

# Weak Parametric Failure Equivalences and Their Congruence Formats

Xiaowei Huang<sup>1</sup>

Li Jiao<sup>2</sup>

Weiming Lu<sup>1</sup>

<sup>1</sup> Academy of Mathematics and System Science,  
Chinese Academy of Sciences, P.R. China.  
Email: xwhuang@amss.ac.cn

<sup>2</sup> State Key Laboratory of Computer Science, Institute of Software,  
Chinese Academy of Sciences, P.R. China.

## Abstract

Weak equivalences are important behavioral equivalences in the course of specifying and analyzing the reactive systems using process algebraic languages. In this paper, we propose a series of weak equivalences named weak parametric failure equivalences, which take two previously-known behavioral equivalences, i.e., the weak failure equivalence and the weak impossible future equivalence, as their special cases. More importantly, based on the idea of the structural operational semantics, a series of rule formats are further presented to congruence format for their corresponding weak parametric failure equivalences, i.e., a specific equivalence is further congruent in any languages satisfying its corresponding congruence format. This series of rule formats reflect the gradual changes in the weak parametric failure equivalences. We conclude that, when the weak parametric failure equivalences become coarser, their corresponding rule formats turn tighter.

*Keywords:* weak failure equivalence, rule formats, structural operational semantics

## 1 Introduction

When using process algebraic languages to specify the distributed systems, a suitable semantic equivalence is usually necessary for reasoning and analyzing. There exist various semantic equivalences to be applied in various situations. An equivalence relation is reflexive, symmetric, and transitive.

Behavioral equivalences are based on the observability and thus equivalences may differ by the notions of observability (van Glabbeek, 2001, 1993). A natural classification of behavioral equivalences is that a given behavioral equivalence may be strong or weak. Their difference mostly exists in the ways of dealing with the internal transitions, which are generally denoted as  $\tau$  transitions. Strong equivalences regard  $\tau$  transitions the same as the observable actions. Weak equivalences, on the other hand, suppose them unobserved by the outer-world. Therefore, 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. In this paper, we will focus on the weak equivalences.

Among various weak semantic equivalences, the weak failure equivalence and the weak impossible future equivalence are two interesting semantic equivalences. The weak impossible future equivalence is strictly finer than the weak failure equivalence. The weak failure semantic is usually denoted, by its denotational characterization, as a set of weak failure pairs. Two processes are weak failure equivalent iff they have the same set of weak failure pairs. Likewise, the weak impossible future semantic is usually denoted as a set of weak impossible future pairs and two processes are weak impossible future equivalent iff they have the same set of weak impossible future pairs.

Looking into the two pairs, we find that their first parameters both express the abilities: some process  $p$  executes a sequence of observable actions and evolves into another process  $p'$ . The difference exists in their second parameters. The second parameter of a weak failure pair is a set of actions which are not enabled by  $p'$ , and the second parameter of a weak impossible future pair is a set of action sequences which are not enabled by  $p'$ .

Based on these observations, we define a series of weak equivalences, called weak parametric failure equivalences. Like the above two weak equivalences, a weak  $i$ -failure pair with  $i \in \mathbb{N} \cup \{\omega\}$  is only different in its second parameter with the weak failure pair, where  $\mathbb{N}$  is the set of natural numbers and  $\omega$  is the cardinality of  $\mathbb{N}$ . Its second parameter is a set of action sequences which are not enabled by  $p'$  and the lengths of these action sequences do not exceed  $i$ . Therefore, the plain weak failure equivalence is the weak 1-failure equivalence in our framework and the weak impossible future equivalence is the weak  $\omega$ -failure equivalence. Furthermore, with the increasing of the parametric  $i$ , the weak  $i$ -failure equivalence becomes finer.

Structural Operational Semantics (SOSs) (Plotkin, 2004) have been widely used in defining the meanings of the operators in various process algebraic language, such as CCS (Milner, 1989) and ACP (Baeten, 1990). The main idea of SOSs is: at first, each process is represented by a closed term and has some out-going transitions to communicate with outer world; then, these processes are coordinated by some specified rules, which are called transition rules, to get a higher-level process. As a result, the out-going transitions of this higher-level process are determined by the out-going transitions of its sub-processes.

Transition System Specifications (TSSs) (Groote, 1992), which borrowed from logic programming, form a theoretical basis for SOSs. By imposing some syntactic restrictions on TSS, 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 is whether or not a behavioral equivalence is congruent for a TSS in this rule format. A congruence is an essential equivalent property - namely that we can 'substitute equals for equals' (Milner, 1999). We will use language as an alias of the TSS. Up to now, some rule formats have been presented to meet the behavioral equivalences, for examples, GSOS format (Bloom, 1995) and ntyft/nxyft format (Groote, 1993) have been proved to

be congruent on strong bisimulation, de Simone (Simone, 1985) format was proved to be congruent on failure equivalence, and so on.

However, more works have been done on pursuing a suitable rule format for a given strong equivalence. On the contrary, much less attentions were paid on the rule formats for weak equivalences. More specifically, to our knowledge, no congruence formats have been presented for the weak failure equivalence or the weak impossible future equivalence.

In the paper, we will propose a series of rule formats for the newly defined weak parametric failure equivalences. In fact, weak 1-failure format is presented for the weak 1-failure equivalence, weak finite failure format is for the weak  $i$ -failure equivalences with  $1 < i < \omega$ , and weak  $\omega$ -failure format is for the weak  $\omega$ -failure equivalence. Then, we prove that the weak parametric failure equivalence can be preserved after composition if the language is in its corresponding rule formats, i.e., these rule formats are all congruence formats for their corresponding equivalences.

Here, we want to sketch out two critical points in pursuing these rule formats:

The first critical point is on the feasibility of allowing the rules with  $\tau$ -conclusion. Rules with  $\tau$ -conclusion are an important class of rules in classical process algebraic languages, notable examples include hiding operator of CSP and parallel composition operator of CCS. However, not all behavioral equivalences can be preserved under these rules, as is pointed out in Rensink and Vogler (Rensink, 2007) that the acceptance testing equivalence may not be preserved under the hiding operator. In this paper, we will take a close look into these rules. In fact, the weak  $i$ -failure equivalences with  $i < \omega$  may not be preserved under these rules, but the weak  $\omega$ -failure equivalence, i.e., the impossible future equivalence, can survive these rules.

The second critical point is whether or not the patience rules for receiving arguments are prerequisite in the rule formats for a given weak parametric failure equivalence. Patience rules, which are used to smooth the evolvment of  $\tau$  transitions of subprocesses, are usually necessary in rule formats for weak equivalences. However, since patience rules are defined in accordance with the arguments of an operator, they can be divided into three classes: patience rules for active arguments, patience rules for receiving arguments and patience rules for other arguments. Though patience rules for active arguments are generally needed, patience rules for receiving arguments are not necessary for some rule formats. We find that, for the weak 1-failure format, patience rules for receiving arguments are not prerequisite by the help of the exclusion of rules with  $\tau$ -conclusion. On the other hand, they are prerequisite for the weak  $i$ -failure format with  $i > 1$ .

As a result, the weak finite failure format is tighter than the weak  $\omega$ -failure format because the rules with  $\tau$ -conclusion should be excluded from the language in the weak finite failure format, and the weak 1-failure format is tighter than the weak finite failure format since it may further exclude the patience rules for receiving arguments from the language. Therefore, we can conclude that, when the weak parametric failure equivalences become coarser, their corresponding rule formats turn tighter.

Finally, we want to say more on the newly-proposed weak  $i$ -failure equivalences with  $1 < i < n$ . In fact, we have not found their niche applications, though they can be used in most applications of the 1-failure equivalences. The reasons that we introduce these intermediate weak equivalences are that

- 1) they can smooth the changes between the weak 1-failure equivalence and the weak  $\omega$ -failure equivalence,
- 2) their congruence format is also an intermediate format between the weak 1-failure format and the weak  $\omega$ -failure format, and

- 3) most importantly, we want to make clear the technical reasons why there exist differences between the weak 1-failure format and the weak  $\omega$ -failure format. Take it more concrete, from the weak  $\omega$ -failure format to the weak 1-failure format, the reason why the rules with  $\tau$  conclusion are excluded is that the parameter  $i$  degrades from infinite to finite, and the reason why the patience rules for receiving arguments are not necessary is that, no matter how they are presented in the language, only the set of next one, but not next finite or infinite, observable actions remains unique.

The structure of this paper is: in the section 2, we will introduce some preliminaries, mainly on the behavioral equivalences and the rule formats in Structural Operational Semantics. Then in Section 3, we will put forward the formal definitions on the weak parametric failure equivalences. Intuitive motivations on their rule formats will be exhibited with examples in Section 4. Section 5 is devoted to the formal definitions of the rule formats, and the proofs on the congruence theorems. And then, in Section 6, we will conclude the paper.

## 2 Preliminaries on Behavioral Equivalences and Rule Formats

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 usually use  $a, b, \dots$  to range over the actions in  $Act$ , and use  $A, B, \dots$  to range over subsets of actions in  $Act$ .  $\tau$  is generally used to denote the internal action which can not be observed by the outer world, and we use  $\alpha, \beta, \dots$  to range over the actions in  $Act \cup \{\tau\}$ .  $\delta, \mu, \sigma, \dots$  is to range over the sequences of actions.  $\Phi, \Psi, \dots$  is to range over the sets of sequences.  $p, q, \dots$  will be used to represent processes.

Any behavioral semantics of some process  $p$  can be characterized by a function  $O(p)$  (van Glabbeek, 2001).  $O(p)$  constitutes the observable behaviors of  $p$ . The equivalence relation  $\sim_O$  can be defined by  $p \sim_O q \iff O(p) = O(q)$ . The readers are referred to van Glabbeek (van Glabbeek, 2001, 1993) for comprehensive reviews of the behavioral equivalences.

SOS has been widely accepted as a tool to define operational semantics of processes. A TSS is a formalization of SOS (Plotkin, 2004). The readers are referred to Aceto, Fokkink and Verhoef (Aceto, 2001) for a comprehensive review on SOS.

**Definition 2.1 (Aceto, 2001)** Let  $V = \{x_1, x_2, \dots\}$  be a set of variables. A signature  $\Sigma$  is a collection of function symbols  $f \notin V$  equipped with a function  $ar : \Sigma \rightarrow N$ . The set  $\mathbb{T}(\Sigma)$  of terms over a signature  $\Sigma$  is defined recursively by: 1)  $V \subseteq \mathbb{T}(\Sigma)$ ; 2) if  $f \in \Sigma$  and  $t_1, \dots, t_{ar(f)} \in \mathbb{T}(\Sigma)$ , then  $f(t_1, \dots, t_{ar(f)}) \in \mathbb{T}(\Sigma)$ .

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

In the paper, we will use  $p, q, \dots$  to range over the closed terms, and call them processes.

**Definition 2.2** A positive  $\Sigma$ -literal is an expression  $t \xrightarrow{\alpha} t'$  and a negative  $\Sigma$ -literal is an expression  $t \xrightarrow{\alpha} t'$  with  $t, t' \in \mathbb{T}(\Sigma)$  and  $\alpha \in Act \cup \{\tau\}$ . A transition rule over  $\Sigma$  is an expression of the form  $\frac{H}{C}$  with  $H$  a set of  $\Sigma$  literals (the premises of the rule) and  $C$  a positive  $\Sigma$ -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  $\alpha$ .

A TSS, written as  $(\Sigma, \Psi)$ , consists of a signature  $\Sigma$  and a set  $\Psi$  of transition rules over  $\Sigma$ . A TSS is positive if the premises of its rules are positive. In the paper, we often use language as an alias of the TSS.

**Definition 2.3** Let  $\Sigma$  be a signature. A context  $C$  of  $n$  holes over  $\Sigma$  is simply a term in  $\mathbb{T}(\Sigma)$  in which  $n$  variables occur, each variable only once. If  $t_1, \dots, t_n$  are terms over  $\Sigma$ , then  $C(t_1, \dots, t_n)$  denotes the term obtained by substituting  $t_1$  for the first variable occurring in  $C$ ,  $t_2$  for the second variable occurring, etc. Thus, if  $x_1, \dots, x_n$  are all different variables, then  $C(x_1, \dots, x_n)$  denotes a context of  $n$  holes in which  $x_i$  is the  $i$ th occurring variable.

