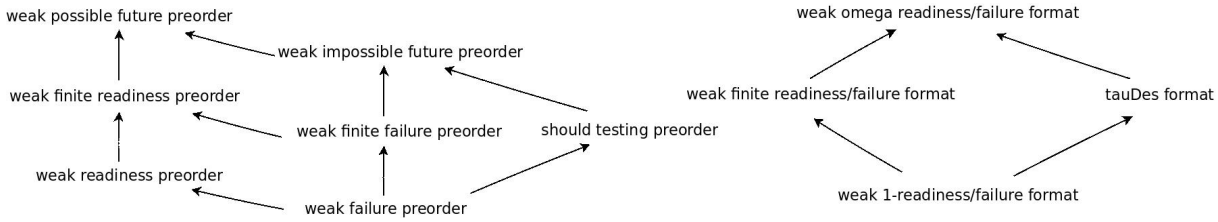

Figure 4: Several decorated trace preorders and their precongruence formats

7 Comparison with Previous Works

Figure 4 makes a simple review on current work and our previous works in [15, 16]. In [15], three congruence formats, i.e., weak 1-readiness format and weak ω-readiness format, are assigned to weak readiness equivalence and weak possible future equivalence, respectively. In [16], two congruence formats, i.e, weak 1-failure format and weak ω-failure format, are assigned to weak failure equivalence and weak impossible future equivalence, respectively. In fact, weak 1-readiness format is the same as weak 1-failure format and weak ω-readiness format is the same as weak ω-failure format.

The left graph in Figure 4 summarizes the different discrimination power of the above-mentioned weak equivalences. For convenience, should testing equivalence is used instead of should testing preorder. The arrows in the graph denote the "coarser than" relations between two related weak equivalences. More specifically, should testing equivalence is coarser than weak impossible future equivalence and is finer than weak failure equivalence.

As a custom in the area of SOS, a coarser equivalence/preorder has a tighter congruence/precongruence format [29]. Therefore, it is generally expected that τDes-format is tighter than weak ω-failure format and is looser than weak 1-failure format. The results in this paper and in [15, 16] justify this intuition in that, compared with τDes-format, weak ω-failure format further needs patience rules for receiving arguments but weak 1-failure format should exclude rules with τ-conclusion. The definition of rules with τ-conclusion are as follows.

Definition 7.0.1 *[15, 16] Let $\mathcal{L} = (\Sigma, \Psi)$ be a de Simone language, and f be a function symbol in Σ. A rule of the form $\frac{H}{f(x_1, \ldots, x_n) \xrightarrow{\tau} t}$ is called a rule with τ-conclusion, if it is not a patience rule and there exists at least one positive Σ-literal in H.*

The right graph in Figure 4 shows the above conclusion. The arrows denote the "looser than" relations between two related congruence/precongruence formats.

In our statements above, the 'tighter than' relation requires that, format A is tighter than format B iff, for any languages $\mathcal{L} = (\Sigma, \Psi)$ in format A, all transition rules in Ψ are also in format B. Note that, 'tighter than' relation is not the same as 'contained in' relation, which requires that, format A is contained in format B iff, any language $\mathcal{L} = (\Sigma, \Psi)$ in format A will also be in format B. It can be easily proved that, 'contained in' relation implies 'tighter than' relation, but not vice versa. Indeed, the τDes format is tighter than the weak ω-failure format, but is not contained in the weak ω-failure format.

8 Conclusions and Related Works

In the paper, we introduce a rule format for the should testing preorder to be precongruent. By the formal proof, we have shown the correctness of our views. Also, we have given its applications on the ACP language.

As another direct application of τDes-format, we can prospect its practical use on designing a plausible specification language for specifying and analyzing protocols. If the specification language is in the format, then two operations can be done safely. The first is that we can substitute a subcomponent of the specification with another subcomponent which is in preorder relation with the original subcomponent, and the obtained implementation is also in preorder relation with the specification. The second is that, for two set of processes which are in preorder relation correspondingly, we can composite them randomly using the operators in the language, and the obtained two higher-level processes are still in preorder relation.

As for the related works, we have noticed that equivalences/preorders in strong notion were paid more attentions than equivalences/preorders in weak notion. In fact, almost all classical strong equivalences/preorders, e.g., strong bisimulation and decorated trace preorders, have found their corresponding congruence/precongruence formats [1, 19].

However, less works have been done on the congruence/precongruence formats for weak equivalences/preorders.

As for the formats for weak bisimulation-like equivalences, [5] introduced congruence formats for (rooted) weak bisimulation and (rooted) branching bisimulation, [9] proposed RBB safe format for rooted branching bisimulation

which generalized the simple RBB cool format of [5] to the setting with negative premises and predicates. And recently, [29] discussed the rule formats for the weak bisimulation, delay bisimulation, η-bisimulation and branching bisimulation. On the other hand, with the way of ordered SOS, [24, 25, 26] proposed several formats and languages for (rooted) branching bisimulation and (rooted) eager bisimulation.

As for the formats for testing-theoretical-based weak equivalences/preorders and weak decorated trace preorders, [24] suggested τDes format, which is a subformat of ISOS format [23], to be a congruence format for testing equivalence. The authors [15, 16] have proposed precongruence formats for weak readiness/failure equivalence and weak possible/impossible future equivalence. But prior to this work, no rule formats have been presented to be a precongruence format for the should testing preorder.

Recently, another proof technique on congruence/precongruence formats based on decomposing of the subclass of modal formulas for some given equivalences/preorders have been studied [6, 10, 11]. Within their works, (bi)simulation-like equivalences and (strong) decorated trace preorders have found their corresponding congruence/precongruence formats. These formats are subformats of ready simulation format, i.e., ntyft/ntyxt format without lookahead. However, the problem whether this proof technique can be fitted into our works aiming at should testing preorder is still open.