Then, we can give the definition on the congruence of an equivalence in a language.

**Definition 2.4** Let  $\mathcal{L} = (\Sigma, \Psi)$  be a language. An equivalence relation  $\sim$  is congruent on language  $\mathcal{L}$  iff  $\forall i \in \{1, \dots, n\} : p_i \sim q_i \implies C(p_1, \dots, p_n) \sim C(q_1, \dots, q_n)$  for any context  $C$  of  $n$  holes in language  $\mathcal{L}$ , where  $p_i$  and  $q_i$  are closed terms, i.e., processes, over  $\Sigma$ .

**Definition 2.5** Let  $\Sigma$  be a signature. A transition relation over  $\Sigma$  is a relation  $\text{Tr} \subseteq \mathbb{T}(\Sigma) \times \text{Act} \cup \{\tau\} \times \mathbb{T}(\Sigma)$ . Element  $(p, \alpha, p')$  of a transition relation is written as  $p \xrightarrow{\alpha} p'$ .

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

Furthermore, for an action sequence  $\delta = \alpha_1 \dots \alpha_n$ , if there exist  $p_1, \dots, p_n \in \mathbb{T}(\Sigma)$  such that  $p \xrightarrow{\alpha_1} p_1 \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_n} p_n$ , then we call  $\delta$  a trace of  $p$ , denoted as  $p \xrightarrow{\delta}$  or  $p \xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_n}$ .

In weak semantics, the weak transition relations and the weak traces are also needed to be defined. Let  $p$  be a process, we write  $p \xRightarrow{a}$  iff  $p \xrightarrow{\tau^*} \xrightarrow{a} \xrightarrow{\tau^*}$ , where  $\tau^*$  denotes any number of internal transitions. Hence, for an observable action sequence  $\delta = a_1 \dots a_n$ ,  $p \xRightarrow{\delta}$  iff  $p \xrightarrow{a_1} \dots \xrightarrow{a_n}$ .

By imposing some syntactic constraints on TSS's, we will obtain the so-called rule formats with some properties on their induced operational semantics. Within these properties, it is specially important that whether a behavioral equivalence can be preserved in the languages with this format. Some rule formats have been proposed to meet the numerous behavioral equivalences, such as GSOS format, de Simone format, ntyft/nxyft format, etc. The readers are referred to Mousavi, Reniers and Groote (Groote, 2007) for a latest review on the rule formats.

The de Simone language will be employed as our starting point in retrieving the rule formats for the weak parametric failure equivalences.

**Definition 2.6 (Simone, 1985)** Let  $\Sigma$  be a signature. A transition rule  $r$  is in de Simone format if it has the form  $\frac{\{x_i \xrightarrow{a_i} y_i\}_{i \in I}}{f(x_1, \dots, x_{ar(f)}) \xrightarrow{a} t}$ , where  $I \subseteq \{1, \dots, ar(f)\}$  and the variables  $x_i$  and  $y_i$  are all distinct and the only variables occurring in  $r$ . 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.

Below, two special classes of rules are defined. They will be discussed in the paper. The first class is the patience rules, and the second class is the rules with  $\tau$ -conclusion.

**Definition 2.7 (Aceto, 2001; Groote, 2007)** Let  $\mathcal{L} = (\Sigma, \Psi)$  be a de Simone language, and  $f$  be a function symbol in  $\Sigma$ . A rule of the form

$$\frac{x_i \xrightarrow{\tau} x'_i}{f(x_1, \dots, x_i, \dots, x_n) \xrightarrow{\tau} f(x_1, \dots, x'_i, \dots, x_n)}$$

with  $1 \leq i \leq n$  is called a patience rule of the  $i$ th argument of  $f$ .

In the following, a rule is called a plain rule if it is not a patience rule.

**Definition 2.8 (van Glabbeek, 2005)** Let  $\mathcal{L} = (\Sigma, \Psi)$  be a de Simone language, and  $f$  be a function symbol in  $\Sigma$ . 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  $f(t_1, \dots, t_n)$  with  $x$  occurring in  $t_i$ .

Then, the set of all arguments  $\text{Arg}$  of an operator can be divided into three classes: active arguments  $\text{Arg}_a$ , receiving arguments  $\text{Arg}_r$ , and others  $\text{Arg}_o$ , which is inspired by van Glabbeek (van Glabbeek, 2005). Therefore,  $\text{Arg} = \text{Arg}_a + \text{Arg}_r + \text{Arg}_o$ .

Similarly, patience rules of an operator can be divided into three classes. It should be noted that an argument may be both an active argument and a receiving argument, i.e.,  $\text{Arg}_a \cap \text{Arg}_r \neq \emptyset$ . However for clarity, from now on, if we say that an argument is a receiving argument, then it should not be an active argument, i.e., receiving arguments below are only those receiving arguments which are not active arguments simultaneously. Therefore,  $\text{Arg}_a$ ,  $\text{Arg}_r$  and  $\text{Arg}_o$  will be disjoint.

**Definition 2.9** Let  $\mathcal{L} = (\Sigma, \Psi)$  be a de Simone language, and  $f$  be a function symbol in  $\Sigma$ . A rule of the form  $\frac{H}{f(x_1, \dots, x_n) \xrightarrow{\tau} t}$  is called a rule with  $\tau$ -conclusion, if it is not a patience rule and there exists at least one positive  $\Sigma$  literal in  $H$ .

An notable example of the rules with  $\tau$ -conclusion, which will be used in Section 4.2, is the first transition rule of the hiding operator in CSP as follows.

$$p/A : \frac{p \xrightarrow{\alpha} p'}{p/A \xrightarrow{\tau} p'/A} \quad \alpha \in A \quad \frac{p \xrightarrow{\alpha} p'}{p/A \xrightarrow{\alpha} p'/A} \quad \alpha \notin A$$

Like the definition of a rule with  $\tau$ -conclusion, a transition rule is a rule with  $\tau$ -premise iff there exists a positive  $\Sigma$  literal like  $t \xrightarrow{\tau} t'$  in its premises. It is trivial that patience rules are rules with  $\tau$ -premise.

Before concluding this section, we will presume a small set of operators with default operational semantics:  $\text{nil}$ : means the successful termination.

$$\begin{aligned} a \cdot X &: a \cdot X \xrightarrow{a} X \\ X \boxplus Y &: \frac{X \xrightarrow{a} X'}{X \boxplus Y \xrightarrow{a} X'} \quad \frac{Y \xrightarrow{a} Y'}{X \boxplus Y \xrightarrow{a} Y'} \quad \frac{X \xrightarrow{\tau} X'}{X \boxplus Y \xrightarrow{\tau} X' \boxplus Y} \\ &\quad \frac{Y \xrightarrow{\tau} Y'}{X \boxplus Y \xrightarrow{\tau} X \boxplus Y'} \\ X \oplus Y &: X \oplus Y \xrightarrow{\tau} X \quad X \oplus Y \xrightarrow{\tau} Y \\ X \triangleright Y &: \frac{X \xrightarrow{a} X'}{X \triangleright Y \xrightarrow{a} X'} \quad \frac{X \xrightarrow{\tau} X'}{X \triangleright Y \xrightarrow{\tau} X' \triangleright Y} \quad X \triangleright Y \xrightarrow{\tau} Y \end{aligned}$$

where  $a \in \text{Act}$ . Operators  $\boxplus, \oplus, \triangleright$  are used to substitute the  $+$  operator and the prefixing with  $\tau$ , because many weak equivalences may not be preserved under the  $+$  operator of CCS. We call this language  $\mathbf{B}$  (Ulidowski, 2000).

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.

### 3 Weak Parametric Failure Equivalences

Before presenting the formal definitions of the weak parametric failure equivalences, two canonical equivalences, i.e., the weak failure equivalence and the weak impossible future equivalence, will be introduced. As we will see, they both are the special cases of the weak parametric failure equivalences.

**Definition 3.1**  $(\sigma, A) \in \text{Act}^* \times \mathcal{P}(\text{Act})$  is a weak failure pair of process  $p$  iff there exists some  $p'$  such that  $p \xRightarrow{\sigma} p' \wedge A \cap \mathcal{S}(p') = \emptyset$ , where  $\mathcal{S}(p') = \{a \in \text{Act} \mid p' \xrightarrow{a}\}$ . The set of all weak failure pairs of process  $p$  is called the weak failure of  $p$ , denoted by  $\mathcal{F}(p)$ .

**Weak Failure Equivalence**  $\sim_f$ : for any two processes  $p$  and  $q$ ,  $p \sim_f q$  iff  $\mathcal{F}(p) = \mathcal{F}(q)$ .

**Definition 3.2**  $(\sigma, \Phi) \in \text{Act}^* \times \mathcal{P}(\text{Act}^*)$  is a weak impossible future pair of process  $p$  iff there exists some  $p'$  such that  $p \xRightarrow{\sigma} p' \wedge \Phi \cap \mathcal{T}(p') = \emptyset$ , where  $\mathcal{T}(p') = \{\delta \in \text{Act}^* \mid p' \xrightarrow{\delta}\}$ . The set of all weak impossible future pairs

of process  $p$  is called the weak impossible future of  $p$ , denoted by  $I\mathcal{F}(p)$ .

**Weak Impossible Future Equivalence**  $\sim_{if}$ :  $p$  and  $q$  are two processes,  $p \sim_{if} q$  iff  $I\mathcal{F}(p) = I\mathcal{F}(q)$ .

As can be seen in the above definitions, the difference between the weak failure pair and the weak impossible future pair exists on their second parameters. The second parameter of the weak failure pair is a set of actions which cannot be enabled by  $p'$ . On the other hand, the second parameter of the weak impossible future pair is a set of action sequences which cannot be enabled by  $p'$ .

By the above observation, we put forward the definition on the weak parametric failure equivalences:

**Definition 3.3**  $(\sigma, \Phi) \in Act^* \times \mathcal{P}(Act^*)$  is a weak  $i$ -failure pair of process  $p$  iff there exists some  $p'$  such that  $p \xrightarrow{\sigma} p' \wedge \Phi \cap \mathcal{T}(p', i) = \emptyset$ , where  $\mathcal{T}(p', i) = \{\delta \in Act^* \mid p' \xrightarrow{\delta} \wedge |\delta| \leq i\}$ . The set of all weak  $i$ -failure pair of process  $p$  is called the weak  $i$ -failure of  $p$ , denoted by  $\mathcal{F}(p, i)$ .

**Weak Parametric Failure Equivalences**  $\sim_f^i$ : for any two processes  $p$  and  $q$ ,  $p \sim_f^i q$  iff  $\mathcal{F}(p, i) = \mathcal{F}(q, i)$ .

Also, we will often say that  $p$  and  $q$  are weak  $i$ -failure equivalent if  $p \sim_f^i q$ .

It is trivial that, in this framework, the weak failure equivalence is the weak 1-failure equivalence and the weak impossible future equivalence is the weak  $\omega$ -failure equivalence. In fact,  $\mathcal{S}(p') = \mathcal{T}(p', 1)$  and  $\mathcal{T}(p') = \mathcal{T}(p', \omega)$ .

The proposition below says that if  $p$  and  $q$  are weak  $j$ -failure equivalent with  $1 \leq j \leq \omega$ , then they are also weak  $i$ -failure equivalent for  $i < j$ .

**Proposition 3.4** Let  $1 \leq i < j \leq \omega$ ,  $p$  and  $q$  are two processes. If  $p \sim_f^j q$  then  $p \sim_f^i q$ .

**Proof** By the definition of weak parametric failure equivalences,  $p \sim_f^j q$  iff  $\mathcal{F}(p, j) = \mathcal{F}(q, j)$ . Then,  $\mathcal{F}(p, i) = \mathcal{F}(q, i)$  can be obtained from Definition 3.3,  $\mathcal{F}(p, j) = \mathcal{F}(q, j)$  and  $1 \leq i < j \leq \omega$ . Therefore,  $p \sim_f^i q$ .  $\square$

Before concluding this section, an alternative characterization of the weak parametric failure equivalences are to be presented. This alternative characterization will be useful in obtaining the rule formats.

**Proposition 3.5** Let  $p, q$  be two processes. For  $1 \leq i \leq \omega$ ,  $p \sim_f^i q$  iff

1) for any  $\sigma \in \mathcal{T}(p, \omega)$  and  $p'$  with  $p \xrightarrow{\sigma} p'$ , there exists  $q'$  such that  $q \xrightarrow{\sigma} q'$  and  $\mathcal{T}(q', i) \subseteq \mathcal{T}(p', i)$ , and

2) for any  $\sigma \in \mathcal{T}(q, \omega)$  and  $q'$  with  $q \xrightarrow{\sigma} q'$ , there exists  $p'$  such that  $p \xrightarrow{\sigma} p'$  and  $\mathcal{T}(p', i) \subseteq \mathcal{T}(q', i)$ .

**Proof** ( $\Leftarrow$ ) It is enough to prove that  $\mathcal{F}(p, i) = \mathcal{F}(q, i)$ . If it is not true, then, without loss of generality, suppose that there exists some  $(\sigma, \Phi) \in (Act^* \times \mathcal{P}(Act^*))$  such that  $(\sigma, \Phi) \in \mathcal{F}(p, i)$  but  $(\sigma, \Phi) \notin \mathcal{F}(q, i)$ .

By  $(\sigma, \Phi) \in \mathcal{F}(p, i)$  and the definition of weak  $i$ -failure pair in Definition 3.3, there must exist some  $p'$  such that  $p \xrightarrow{\sigma} p' \wedge \Phi \cap \mathcal{T}(p', i) = \emptyset$ .

By  $p \xrightarrow{\sigma} p'$  and the hypothesis, there exists  $q'$  such that  $q \xrightarrow{\sigma} q'$  and  $\mathcal{T}(q', i) \subseteq \mathcal{T}(p', i)$ . Then, from  $\Phi \cap \mathcal{T}(p', i) = \emptyset$ , we have  $\Phi \cap \mathcal{T}(q', i) = \emptyset$ .

Therefore, there exists  $q'$  such that  $q \xrightarrow{\sigma} q'$  and  $\Phi \cap \mathcal{T}(q', i) = \emptyset$ , which contradicts with  $(\sigma, \Phi) \notin \mathcal{F}(q, i)$ .

( $\Rightarrow$ ) By the symmetry, we need only prove the first point. Suppose that  $\sigma$  is any trace in  $\mathcal{T}(p, \omega)$  and  $p'$  is a process such that  $p \xrightarrow{\sigma} p'$ . Let  $\Phi = I_{all}^i - \mathcal{T}(p', i)$  with  $I_{all}^i$  is the set of all action sequences whose lengths are not exceed number  $i$ . Then, we have  $(\sigma, \Phi) \in \mathcal{F}(p, i)$ .

By Definition 3.3,  $(\sigma, \Phi) \in \mathcal{F}(q, i)$ . Hence, there exists some  $q'$  such that  $q \xrightarrow{\sigma} q'$  and  $\Phi \cap \mathcal{T}(q', i) = \emptyset$ . By

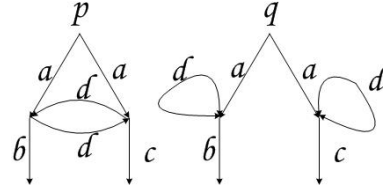


Figure 1:  $p$  and  $q$  are weak 1-failure equivalent, but  $p/\{d\}$  and  $q/\{d\}$  are not weak 1-failure equivalent.

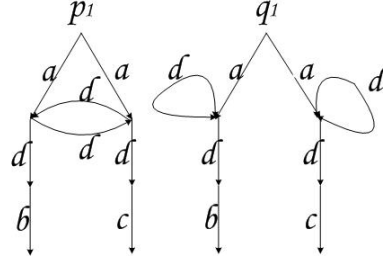


Figure 2:  $p_1$  and  $q_1$  are weak 2-failure equivalent, but  $p_1/\{d\}$  and  $q_1/\{d\}$  are not weak 2-failure equivalent.

$\Phi = I_{all}^i - \mathcal{T}(p', i)$ , we have  $(I_{all}^i - \mathcal{T}(p', i)) \cap \mathcal{T}(q', i) = \emptyset$ . Therefore,  $\mathcal{T}(q', i) \subseteq \mathcal{T}(p', i)$ .  $\square$

## 4 Intuitive Motivations on Rule Formats

This section gives several representative examples to show some intuitive motivations on the rule formats of the weak parametric failure equivalences. However, we do not want to discuss them from the scratch, only the two critical points sketched in the introduction are to be mentioned: the first subsection is to observe the feasibility of adding rules with  $\tau$ -conclusion; the second subsection is to inspect the necessity of the patience rules for receiving arguments.

It should be noted that, in this section, we mainly concern 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 in section 2.

### 4.1 On Rules with $\tau$ -conclusion

Let's see an example in Figure 1 and Figure 2. Firstly, the two graphs in Figure 1, i.e.,  $p$  and  $q$ , are weak 1-failure equivalent. However, after hiding  $d$  actions,  $p/\{d\}$  is not weak 1-failure equivalent to  $q/\{d\}$ , which can be seen from the weak 1-failure pair  $(a, \{c\}) \in \mathcal{F}(q/\{d\}, 1)$  but  $(a, \{c\}) \notin \mathcal{F}(p/\{d\}, 1)$ . If we take the weak 2-failure equivalence into consideration, we may find that  $p$  and  $q$  are not yet weak 2-failure equivalent, because  $(a, \{b, db\}) \in \mathcal{F}(q, 1)$  but  $(a, \{b, db\}) \notin \mathcal{F}(p, 1)$ .

As for the weak 2-failure equivalence,  $p_1$  and  $q_1$ , the two graphs in Figure 2, are weak 2-failure equivalent. However, this equivalence also cannot be preserved after hiding  $d$  actions. In fact, for any weak  $i$ -failure equivalence with  $i < \omega$ , a similar counterexample exists. On the contrary, the weak  $\omega$ -failure equivalence can be preserved under the hiding operator.

Generalizing to any rules with  $\tau$ -conclusion, a common characterization of these rules is that they all consume the observable actions of the subprocesses and produce  $\tau$  transitions at the same time. Therefore, we conjecture that any weak  $i$ -failure equivalence with  $i < \omega$  will probably be broken under the rules with  $\tau$ -conclusion, but the weak  $\omega$ -failure equivalence will be preserved.

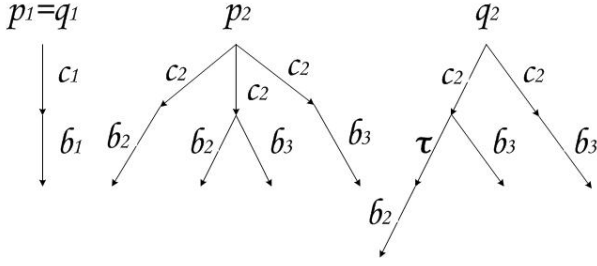


Figure 3:  $p_1$  and  $q_1$ ,  $p_2$  and  $q_2$  are weak  $i$ -failure equivalent for  $i \in \mathbb{N} \cup \{\omega\}$ .

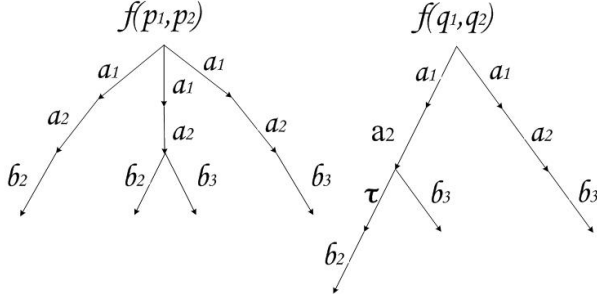


Figure 4:  $f(p_1, p_2)$  and  $f(q_1, q_2)$  are weak 1-failure equivalent but not weak 2-failure equivalent.

#### 4.2 Patience Rules for Receiving Arguments

Similarly, see an example in Figure 3, Figure 4 and Figure 5. Consider adding the rules  $r_1 = \frac{x_1 \xrightarrow{c_1} x'_1, x_2 \xrightarrow{c_2} x'_2}{f(x_1, x_2) \xrightarrow{a_1} g(x'_1, x'_2)}$ ,  $r_2 = \frac{x_1 \xrightarrow{b_1} x'_1}{g(x_1, x_2) \xrightarrow{a_2} h(x_2)}$ ,  $r_3 = \frac{x_1 \xrightarrow{b_3} x'_1}{h(x_1) \xrightarrow{b_3} nil}$ ,  $r_4 = \frac{x_1 \xrightarrow{b_2} x'_1}{h(x_1) \xrightarrow{b_2} nil}$  and their associated patience rules for active arguments into language **B**. Note that  $g(x_1, x_2)$  has its second argument as a receiving argument.

Let  $p_1, p_2, q_1, q_2$  be the processes shown in Figure 3. It can be easily verified that  $p_1 \sim_f^1 q_1$  and  $p_2 \sim_f^1 q_2$ . According to the above rules, we have  $f(p_1, p_2)$  and  $f(q_1, q_2)$  shown in the two graphs of Figure 4. Now,  $f(p_1, p_2)$  and  $f(q_1, q_2)$  are also weak 1-failure equivalent, i.e.,  $f(p_1, p_2) \sim_f^1 f(q_1, q_2)$ . Therefore, It seems that patience rules for receiving arguments are not prerequisite for the weak 1-failure equivalence.

When it turns to the weak 2-failure equivalence, we also have  $p_1 \sim_f^2 q_1$  and  $p_2 \sim_f^2 q_2$ . However,  $f(p_1, p_2)$  and  $f(q_1, q_2)$  are not weak 2-failure equivalent yet, because  $(a_1, \{a_2, b_3\})$  is a weak 2-failure pair of  $f(p_1, p_2)$  but not a weak 2-failure pair of  $f(q_1, q_2)$ . Hence, the weak 2-failure equivalence is not preserved under the above rules. However, if we further add patience rule for the second argument of  $g(x_1, x_2)$  into language **B**, then  $f(p_1, p_2)$  and  $f(q_1, q_2)$  is the two graphs in Figure 5. Now, it can be easily verified that  $f(p_1, p_2) \sim_f^2 f(q_1, q_2)$ . Therefore, adding the patience rules for receiving arguments may preserve the weak 2-failure equivalence.

In fact, we will assert, in the next section, that the patience rules for receiving arguments are prerequisite for the weak  $i$ -failure equivalence with  $i > 1$ , but, they are not necessary for the weak 1-failure equivalence.

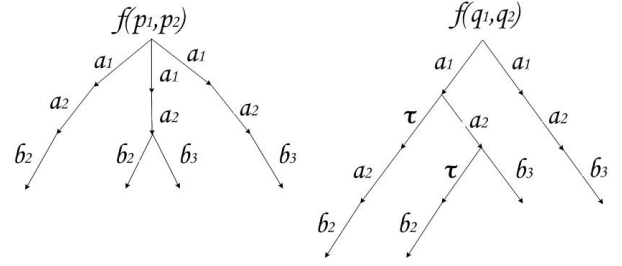


Figure 5:  $f(p_1, q_1)$  and  $f'(p_2, q_2)$  are weak 2-failure equivalent.

## 5 Rule Formats for Weak Parametric failure Equivalence

After the intuitive observations in the preceding section, we will, in this section, present formally the rule formats for the weak parametric failure equivalences. In fact, as stated in the introduction, we will present three different rule formats: weak 1-failure format for the weak 1-failure equivalence, weak finite failure format for the weak  $i$ -failure equivalence with  $1 < i < \omega$  and weak  $\omega$ -failure format for the weak  $\omega$ -failure equivalence.

### 5.1 Formal Definitions on the Rule Formats

The de Simone language is employed as our starting point in retrieving the rule formats for the weak parametric failure equivalences.

**Definition 5.1.1** A de Simone language  $\mathcal{L}$  is in weak 1-failure format if

- 1) patience rules are the only rules with  $\tau$ -premises,
- 2) patience rules for active arguments are prerequisite,
- 3) rules with  $\tau$ -conclusion are not permitted.