References

[1] Luca Aceto, Willem Jan (Wan) Fokkink, and Chris Verhoef. Structural operational semantics. In Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors, *Handbook of Process Algebra, Chapter 3*, pages 197–292. Elsevier Science, Dordrecht, The Netherlands, 2001.

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

[3] Bloom, Istrail, and Meyer. Bisimulation can't be traced. *JACM: Journal of the ACM*, 42, 1995.

[4] Bard Bloom. *Ready simulation, bisimulation, and the semantics of CSS-like languages*. PhD thesis, Massachusetts Institute of Technology, 1989.

[5] Bard Bloom. Structural operational semantics for weak bisimulations. *Theor. Comput. Sci.*, 146(1&2):25–68, 1995.

[6] Bard Bloom, Willem Jan (Wan) Fokkink, and Robert Jan (Rob) van Glabbeek. Precongruence formats for decorated trace semantics. *ACM Transactions on Computational Logic*, 5(1):26–78, 2004.

[7] Ed Brinksma, Arend Rensink, and Walter Vogler. Fair testing. In Insup Lee and Scott A. Smolka, editors, *CONCUR*, volume 962 of *Lecture Notes in Computer Science*, pages 313–327. Springer, 1995.

[8] Robert de Simone. Higher-level synchronising devices in meije-sccs. *Theor. Comput. Sci.*, 37:245–267, 1985.

[9] Wan Fokkink. Rooted branching bisimulation as a congruence. *J. Comput. Syst. Sci.*, 60(1):13–37, 2000.

[10] Wan Fokkink, Rob van Glabbeek, and Paulien de Wind. Divide and congruence applied to [eta]-bisimulation. *Electronic Notes in Theoretical Computer Science*, 156(1):97–113, 2006.

[11] Wan Fokkink, Rob J. van Glabbeek, and Paulien de Wind. Divide and congruence: From decomposition of modalities to preservation of branching bisimulation. In Frank S. de Boer, Marcello M. Bonsangue, Susanne Graf, and Willem P. de Roever, editors, *FMCO*, volume 4111 of *Lecture Notes in Computer Science*, pages 195–218. Springer, 2005.

[12] Jan Friso Groote. Transition system specifications with negative premises. *Theor. Comput. Sci.*, 118(2):263–299, 1993.

[13] Jan Friso Groote and Frits W. Vaandrager. Structural operational semantics and bisimulation as a congruence (extended abstract). In Giorgio Ausiello, Mariangiola Dezani-Ciancaglini, and Simona Ronchi Della Rocca, editors, *ICALP*, volume 372 of *Lecture Notes in Computer Science*, pages 423–438. Springer, 1989.

[14] CAR Hoare. Communicating sequential processes. *Comm. ACM*, 21(8):666–677, August 1978.

[15] Xiaowei Huang, Li Jiao, and Weiming Lu. Rule formats for weak readiness equivalence and weak possible future equivalence. *The Computer Journal*, 2008.

[16] Xiaowei Huang, Li Jiao, and Weiming Lu. Weak parametric failure equivalences and their congruence formats. In James Harland and Prabhu Manyem, editors, *CATS*, volume 77 of *CRPIT*, pages 15–26. Australian Computer Society, 2008.

[17] R. Milner. A calculus of communicating systems. *LNCS*, 92, 1980.

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

[19] Mohammad Reza Mousavi, Michel A. Reniers, and Jan Friso Groote. Sos formats and meta-theory: 20 years after. *Theor. Comput. Sci.*, 373(3):238–272, 2007.

[20] Rocco De Nicola and Matthew Hennessy. Testing equivalences for processes. *Theor. Comput. Sci*, 34:83–133, 1984.

[21] G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.

[22] Arend Rensink and Walter Vogler. Fair testing. *Inf. Comput.*, 205(2):125–198, 2007.

[23] Irek Ulidowski. Equivalences on observable processes. In *LICS*, pages 148–159. IEEE Computer Society, 1992.

[24] Irek Ulidowski. Finite axiom systems for testing preorder and de simone process languages. *Theor. Comput. Sci.*, 239(1):97–139, 2000.

[25] Irek Ulidowski and Iain C. C. Phillips. Ordered sos process languages for branching and eager bisimulations. *Inf. Comput.*, 178(1):180–213, 2002.

[26] Irek Ulidowski and Shoji Yuen. Process languages with discrete relative time based on the ordered sos format and rooted eager bisimulation. *J. Log. Algebr. Program.*, 60-61:401–460, 2004.

[27] R.J. van Glabbeek. *The Linear Time – Branching Time Spectrum I. The Semantics of Concrete, Sequential Processes*, pages 3–99. Elsevier Science, 2001.

[28] Rob J. van Glabbeek. The linear time - branching time spectrum ii. In Eike Best, editor, *CONCUR*, volume 715 of *Lecture Notes in Computer Science*, pages 66–81. Springer, 1993.

[29] Rob J. van Glabbeek. On cool congruence formats for weak bisimulations. In Dang Van Hung and Martin Wirsing, editors, *ICTAC*, volume 3722 of *Lecture Notes in Computer Science*, pages 318–333. Springer, 2005.

[30] Rob J. van Glabbeek and Marc Voorhoeve. Liveness, fairness and impossible futures. In Christel Baier and Holger Hermanns, editors, *CONCUR*, volume 4137 of *Lecture Notes in Computer Science*, pages 126–141. Springer, 2006.