Following it, the weak finite failure format is:

**Definition 5.1.2** A de Simone language  $\mathcal{L}$  is in weak finite failure format if

- 1) patience rules are the only rules with  $\tau$ -premises,
- 2) patience rules for active arguments and receiving arguments are all prerequisite,
- 3) rules with  $\tau$ -conclusion are not permitted.

Then, the weak  $\omega$ -failure format for the weak  $\omega$ -failure equivalence is to be presented.

**Definition 5.1.3** A de Simone language  $\mathcal{L}$  is in weak  $\omega$ -failure format if

- 1) patience rules are the only rules with  $\tau$ -premises, and
- 2) patience rules for active arguments and receiving arguments are all prerequisite.

It should be pointed out that the exclusion of rules with  $\tau$ -conclusion and the allowance of dropping the patience rules for receiving arguments are not two separated restrictions. In fact, to obtain the effect of the allowance of dropping the patience rules for receiving arguments in the weak 1-failure format, the exclusion of rules with  $\tau$ -conclusion is a precondition. We will prove this conclusion in Lemma 5.3.5.

Below, we will study the relations between the above three rule formats. To fulfil this purpose, we need to define the 'tighter than' relation between rule formats.

**Definition 5.1.4** Let  $A$  and  $B$  be two rule formats.  $A$  is tighter than  $B$  iff, for any languages  $\mathcal{L} = (\Sigma, \Psi)$  in format  $A$ , all transition rules in  $\Psi$  are also in format  $B$ . Moreover,  $A$  is strictly tighter than  $B$  iff  $A$  is tighter than  $B$  and there exists some languages  $\mathcal{L} = (\Sigma, \Psi)$  in format  $B$  such that at least one of the transition rules in  $\Psi$  are not in format  $A$ .

**Theorem 5.1.5** The weak 1-failure format is strictly tighter than the weak finite failure format and the weak finite failure format is strictly tighter than the weak  $\omega$ -failure format.

**Proof** Comparing Definition 5.1.1 and Definition 5.1.2, patience rules for receiving arguments are not yet necessary for weak 1-failure format. Therefore, for any languages  $\mathcal{L}$  in weak 1-failure format, its transition rules will also be in weak finite failure format. Likewise, Definition 5.1.2 and Definition 5.1.3 are only different on the rules with  $\tau$ -conclusion. Therefore, after refusing all rules with  $\tau$ -conclusion, any languages  $\mathcal{L}$  in weak finite failure format will have its transition rules in the weak  $\omega$ -failure format.

The strictness between weak 1-failure format and weak finite failure format can be witnessed by the languages in Section 4.2. Introducing the patience rules for receiving arguments, it is a weak finite failure language. However, patience rules for receiving arguments are not in weak 1-failure format. The strictness between weak finite failure format and weak  $\omega$ -failure format can be witnessed by introducing the hiding operator of CSP into any weak  $\omega$ -failure language. The obtained languages are still weak  $\omega$ -failure languages. However, one of transition rules of hiding operator is not in weak finite failure format.  $\square$

## 5.2 Ruloids And Ruloid Theorems On The Two Formats

Ruloids and the ruloid theorem originated from the works of Bloom (Bloom, 1995, 1990) for the GSOS format. In this section, we will introduce the ruloids and the ruloid theorem for the weak  $\omega$ -failure format. And the ruloids and the ruloid theorem for the other two formats can be retrieved in the same way. The ruloid theorems will be useful for the proving the congruence theorems in the next three subsections.

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

$$\frac{\{x_i \xrightarrow{\alpha_i} x'_i\}_{i \in I}}{C(x_1, \dots, x_n) \xrightarrow{\alpha} D(y_1, \dots, y_n)} \quad (1)$$

such that  $y_i = x'_i$  for  $i \in I$  and  $y_i = x_i$  for  $i \notin I$ , where  $I \subseteq \{1, 2, \dots, n\}$ . These expressions characterize all possible behaviors of the context  $C$  in the language.

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 weak  $\omega$ -failure 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 (1), we still write it as  $D(y_1, \dots, y_n)$ .

Furthermore, two properties are needed to be imposed on  $\mathcal{R}(C, \alpha)$ , we call them soundness property and completeness property, by a little abusing the terminologies.

**Definition 5.2.1** Let  $\mathcal{L} = (\Sigma, \Psi)$  be a language in the weak  $\omega$ -failure format, and  $C(x_1, \dots, x_n)$  be any context of  $n$  holes in  $\mathcal{L}$ . A set  $\mathcal{R}(C, \alpha)$  of ruloids of form (1) are ruloids of context  $C$  and action  $\alpha$ , with  $\alpha \in Act \cup \{\tau\}$ , iff

1) Soundness. Let  $r \in \mathcal{R}(C, \alpha)$  be a ruloid of form (1). If  $\zeta$  is a closed  $\Sigma$  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  $\zeta(C(x_1, \dots, x_n)) \xrightarrow{\alpha} \zeta(D(y_1, \dots, y_n))$ .

2) Completeness. Let  $\zeta$  be any closed  $\Sigma$  substitution. If  $\zeta(C(x_1, \dots, x_n)) \xrightarrow{\alpha}$ , then there must exist a ruloid  $r$  of form (1) in ruloids  $\mathcal{R}(C, \alpha)$ , 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  $\alpha$ , and then prove that the obtained ruloids satisfy the above two properties, which form the ruloid theorem.

**Strategy 5.2.2** Let  $\mathcal{L} = (\Sigma, \Psi)$  be a language in the weak  $\omega$ -failure format.  $C(x_1, \dots, x_n)$  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  $\mathcal{R}(C, \alpha) = \{x \xrightarrow{\alpha} x'\};$   
 $x \xrightarrow{\alpha} x'$   
 2) If  $C = f(x_1, \dots, x_n)$  with  $f \in \Sigma$  and  $ar(f) = n$ , then  $\mathcal{R}(C, \alpha) = (f, \alpha)$ , where  $(f, \alpha)$  denotes the set of all rules in  $\Psi$  whose source is  $f(x_1, \dots, x_n)$  and output is  $\alpha$ .

3) If  $C$  is any context. We can rewrite  $C(x_1, \dots, x_n)$  as  $f(C_1[X_1], \dots, C_m[X_m])$ , 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  $X_i = x_{i1}x_{i2}\dots x_{im_i}$  for  $C_i$  is a context of  $m_i$  holes. Now, let  $r$  be any ruloid of form (1) in  $(f, \alpha)$  and  $\mathcal{R}(C_i, \alpha_i)$  be ruloids of context  $C_i$  and action  $\alpha_i$  retrieved by induction on this strategy. Then, any ruloids in  $\mathcal{R}(C, \alpha)$  can be obtained by the following steps:

i) pick out randomly from  $\mathcal{R}(C_i, \alpha_i)$  a rule  $r_i$ , for all  $i \in I$ ;

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

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

4)  $\mathcal{R}(C, \alpha)$  is the set of all possible ruloids that can be retrieved from step 3).  $\square$

**Theorem 5.2.3** Let  $\mathcal{L} = (\Sigma, \Psi)$  be a language in the weak  $\omega$ -failure format, and  $C(x_1, \dots, x_n)$  be any context of  $n$  holes in  $\mathcal{L}$ . The set of ruloids  $\mathcal{R}(C, \alpha)$  obtained from the Strategy 5.2.2 are ruloids of context  $C$  and action  $\alpha$  with  $\alpha \in Act \cup \{\tau\}$ .

**Proof** First, the obtained ruloids  $\mathcal{R}(C, \alpha)$  of context  $C$  and action  $\alpha$  are all in form (1), which can be easily retrieved from the construction procedure in the Strategy 5.2.2.

Second, the obtained ruloids  $\mathcal{R}(C, \alpha)$  of context  $C$  and action  $\alpha$  satisfy the soundness property. Let  $r \in \mathcal{R}(C, \alpha)$  be a ruloid of form (1), where  $C$  is a context of  $n$  holes and  $\alpha \in Act \cup \{\tau\}$  is an action.  $\zeta$  is a closed  $\Sigma$  substitution such that  $\zeta(x_i) \xrightarrow{\alpha_i} \zeta(x'_i)$  for all  $i \in I$ . Make an induction on the context  $C$ .

i) if  $C \in V$ , then, without loss of generality, suppose  $C = x$ . The soundness property is trivial from  $\mathcal{R}(C, \alpha) =$

$$\{x \xrightarrow{\alpha} x'\};$$

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

iii) if  $C$  is any context of  $n$  holes, then, from the strategy, there exist contexts  $C_1[X_1], \dots, C_m[X_m]$  such that, after substitution,  $ante(r_i)$  is a part of the premise of  $r$  for  $1 \leq i \leq m$ , where  $r_i \in \mathcal{R}(C_i, \alpha_i)$ . Now, by the hypothesis,  $\zeta$  is a closed  $\Sigma$  substitution making all premises of  $r$  enable. Hence,  $cons(r_i)$  is enabled, which means that  $C_i[X_i] \xrightarrow{\alpha_i} D_i[Y_i]$  for all  $1 \leq i \leq m$ . Furthermore,  $C$  is rewritten as  $f(C_1[X_1], \dots, C_m[X_m])$ . Therefore, the transition rules in  $(f, \alpha)$  guarantee the enablement of  $C(x_1, \dots, x_n) \xrightarrow{\alpha}$ .

Third, the obtained ruloids  $\mathcal{R}(C, \alpha)$  of context  $C$  and action  $\alpha$  satisfy the completeness property, which can also be easily retrieved from the construction procedure of the Strategy 5.2.2.  $\square$

As we can see that, for a ruloid of form (1), 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  $i \in \{1, \dots, n\} \setminus I$ , into the premises, as in the form (2). And form (1) and form (2) are equivalent when any closed  $\Sigma$  substitution  $\zeta$  is applied on them. In this case,  $\zeta(x_i) \xrightarrow{\epsilon} \zeta(x'_i)$  denotes that subprocess  $\zeta(x_i)$  executes no transition.

$$\frac{\{x_i \xrightarrow{\alpha_i} x'_i\}_{i \in I} \{x_i \xrightarrow{\epsilon} x'_i\}_{i \in \{1, \dots, n\} \setminus I}}{C(x_1, \dots, x_n) \xrightarrow{\alpha} D(y_1, \dots, y_n)} \quad (2)$$

Like the definitions on the transition rules, we can also define the patience ruloids and ruloids with  $\tau$ -conclusion.

**Theorem 5.2.3** Let  $\mathcal{L} = (\Sigma, \Psi)$  be a language in the weak  $\omega$ -failure format, and  $C(x_1, \dots, x_n)$  be any context of  $n$  holes in  $\mathcal{L}$ . The set of ruloids  $\mathcal{R}(C, \alpha)$  obtained from the Strategy 5.2.2 is the ruloids of context  $C$  and action  $\alpha$  satisfies  $\alpha \in Act \cup \{\tau\}$ .

**Proof** Firstly, the obtained ruloids  $\mathcal{R}(C, \alpha)$  of context  $C$  and action  $\alpha$  are all in form (1), which can be easily retrieved from the construction procedure in the Strategy 5.2.2.

Secondly, the obtained ruloids  $\mathcal{R}(C, \alpha)$  of context  $C$  and action  $\alpha$  satisfy the soundness property. Let  $r \in \mathcal{R}(C, \alpha)$  be a ruloid of form (1), where  $C$  is a context of  $n$  holes and  $\alpha \in Act \cup \{\tau\}$  is an action.  $\zeta$  is a closed  $\Sigma$  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 trivial from  $\mathcal{R}(C, \alpha) =$

$$\left\{ \begin{array}{l} x \xrightarrow{\alpha} x' \\ x \xrightarrow{\alpha} x' \end{array} \right\};$$

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

iii) if  $C$  is any context of  $n$  holes, then by Strategy 5.2.2,  $C(x_1, \dots, x_n)$  can be rewritten as  $f(C_1(X_1), \dots, C_m(X_m))$  for some operator  $f \in \Sigma$  and  $ar(f) = m$ , and  $ante(r)$  consist of  $anti(r_1), \dots, ante(r_m)$ , where  $r_i \in \mathcal{R}(C_i, \alpha_i)$  for all  $1 \leq i \leq m$ . By the assumption of the soundness property that  $ante(r)$  is enabled in closed  $\Sigma$  substitution  $\zeta$ . Therefore, by the induction hypothesis,  $cons(r_1), \dots, cons(r_m)$  are all enabled in  $\zeta$ . 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,  $cons(r_1), \dots, cons(r_m)$  constitute  $ante(f)$ . Still by the induction hypothesis on operator  $f$ , the transition rules in  $(f, \alpha)$  guarantee the enableness of  $C(x_1, \dots, x_n) \xrightarrow{\alpha}$ .

Last, the obtained ruloids  $\mathcal{R}(C, \alpha)$  of context  $C$  and action  $\alpha$  satisfy the completeness property, which can also be easily retrieved from the construction procedure of the Strategy 5.2.2.  $\square$

As we can see that, for a ruloid of form (1), 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  $i \in \{1, \dots, n\} \setminus I$ , into the premises, as in the form (2). In this case,  $\zeta(x_i) \xrightarrow{\epsilon} \zeta(x'_i)$  denotes that subprocess  $\zeta(x_i)$  executes no transition.

$$\frac{\{x_i \xrightarrow{\alpha_i} x'_i\}_{i \in I} \{x_i \xrightarrow{\epsilon} x'_i\}_{i \in \{1, \dots, n\} \setminus I}}{C(x_1, \dots, x_n) \xrightarrow{\alpha} D(y_1, \dots, y_n)} \quad (3)$$

The  $\epsilon$  transition will not be added to the TSS. In fact, a TSS is a pair  $(\Sigma, \Psi)$ , where  $\Sigma$  is a set of function symbols and  $\Psi$  is a set of transition rules assigned to the function symbols. Therefore, even no ruloids are in the TSS.

The introducing of  $\epsilon$  transition is to substitute the ruloids of form (1) with the ruloids of form (2), since these two forms are equivalent when any closed  $\Sigma$  substitution  $\zeta$  is applied. In fact, we want to express a viewpoint that, for any ruloid  $r$ , it should have two different but equivalent forms, i.e., form (1) and form (2).

For the equivalence between form (1) and form (2), we want to take an example to show it. Let  $x_j \xrightarrow{\epsilon} x'_j$  be any  $\epsilon$ -premise in some ruloid  $r$ . In fact, it denotes that, when ruloid  $r$  is applied in some  $\Sigma$  substitution  $\zeta$ , subprocess  $\zeta(x_j)$  is not fired at all. Also, if the form (1) of  $r$  is applied, the same results are retrieved.

The introducing of  $\epsilon$  transitions and thus form (2) will make Lemma 5.3.1 and its proof prone to be comprehended. In Lemma 5.3.1, we will see that, in the weak  $\omega$ -failure languages, when process  $C(p_1, \dots, p_n)$  evolves into  $C'(p_1, \dots, p'_n)$  by applying a ruloid and produce a transition (observable action or  $\tau$  transition), each subprocess  $p_i$  will

also evolve into  $p'_i$  and produce a transition (observable action,  $\tau$  transition or  $\epsilon$  transition).

Based on the ruloids and the ruloid theorem, we may restate several classes of rules, which have been defined previously, with the notion of ruloids. And, they will be more intuitive and prone to be used in the following.

The first class of rules which we concern is the patience rules. As their counterparts, the definition of patience ruloids is as follows.

**Definition 5.2.4** A ruloid of the form 
$$\frac{x_i \xrightarrow{\tau} x'_i}{C(x_1, \dots, x_i, \dots, x_n) \xrightarrow{\tau} C(x_1, \dots, x'_i, \dots, x_n)}$$
 with  $1 \leq i \leq n$  is called a patience ruloid of the  $i$ th argument of the context  $C$ .

In the following, a ruloid is called a plain ruloid if it is not a patience ruloid. Similar to the division in the patience rules, we also need to 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 5.2.2 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 only adding the patience rules for active arguments into the language, then, after using Strategy 5.2.2, the patience ruloids in  $\mathcal{R}(C, \tau)$  are patience ruloids for active arguments.

2) If further adding the patience rules for receiving arguments into the language, then, after using Strategy 5.2.2, the patience ruloids in  $\mathcal{R}(C, \tau)$  are patience ruloids for active arguments and receiving arguments. Therefore, getting rid of the patience ruloids for active arguments, we can obtain the patience ruloids for receiving arguments.

This division is obtained indirectly from Strategy 5.2.2 and patience rules, and thus it is hard to be used in the following. Here, we will propose another division which is directly based on the arguments of a context.

**Definition 5.2.5** Let  $\mathcal{L} = (\Sigma, \Psi)$  be a weak  $\omega$ -failure 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 (1) in  $\mathcal{R}(C, \tau)$  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 exist another context  $D$  and a plain ruloid  $r$  of form (1) in  $\mathcal{R}(D)$  such that  $C(x'_1, \dots, x'_n)$  appears as the target of  $r$  and  $x'_i$  appears as right-hand side of a premise.

Below, we will prove that these two divisions are indeed equivalent, i.e., a patience ruloid of some context  $C$  is a patience ruloid for active (resp. receiving, other) argument obtained from Strategy 5.2.2 and patience rules iff it is a patience ruloid for active (resp. receiving, other) argument defined by Definition 5.2.5.

**Proposition 5.2.6** The division defined by Definition 5.2.5 is equivalent to the division obtained from Strategy 5.2.2 and patience rules.

**Proof** ( $\Leftarrow$ ) Let  $\mathcal{L} = (\Sigma, \Psi)$  be a de Simone language, and  $C$  be any context of  $n$  holes.

If only adding the patience rules for active arguments into the language, we need to prove that each active argument of the context  $C$  defined by Definition 5.2.5 has a patience ruloid. We will prove by making an induction on the context  $C$  and Strategy 5.2.2.

1) If  $C \in V$  or  $C \in \Sigma$ , then it can be easily obtained from Strategy 5.2.2 and Definition 2.8.

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

3) We need to prove that each active argument of  $C$  defined by Definition 5.2.5 has a patience ruloid. Suppose that the  $i$ th argument of  $C$  is an active argument. Then, by Definition 5.2.5, there exists a plain ruloid  $r$  of form (1) in  $\mathcal{R}(C, \tau)$  such that  $x_i$  appears as left-hand side of a premise. By Strategy 5.2.2,  $x_i$  must appear as left-hand

side of a premise of some context. Without loss of generality, assume that  $x_i$  is the  $k$ th argument of the  $C_j$ . By the induction hypothesis, the  $k$ th argument of  $C_j$  is active and thus has a patience ruloid. Also by Strategy 5.2.2, the  $j$ th argument of functor  $f$  is active and thus has a patience ruloid. Therefore, we have that the  $i$ th argument of  $C$  has a patience ruloid by Strategy 5.2.2 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 5.2.5 has a patience ruloid. Assume that the  $i$ th argument of context  $C$  is receiving. Then, by Definition 5.2.5, there exist another context  $D$  and a plain ruloid  $r$  of form (1) in  $\mathcal{R}(D)$  such that  $C(x'_1, \dots, x'_n)$  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 5.2.2.

1) If  $C \in V$  or  $C \in \Sigma$ , then, by Definition 2.8, the  $i$ th argument of  $C$  is receiving. Therefore, it should have a patience rule by the hypothesis. By Strategy 5.2.2, each patience rule is also a patience ruloid.

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

3) We need to prove that the  $i$ th argument of  $C$  defined by Definition 5.2.5 has a patience ruloid. By Strategy 5.2.2,  $x_i$  must appear as right-hand side of a premise of some context. Without loss of generality, assume that  $x_i$  is the  $k$ th argument of the  $C_j$ . By the induction hypothesis, the  $k$ th argument of  $C_j$  is receiving or active and thus has a patience ruloid. Also by Strategy 5.2.2, the  $j$ th argument of functor  $f$  is receiving or active and thus has a patience ruloid. Therefore, we have that the  $i$ th argument of  $C$  has a patience ruloid by Strategy 5.2.2 and the above two patience ruloids for  $C_j$  and  $f$ , respectively.

( $\Rightarrow$ ) It is trivially true since, according to Strategy 5.2.2, 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 5.2.5, and then using Strategy 5.2.2 to obtain the division from Strategy 5.2.2 and patience rules.  $\square$

The second class of rules is the rules with  $\tau$  conclusion. Likewise, we may define the ruloids with  $\tau$  conclusion as their counterparts.

**Definition 5.2.7** A ruloid of the form  $\frac{H}{C(x_1, \dots, x_n) \xrightarrow{\tau} D(y_1, \dots, y_n)}$  is called a ruloid with  $\tau$ -conclusion, if it is not a patience ruloid and there exists at least one positive  $\Sigma$  literal in  $H$ .

Also, we want to show that the exclusion of rules with  $\tau$ -conclusion is equivalent to the exclusion of ruloids with  $\tau$ -conclusion.

**Proposition 5.2.8** Let  $\mathcal{L}$  be a weak  $\omega$ -failure language, and  $C(x_1, \dots, x_n)$  be any context of  $n$  holes. If no rule with  $\tau$ -conclusion is allowed, then no ruloid with  $\tau$ -conclusion can be in  $\mathcal{R}(C, \tau)$ , and vice versa.

**Proof** This can be easily obtained from a fact that, by Strategy 5.2.2, the output of any ruloid is in fact the output of some rule.  $\square$

### 5.3 Weak 1-Failure Format for Weak 1-Failure Equivalence

In this subsection, several necessary lemmas are to be presented and the usage of them to prove the congruence theorems in the following three subsections has been listed in Table 1. The symbol  $\checkmark$  in the table denotes that some lemma is to be used in the proof of the congruence theorem for the corresponding format. For example, the congruence theorem for the weak  $\omega$ -failure format needs the first three lemmas, i.e., Lemma 5.3.1, Lemma 5.3.2 and Lemma 5.3.3.

The following lemma states that, in the weak  $\omega$ -failure languages, any weak trace of a composite process may be

Table 1: The Usage of Lemmas in Section 5.3 in Proving the Congruence Theorems

format \ Lemma	5.3.1	5.3.2	5.3.3	5.3.4	5.3.5
1-failure format	$\checkmark$	$\checkmark$		$\checkmark$	$\checkmark$
finite failure format	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	
$\omega$ -failure format	$\checkmark$	$\checkmark$	$\checkmark$		

decomposed into weak traces of its subprocesses. Besides, this lemma also holds in weak finite failure languages and weak 1-failure languages.

**Lemma 5.3.1** Let  $\mathcal{L} = (\Sigma, \Psi)$  be a weak  $\omega$ -failure language, and  $C(x_1, \dots, x_n)$  be any context of  $n$  holes. Suppose that  $\zeta$  is any closed  $\Sigma$  substitution mapping  $x_i$  into  $p_i$ . If  $\sigma$  is a trace in  $\mathcal{T}(C(p_1, \dots, p_n), \omega)$ , then, for all  $1 \leq i \leq n$ , there is a trace  $\sigma_i$  in  $\mathcal{T}(p_i, \omega)$  such that, when  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$ , we have  $p_i \xrightarrow{\sigma_i} p'_i$ .

**Proof** Since  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$ , we have  $C(p_1, \dots, p_n) = C_0(p_{10}, \dots, p_{n0}) \xrightarrow{\alpha_1} C_1(p_{11}, \dots, p_{n1}) \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_m} C_m(p_{1m}, \dots, p_{nm}) = C'(p'_1, \dots, p'_n)$ , where  $\forall 1 \leq j \leq m : \alpha_j \in Act \cup \{\tau\}$  and  $\sigma' = \alpha_1 \dots \alpha_m$  is equivalent to  $\sigma$  if all its  $\tau$  transitions are omitted.

We will prove this lemma by making an induction on the length of  $\sigma'$ .

1)  $|\sigma'| = 1$ . Let  $\sigma' = \alpha$ . By the completeness property of the ruloids, there should be a ruloid of form (1) in  $\mathcal{R}(C, \alpha)$ , and  $p_i \xrightarrow{\alpha} p'_i$  for all  $i \in I$ . As is shown before that, we have a ruloid of form (2) corresponding with form (1). Therefore, there exist  $p_i \xrightarrow{\alpha_i} p'_i$  for all  $i \in I$  and  $p_i \xrightarrow{\epsilon} p'_i$  for all  $i \in \{1, \dots, n\} - I$ , i.e., when  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$ , we have  $p_i \xrightarrow{\alpha_i} p'_i$  for all  $i \in I$  and  $p_i \xrightarrow{\tau} p'_i$  for all  $i \in \{1, \dots, n\} - I$ .

2) Assume that, when  $|\sigma'| = m - 1$  with  $m \geq 1$ , if  $\sigma$  is a trace in  $\mathcal{T}(C(p_1, \dots, p_n), \omega)$  then, for all  $1 \leq i \leq n$ , there should be a trace  $\sigma_i$  in  $\mathcal{T}(p_i, \omega)$  such that, when  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$ , we have  $p_i \xrightarrow{\sigma_i} p'_i$ .

3) For  $|\sigma'| = m$ , suppose that  $C(p_1, \dots, p_n) = C_0(p_{10}, \dots, p_{n0}) \xrightarrow{\alpha_1} C_1(p_{11}, \dots, p_{n1}) \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_m} C_m(p_{1m}, \dots, p_{nm}) = C'(p'_1, \dots, p'_n)$ . By the induction hypothesis, when  $C(p_1, \dots, p_n) = C_0(p_{10}, \dots, p_{n0}) \xrightarrow{\alpha_1} C_1(p_{11}, \dots, p_{n1}) \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_{m-1}} C_{m-1}(p_{1(m-1)}, \dots, p_{n(m-1)}) = C''(p''_1, \dots, p''_n)$ , we have  $p_i \xrightarrow{\sigma'_i} p''_i$ . Now, when  $C_{m-1}(p_{1(m-1)}, \dots, p_{n(m-1)}) = C''(p''_1, \dots, p''_n) \xrightarrow{\alpha_m} C_m(p_{1m}, \dots, p_{nm}) = C'(p'_1, \dots, p'_n)$ , we have  $p''_i \xrightarrow{\alpha'_m} p'_i$ . Therefore, when  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$ , we have  $p_i \xrightarrow{\sigma'_i \alpha'_m} p'_i$ .  $\square$

The following lemma states that, in the weak  $\omega$ -failure languages, the weak trace equivalence will be preserved and the composite processes can reach the same contexts after same weak traces. The definition of weak trace equivalence is that: two processes  $p$  and  $q$  are weak trace equivalent, denoted as  $p \sim_t q$ , iff they have the same set of weak traces, i.e.,  $p \sim_t q$  iff  $\mathcal{T}(p, \omega) = \mathcal{T}(q, \omega)$ . This lemma also holds in weak finite failure languages and weak 1-failure languages.

Before that, we need one more definition on delay processes. Suppose that  $\sigma \in \mathcal{T}(p, \omega)$  for some process  $p$ , then delay processes of  $p \xrightarrow{\sigma}$  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  $p \xrightarrow{\sigma}^a p'$ .

**Lemma 5.3.2** Let  $\mathcal{L} = (\Sigma, \Psi)$  be a weak  $\omega$ -failure language, and  $C(x_1, \dots, x_n)$  be any context of  $n$  holes. Suppose that  $\zeta$  and  $\xi$  are any two closed  $\Sigma$  substitution mapping  $x_i$  into  $p_i$  and  $q_i$ , respectively. If for all  $1 \leq i \leq n$ ,  $p_i \sim_t q_i$ , then

1) for any trace  $\sigma \in \mathcal{T}(C(p_1, \dots, p_n), \omega)$  and some context  $C'$  such that  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$ , there exist  $q'_1, \dots, q'_n$  such that  $C(q_1, \dots, q_n) \xrightarrow{\sigma} C'(q'_1, \dots, q'_n)$ , and

2) if there exists a patience ruloid for the  $i$ th argument of context  $C'$  then  $q'_i$  can be any process such that  $q_i \xrightarrow{\sigma_i} q'_i$ , and if there does not exist a patience ruloid for the  $i$ th argument of context  $C'$  then  $q'_i$  can be any delay processes of  $q_i \xrightarrow{\sigma_i}$ , where  $\sigma_i$  is obtained by decomposing  $\sigma$  into the weak traces of subprocess  $p_i$ .

**Proof** Suppose that  $\sigma$  is a trace of  $C(p_1, \dots, p_n)$ , and  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$ . By Lemma 5.3.1, when  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$ , we have  $p_i \xrightarrow{\sigma_i} p'_i$  for all  $1 \leq i \leq n$ . Then, by  $p_i \sim_t q_i$ , we have  $q_i \xrightarrow{\sigma_i}$  for all  $1 \leq i \leq n$ .

For  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$ , we have  $C(p_1, \dots, p_n) = C_0(p_{10}, \dots, p_{n0}) \xrightarrow{\alpha_1} C_1(p_{11}, \dots, p_{n1}) \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_m} C_m(p_{1m}, \dots, p_{nm}) = C'(p'_1, \dots, p'_n)$ , where  $\forall 1 \leq j \leq m : \alpha_j \in Act \cup \{\tau\}$  and  $\sigma' = \alpha_1 \dots \alpha_m$  is equivalent to  $\sigma$  if all its  $\tau$  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  $C(q_1, \dots, q_n) \xrightarrow{\sigma} C'(q'_1, \dots, q'_n)$  for some  $q'_1, \dots, q'_n$ . Furthermore, if there exists a patience ruloid for the  $i$ th argument of context  $C'$  then  $q'_i$  can be any process such that  $q_i \xrightarrow{\sigma_i} q'_i$ , and if there does not exist a patience ruloid for the  $i$ th argument of context  $C'$  then  $q'_i$  can be any delay processes of  $q_i \xrightarrow{\sigma_i}$ .

We will prove it by making an induction on  $k$ .

1)  $k = 0$ . Then,  $C = C'$  and only patience ruloids are applied when  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$  and thus  $\sigma = \tau^*$ . By Lemma 5.3.1,  $\sigma_i = \tau^*$  for all  $1 \leq i \leq n$ . Therefore, there must exist  $q'_1, \dots, q'_n$  such that  $C(q_1, \dots, q_n) \xrightarrow{\tau^*} C'(q'_1, \dots, q'_n)$  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  $i$ th argument of context  $C'$  then  $q'_i$  can be any process such that  $q_i \xrightarrow{\tau^*} q'_i$  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  $i$ th 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,  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C''(p''_1, \dots, p''_n)$  and  $C''(p''_1, \dots, p''_n) \xrightarrow{\delta} C'(p'_1, \dots, p'_n)$ , where the first  $k - 1$  plain ruloids of  $r_1 r_2 \dots r_k$  are applied when  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C''(p''_1, \dots, p''_n)$  and the  $k$ th plain ruloid are applied when  $C''(p''_1, \dots, p''_n) \xrightarrow{\delta} C'(p'_1, \dots, p'_n)$ .

By Lemma 5.3.1, there exist  $\sigma_i, \delta_i$  for all  $1 \leq i \leq n$  such that  $p_i \xrightarrow{\sigma_i} p''_i$  and  $p''_i \xrightarrow{\delta_i} p'_i$ . By  $p_i \sim_t q_i$ , we have  $q_i \xrightarrow{\sigma_i \delta_i}$ .

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  $i$ th argument of context  $C''$  then  $q''_i$  can be any process such that

$q_i \xrightarrow{\sigma_i} q''_i$ , and if there does not exist a patience ruloid for the  $i$ th argument of context  $C''$  then  $q''_i$  can be any delay process of  $q_i \xrightarrow{\sigma_i}$ .

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

Suppose that the  $k$ th ruloid  $r_k$  is in form (1). Then, by the definition of the weak  $\omega$ -failure format, all arguments in  $I$  have corresponding patience ruloids since they are all active arguments of  $C''$  by Definition 5.2.5. Therefore, by the soundness property of the ruloids, we may apply the patience ruloids for the arguments in  $I$  and obtain  $C''(q''_1, \dots, q''_n) \xrightarrow{\tau^*} C''(q'''_1, \dots, q'''_n)$ , such that  $q''_i \equiv q'''_i$  if  $i \notin I$  and  $q'''_i \xrightarrow{\delta_i}$  if  $i \in I$ . Then, also by the soundness property of the ruloids, ruloid  $r_k$  will be applied and  $C''(q'''_1, \dots, q'''_n) \xrightarrow{\delta} C'(q''''_1, \dots, q''''_n)$ , where  $q''''_i \equiv q''_i$  if  $i \notin I$  and  $q''''_i$  is any process satisfying  $q''''_i \xrightarrow{\delta_i} q''''_i$  if  $i \in I$ .

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

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

As a strengthened results of the above lemma, Lemma 5.3.3 below will show that, in weak finite failure languages and weak  $\omega$ -failure languages, if the  $i$ th argument is neither an active argument nor a receiving argument, i.e., is an other argument, then  $q'_i$  can be  $q_i$  or any process such that  $q_i \xrightarrow{\sigma_i} q'_i$ . Though we only prove this lemma in weak  $\omega$ -failure languages, it also holds in weak finite failure languages.

**Lemma 5.3.3** Let  $\mathcal{L} = (\Sigma, \Psi)$  be a weak  $\omega$ -failure language, and  $C(x_1, \dots, x_n)$  be any context of  $n$  holes. Suppose that  $\zeta$  and  $\xi$  are any two closed  $\Sigma$  substitution mapping  $x_i$  into  $p_i$  and  $q_i$ , respectively. For all  $1 \leq i \leq n$ ,  $p_i \sim_t q_i$ , and thus for any trace  $\sigma \in \mathcal{T}(C(p_1, \dots, p_n), \omega)$  and some context  $C'$  with  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$ , there exist  $q'_1, \dots, q'_n$  such that  $C(q_1, \dots, q_n) \xrightarrow{\sigma} C'(q'_1, \dots, q'_n)$ . Now, if the  $i$ th argument of  $C'$  is an other argument, then  $q'_i$  can be  $q_i$  or any process such that  $q_i \xrightarrow{\sigma_i} q'_i$ .

**Proof** Like the proof of Lemma 5.3.2, suppose that the sequence of plain ruloids applied in the procedure  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$  is  $r_1 r_2 \dots r_k$ .

We will prove it by making an induction on  $k$ .

1)  $k = 0$ . Then,  $C = C'$  and only patience ruloids are applied when  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$  and thus  $\sigma = \tau^*$ . In this case, we can let  $q'_i$  be  $q_i$  if the  $i$ th argument of  $C'$  is an other argument.

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

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

However, observe that the  $i$ th argument of  $C''$  cannot be in set  $I$  of ruloid  $r_k$ . Or else, the  $i$ th argument of  $C'$  will

be at least a receiving argument by Definition 5.2.5. We separate it into two cases:

i) If the  $i$ th argument of  $C''$  is an active argument or a receiving argument, then it has a patience ruloid since the language  $\mathcal{L}$  is a weak  $\omega$ -failure language. Therefore, by

Lemma 5.3.2,  $q'_i$  can be any process such that  $q_i \xrightarrow{\sigma_i \delta_i} q'_i$  with  $\delta_i = \tau^*$ .

ii) If the  $i$ th argument of  $C''$  is an other argument, then, by the induction hypothesis,  $q'_i$  can be  $q_i$  or any process such that  $q_i \xrightarrow{\sigma_i} q'_i$ . Now, since  $\mathcal{L}$  is a weak  $\omega$ -failure language, no patience ruloid for the  $i$ th argument of  $C''$  and the  $i$ th argument of  $C'$ . Therefore,  $q'_i$  is just  $q_i$ , and thus  $q'_i$  can be  $q_i$  or any process such that  $q_i \xrightarrow{\sigma_i} q'_i$ . Then, by  $\delta_i = \tau^*$ ,  $q'_i$  can be  $q_i$  or any process such that  $q_i \xrightarrow{\sigma_i \delta_i} q'_i$ .  $\square$

Note that, the above lemma does not hold in weak 1-failure languages since patience ruloids for receiving arguments are needed when proving it. Therefore, it will not be used when proving the congruence theorem for the weak 1-failure format.

The following lemma shows that, in a weak finite failure language, if a process executes an action sequence (weak trace) with length  $k$ , then, at the same time, all the lengths of the action sequences executed by its subprocesses may not exceed  $k$ . This lemma also holds in weak 1-failure languages.

**Lemma 5.3.4** Let  $\mathcal{L}$  be a weak finite failure language.  $C(x_1, \dots, x_n)$  is any context of  $n$  holes in  $\mathcal{L}$ . Suppose that  $\zeta$  is any closed  $\Sigma$  substitution mapping  $x_i$  into  $p_i$ . If  $C(p_1, \dots, p_n)$  is a process and  $\sigma$  is a weak trace of  $C(p_1, \dots, p_n)$ , then each  $p_i$  will execute a weak trace  $\sigma_i$  at the same time, for  $1 \leq i \leq n$ . We can conclude that  $\forall 1 \leq i \leq n : |\sigma_i| \leq k$  when  $|\sigma| = k$ .

**Proof** By Lemma 5.3.1, if  $C(p_1, \dots, p_n)$  is a process and  $\sigma$  is a weak trace of  $C(p_1, \dots, p_n)$ , then each  $p_i$  will execute a weak trace  $\sigma_i$  at the same time. We also need to prove that  $|\sigma_i| \leq k$  when  $|\sigma| = k$ .

We will prove it by making an induction on  $|\sigma| = k$ .

1)  $k = 0$ . Then,  $C(p_1, \dots, p_n) \xrightarrow{\tau^*} C'(p'_1, \dots, p'_n)$ . By the definition of the weak  $\omega$ -failure format, the ruloids applied in this procedure can only be patience ruloids or ruloids with  $\tau$  conclusion. By the hypothesis, no rules with  $\tau$  conclusion are present in  $\mathcal{L}$ , and thus, by Proposition 5.2.8, no ruloids with  $\tau$  conclusion are present in  $\mathcal{R}(C, \tau)$ .

Therefore, when  $C(p_1, \dots, p_n) \xrightarrow{\tau^*} C'(p'_1, \dots, p'_n)$ , only patience ruloids are applied. However, from the definition of patience ruloids and its corresponding ruloids in form (2),  $p_i \xrightarrow{\tau} p'_i$  or  $p_i \xrightarrow{\epsilon} p'_i$  for all  $1 \leq i \leq n$ . And  $|\tau^*| = |\epsilon| = 0$ .

2) Assume that, when  $k = m - 1$  with  $m \geq 1$ , we have  $|\sigma_i| \leq k$  when  $|\sigma| = k$ .

3) For  $k = m$ , let  $\sigma = \sigma' \alpha$ . Then, we have  $C(p_1, \dots, p_n) \xrightarrow{\sigma'} C''(p''_1, \dots, p''_n) \xrightarrow{\alpha} C'(p'_1, \dots, p'_n)$ .

Extending it, we obtain that  $C(p_1, \dots, p_n) \xrightarrow{\sigma'} C''(p''_1, \dots, p''_n) \xrightarrow{\tau^*} C^1(p^1_1, \dots, p^1_n) \xrightarrow{\alpha} C^2(p^2_1, \dots, p^2_n) \xrightarrow{\tau^*} C'(p'_1, \dots, p'_n)$ .

By Lemma 5.3.1, for all  $1 \leq i \leq n$ , there exist  $p_i \xrightarrow{\sigma'_i} p''_i \xrightarrow{\sigma^1_i} p^1_i \xrightarrow{\alpha_i} p^2_i \xrightarrow{\sigma^2_i} p'_i$ .

It is trivial that  $|\sigma'| = m - 1$ . Therefore, by the induction hypothesis, we have  $|\sigma'_i| \leq m - 1$ . Also, when  $C''(p''_1, \dots, p''_n) \xrightarrow{\tau^*} C^1(p^1_1, \dots, p^1_n)$  and  $C^2(p^2_1, \dots, p^2_n) \xrightarrow{\tau^*} C'(p'_1, \dots, p'_n)$ ,  $|\tau^*| = 0$ . Therefore, by the induction base, we have  $|\sigma^1_i| = |\sigma^2_i| = 0$ . Furthermore,  $|\alpha_i| \leq 1$  can be obtained by the ruloids of form (2).

In all,  $|\sigma_i| \leq k$  when  $|\sigma| = k$ .  $\square$

The following lemma shows that, in weak 1-failure languages, process  $C(p_1, \dots, p_i, \dots, p_n)$  and  $C(p_1, \dots, p'_i, \dots, p_n)$  have the same sets of next observable actions if  $p_i \xrightarrow{\tau^*} p'_i$  and the  $i$ th argument is not an active argument.

**Lemma 5.3.5** Let  $\mathcal{L} = (\Sigma, \Psi)$  be a weak 1-failure language, and  $C(x_1, \dots, x_n)$  be any context of  $n$  holes. Suppose that  $\zeta$  is any closed  $\Sigma$  substitution mapping  $x_i$  into  $p_i$ . If the  $i$ th argument is not an active argument of  $C(x_1, \dots, x_n)$  and  $p_i \xrightarrow{\tau^*} p'_i$ , then  $\mathcal{T}(C(p_1, \dots, p_i, \dots, p_n), 1) = \mathcal{T}(C(p_1, \dots, p'_i, \dots, p_n), 1)$ .

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

1) If the next ruloid is a patience ruloid, then it should be a patience ruloid for active argument, since  $\mathcal{L}$  is a weak 1-failure language. However, applying the patience ruloid will not produce observable actions for  $C(p_1, \dots, p_i, \dots, p_n)$  and  $C(p_1, \dots, p'_i, \dots, p_n)$ . Because the  $i$ th argument is not an active argument,  $C(p_1, \dots, p_i, \dots, p_j, \dots, p_n) \xrightarrow{\tau} C(p_1, \dots, p_i, \dots, p'_j, \dots, p_n)$  and  $C(p_1, \dots, p'_i, \dots, p_j, \dots, p_n) \xrightarrow{\tau} C(p_1, \dots, p'_i, \dots, p'_j, \dots, p_n)$  when the  $j$ th argument of context  $C$  is an active argument and  $p_j \xrightarrow{\tau} p'_j$ . Now, it is enough to consider the set of next observable actions of  $C(p_1, \dots, p_i, \dots, p'_j, \dots, p_n)$  and  $C(p_1, \dots, p'_i, \dots, p'_j, \dots, p_n)$ .

2) If the next ruloid is a plain ruloid, then it should not be a ruloid with  $\tau$  conclusion, since  $\mathcal{L}$  is a weak 1-failure language. Suppose that the applied ruloid  $r$  is in form (1), then the  $i$ th 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  $A_1 = A_2$ .  $\square$

## 5.4 Weak 1-Failure Format for Weak 1-Failure Equivalence

Now, we will prove the congruence theorem for the weak 1-failure format.

**Theorem 5.4.1** The weak 1-failure format is a congruence format for the weak 1-failure equivalence.

**Proof** It is enough to prove that if  $\forall 1 \leq j \leq n : p_j \sim_f^1 q_j$  then  $C(p_1, \dots, p_n) \sim_f^1 C(q_1, \dots, q_n)$ , where  $C$  is any context of  $n$  holes in a weak 1-failure language  $\mathcal{L}$ . By the symmetry of the alternative characterization of weak 1-failure equivalence in Proposition 3.5, we only need to prove that if  $\forall 1 \leq j \leq n : p_j \sim_f^1 q_j$ , then, for any  $\sigma \in \mathcal{T}(C(p_1, \dots, p_n), \omega)$  and  $C'(p'_1, \dots, p'_n)$  with  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$ , there exists  $C'(q'_1, \dots, q'_n)$  such that  $C(q_1, \dots, q_n) \xrightarrow{\sigma} C'(q'_1, \dots, q'_n)$  and  $\mathcal{T}(C'(q'_1, \dots, q'_n), 1) \subseteq \mathcal{T}(C'(p'_1, \dots, p'_n), 1)$ . Observe that, though it needs only there exists some  $C''(q''_1, \dots, q''_n)$  such that  $C(q_1, \dots, q_n) \xrightarrow{\sigma} C''(q''_1, \dots, q''_n)$  and  $\mathcal{T}(C''(q''_1, \dots, q''_n), 1) \subseteq \mathcal{T}(C'(p'_1, \dots, p'_n), 1)$ , we will prove in the following that we can safely let  $C''$  be  $C'$ , and thus we write  $C''(q''_1, \dots, q''_n)$  as  $C'(q'_1, \dots, q'_n)$ .

By Lemma 5.3.1, when  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$ , there exists  $p_j \xrightarrow{\sigma_j} p'_j$  for all subprocess  $p_j$  with  $1 \leq j \leq n$ . Similarly, for all  $a \in \mathcal{T}(C'(p'_1, \dots, p'_n), 1)$ , when  $C'(p'_1, \dots, p'_n) \xrightarrow{a} \delta_j$ , we have  $p'_j \xrightarrow{\delta_j}$  for all subprocess  $p_j$  with  $1 \leq j \leq n$ . Let  $A'_j$  be the set of all  $\delta_j$ . Note that, for some  $a \in A$ , there may exist several  $\delta_j$  corresponding with

it. And we should add all of them into the set  $A'_j$ .

Then, by Lemma 5.3.4, the exclusion of the rules with  $\tau$ -conclusion will make the length of  $\delta_j$  not exceed 1, i.e.,  $\forall \delta_j \in A'_j : |\delta_j| \leq 1$ . Therefore, for all  $1 \leq j \leq n$ , we have  $A'_j \subseteq \mathcal{T}(p'_j, 1)$ .

Now, by  $p_j \sim_f^1 q_j$  and Proposition 3.5, there exists some  $q'_j$  such that  $q_j \xrightarrow{\sigma_j} q'_j$  and  $\mathcal{T}(q'_j, 1) \subseteq \mathcal{T}(p'_j, 1)$ .

By Lemma 5.3.2,  $\sigma$  is also a trace of  $C(q_1, \dots, q_n)$  and  $C(q_1, \dots, q_n) \xrightarrow{\sigma} C'(q'_1, \dots, q'_n)$ . Observe that, it is possible that  $q'_j$  is not equivalent to  $q_j$ . The reason is that, from Lemma 5.3.2, we can only obtain that, there exist  $q'_1, \dots, q'_n$  such that  $C(q_1, \dots, q_n) \xrightarrow{\sigma} C'(q'_1, \dots, q'_n)$ , but not the very  $q'_1, \dots, q'_n$  which are obtained from  $p_j \sim_f^1 q_j$  and Proposition 3.5.

By Lemma 5.3.2, if the  $j$ th argument of  $C'$  is an active argument, then  $q'_j$  can be any process such that  $q_j \xrightarrow{\sigma_j} q''_j$  and thus we may let  $q'_j$  be  $q''_j$  safely since  $q_j \xrightarrow{\sigma_j} q'_j$ . On the other hand, if the  $j$ th argument of  $C'$  is not an active argument, then no patience ruloids are present in the language. Therefore,  $q'_j$  can be any delay process of  $q_i \xrightarrow{\sigma_j}$ . Note that, for  $q'_j$ , there must exist some delay process  $q''_j$  such that  $q_j \xrightarrow{\sigma_j} q''_j \xrightarrow{\tau^*} q'_j$ .

Now, by Lemma 5.3.5 and  $q''_j \xrightarrow{\tau^*} q'_j$ , we assert that  $\mathcal{T}(C'(q'_1, \dots, q'_n), 1) = \mathcal{T}(C'(q''_1, \dots, q''_n), 1)$ . Note that, there may exist several arguments of  $C'$  such that they are not active arguments. However, we can finally obtain  $\mathcal{T}(C'(q'_1, \dots, q'_n), 1) = \mathcal{T}(C'(q''_1, \dots, q''_n), 1)$  by applying Lemma 5.3.5 for several times.

Moreover, by  $\mathcal{T}(q'_j, 1) \subseteq \mathcal{T}(p'_j, 1)$ , we have  $\mathcal{T}(C'(q'_1, \dots, q'_n), 1) \subseteq \mathcal{T}(C'(p'_1, \dots, p'_n), 1)$ .

Finally, we obtain that  $\mathcal{T}(C'(q'_1, \dots, q'_n), 1) \subseteq \mathcal{T}(C'(p'_1, \dots, p'_n), 1)$ .  $\square$

## 5.5 Weak Finite Failure Format for Weak $i$ -Failure Equivalence

The following is the congruence theorem for the weak finite failure format.

**Theorem 5.5.1** The weak finite failure format is a congruence format for the weak  $i$ -failure equivalence with  $1 < i < \omega$ .

**Proof** Similar with Theorem 5.4.1, it is enough to prove that if  $\forall 1 \leq j \leq n : p_j \sim_f^i q_j$  then  $C(p_1, \dots, p_n) \sim_f^i C(q_1, \dots, q_n)$ , where  $C$  is any context of  $n$  holes in a weak 1-failure language  $\mathcal{L}$ . By the symmetry of the alternative characterization of weak  $i$ -failure equivalence in Proposition 3.5, we only need to prove that if  $\forall 1 \leq j \leq n : p_j \sim_f^i q_j$ , then, for any  $\sigma \in \mathcal{T}(C(p_1, \dots, p_n), \omega)$  and  $C'(p'_1, \dots, p'_n)$  with  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$ , there exists  $C'(q'_1, \dots, q'_n)$  such that  $C(q_1, \dots, q_n) \xrightarrow{\sigma} C'(q'_1, \dots, q'_n)$  and  $\mathcal{T}(C'(q'_1, \dots, q'_n), i) \subseteq \mathcal{T}(C'(p'_1, \dots, p'_n), i)$ .

By Lemma 5.3.1, when  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$ , there exists  $p_j \xrightarrow{\sigma_j} p'_j$  for all subprocess  $p_j$  with  $1 \leq j \leq n$ .

Similarly, for all  $\delta \in \Phi$ , when  $C'(p'_1, \dots, p'_n) \xrightarrow{\delta}$ , we have  $p'_j \xrightarrow{\delta_j}$  for all subprocess  $p_j$  with  $1 \leq j \leq n$ . Let  $\Phi'_j$  be the set of all  $\delta_j$ . Note that, for some  $\delta \in \Phi$ , there may exist several  $\delta_j$  corresponding with it. And we should add all of them into the set  $\Phi'_j$ .

By Lemma 5.3.4, the exclusion of the rules with  $\tau$ -conclusion will make the length of  $\delta_j$  not exceed  $i$ , i.e.,

$\forall \delta_j \in \Phi'_j : |\delta_j| \leq i$ . Therefore, for all  $1 \leq j \leq n$ , we have  $\Phi'_j \subseteq \mathcal{T}(p'_j, i)$ .

Now, by  $p_j \sim_f^i q_j$ , there exists some  $q'_j$  such that  $q_j \xrightarrow{\sigma_j} q'_j$  and  $\mathcal{T}(q'_j, i) \subseteq \mathcal{T}(p'_j, i)$ .

By Lemma 5.3.2,  $\sigma$  is also a trace of  $C(q_1, \dots, q_n)$  and  $C(q_1, \dots, q_n) \xrightarrow{\sigma} C'(q'_1, \dots, q'_n)$ . Moreover,

1) if the  $j$ th argument of  $C'$  is a receiving argument or an active argument, then it has a patience ruloid since the language is a weak finite failure language. Therefore, by Lemma 5.3.2, we can let  $q'_j$  be  $q_j$  since  $q'_j$  can be any process such that  $q_j \xrightarrow{\sigma_j} q''_j$ , and

2) if the  $j$ th argument of  $C'$  is an other argument, then, by Lemma 5.3.3, we can let  $q'_j$  be  $q_j$  or any process such that  $q_j \xrightarrow{\sigma_j} q''_j$ . We want to separate it into two cases:

i) if  $q'_j$  is any process such that  $q_j \xrightarrow{\sigma_j} q''_j$ , then we can also let  $q'_j$  be  $q''_j$ .

ii) if  $q'_j$  is  $q_j$ , then  $q'_j$  and  $q''_j$  are both delay processes since the  $j$ th argument of  $C'$  is an other argument and thus no patience ruloid for it. Therefore, we can obtain that  $q'_j \equiv q''_j \equiv q_j$ .

In all, we can always let  $q'_j$  be  $q''_j$ , i.e.,  $C(q_1, \dots, q_n) \xrightarrow{\sigma} C'(q'_1, \dots, q'_n)$ .

Moreover, by  $\mathcal{T}(q'_j, i) \subseteq \mathcal{T}(p'_j, i)$ , we have  $\mathcal{T}(C'(q'_1, \dots, q'_n), i) \subseteq \mathcal{T}(C'(p'_1, \dots, p'_n), i)$ .  $\square$

## 5.6 Weak $\omega$ -Failure Format for Weak $\omega$ -Failure Equivalence

The congruence theorem for the weak  $\omega$ -failure format is as follows.

**Theorem 5.6.1** The weak  $\omega$ -failure format is a congruence format for the weak  $\omega$ -failure equivalence.

**Proof** Similar with Theorem 5.4.1, it is enough to prove that if  $\forall 1 \leq j \leq n : p_j \sim_f^\omega q_j$  then  $C(p_1, \dots, p_n) \sim_f^\omega C(q_1, \dots, q_n)$ , where  $C$  is any context of  $n$  holes in a weak 1-failure language  $\mathcal{L}$ . By the symmetry of the alternative characterization of weak  $\omega$ -failure equivalence in Proposition 3.5, we only need to prove that if  $\forall 1 \leq j \leq n : p_j \sim_f^\omega q_j$ , then, for any  $\sigma \in \mathcal{T}(C(p_1, \dots, p_n), \omega)$  and  $C'(p'_1, \dots, p'_n)$  with  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$ , there exists  $C'(q'_1, \dots, q'_n)$  such that  $C(q_1, \dots, q_n) \xrightarrow{\sigma} C'(q'_1, \dots, q'_n)$  and  $\mathcal{T}(C'(q'_1, \dots, q'_n), \omega) \subseteq \mathcal{T}(C'(p'_1, \dots, p'_n), \omega)$ .

By Lemma 5.3.1, when  $C(p_1, \dots, p_n) \xrightarrow{\sigma} C'(p'_1, \dots, p'_n)$ , there exists  $p_j \xrightarrow{\sigma_j} p'_j$  for all subprocess  $p_j$  with  $1 \leq j \leq n$ .

Similarly, for all  $\delta \in \Phi$ , when  $C'(p'_1, \dots, p'_n) \xrightarrow{\delta}$ , we have  $p'_j \xrightarrow{\delta_j}$  for all subprocess  $p_j$  with  $1 \leq j \leq n$ . Let  $\Phi'_j$  be the set of all  $\delta_j$ . Note that, for some  $\delta \in \Phi$ , there may exist several  $\delta_j$  corresponding with it. And we should add all of them into the set  $\Phi'_j$ .

Therefore, for all  $1 \leq j \leq n$ , we have  $\Phi'_j \subseteq \mathcal{T}(p'_j, \omega)$ .

Now, by  $p_j \sim_f^\omega q_j$ , there exists some  $q'_j$  such that  $q_j \xrightarrow{\sigma_j} q'_j$  and  $\mathcal{T}(q'_j, \omega) \subseteq \mathcal{T}(p'_j, \omega)$ .

By Lemma 5.3.2,  $\sigma$  is also a trace of  $C(q_1, \dots, q_n)$  and  $C(q_1, \dots, q_n) \xrightarrow{\sigma} C'(q'_1, \dots, q'_n)$ . Moreover,

1) if the  $j$ th argument of  $C'$  is a receiving argument or an active argument, then it has a patience ruloid since the language is a weak  $\omega$ -failure language. Therefore, by Lemma 5.3.2, we can let  $q'_j$  be  $q_j$  since  $q'_j$  can be any process such that  $q_j \xrightarrow{\sigma_j} q''_j$ , and

2) if the  $j$ th argument of  $C'$  is an other argument, then, by Lemma 5.3.3, we can let  $q'_j$  be  $q_j$  or any process such

that  $q_j \xRightarrow{\sigma_j} q'_j$ . We want to separate it into two cases:

i) if  $q'_j$  is any process such that  $q_j \xRightarrow{\sigma_j} q'_j$ , then we can also let  $q'_j$  be  $q'_j$ .

ii) if  $q'_j$  is  $q_j$ , then  $q'_j$  and  $q''_j$  are both delay processes since the  $j$ th argument of  $C'$  is an other argument and thus no patience ruloid for it. Therefore, we can obtain that  $q'_j \equiv q'_j \equiv q_j$ .

In all, we can always let  $q'_j$  be  $q'_j$ , i.e.,  $C(q_1, \dots, q_n) \xRightarrow{\sigma} C'(q'_1, \dots, q'_n)$ .

Moreover, by  $\mathcal{T}(q'_j, \omega) \subseteq \mathcal{T}(p'_j, \omega)$ , we have  $\mathcal{T}(C'(q'_1, \dots, q'_n), \omega) \subseteq \mathcal{T}(C'(p'_1, \dots, p'_n), \omega)$ .  $\square$

## 6 Conclusions

In the paper, we first introduce a series of behavioral equivalences, named weak parametric failure equivalences, which take the weak failure equivalence and the weak impossible future equivalence as their special cases. Then, based on the idea of Structural Operational Semantics, rule formats are proposed to meet these behavioral equivalences. By the intuitive opinions and formal proofs, we have shown that these rule formats are all congruence formats of their corresponding behavioral equivalences.

An advantage of these rule formats is that we can easily decide whether a behavioral equivalence is congruent under a given operator. In fact, for any behavioral equivalences, one of the most frequently-asked problems 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 (Milner, 1989), CSP (Hoare, 1985) and ACP (Baeten, 1990). Generally, there exist two ways to deal with this problem: The first one is to prove the congruence properties of these operators one by one. It is a straightforward and intuitive way, but may be somewhat clumsy. The second one is to pursue a rule format for this specified behavioral equivalence. And the given behavioral equivalence can be preserved under any operators in this format.

However, we have noticed that equivalences in strong notion, such as strong bisimulation and decorated trace semantics, were paid more attentions to than equivalences in weak notion, such as weak bisimulation and testing theory. In fact, almost all classical strong equivalences have found their corresponding rule formats, but much less works have been done on the rule formats of weak equivalences, especially on the rule formats of the equivalences in testing theoretical notions. And more specifically, no rule formats have been presented to be congruence formats for the weak failure equivalence or the weak impossible future equivalence. The difference may exist in the increasing complexity after introducing  $\tau$  transitions by weak equivalences. The aim of our paper is to make a progress along this direction.

## References

R.J. van Glabbeek. The Linear Time - Branching Time Spectrum I: The Semantics of Concrete, Sequential Processes. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, Handbook of Process Algebra, chapter 1, pages 3-100. Elsevier, 2001.

R.J. van Glabbeek. The Linear Time - Branching Time Spectrum II: The semantics of sequential systems with silent moves. In E. Best, editor, Concur'93, LNCS 715, pages 66-81. Springer-Verlag, 1993.

R. Milner. Communication and Concurrency. Prentice-Hall, 1989.

M.R. Mousavi, M.A. Reniers, J.F. Groote (2007). SOS formats and meta-theory: 20 years after. Theoretical Computer Science 373, pages 238-272.

L. Aceto, W.J. Fokkink and C. Verhoef. Structural Operational Semantics. In J.A. Bergstra, A. Ponse and S.A. Smolka, editors, Handbook of Process Algebra, Chapter 3, pages 197-292. Elsevier, 2001.

G.D. Plotkin. A Structural Approach to Operational Semantics. The Journal of Logic and Algebraic Programming 60-61, 17-139, 2004.

J.C.M. Baeten and W.P. Weijland. Process Algebra. volume 18 of Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 1990.

J.F. Groote and F.W. Waandrager. Structural Operational Semantics and Bisimulation as a Congruence. Information and Computation 100(2), 202-260, 1992.

R.D. Simone. Higher-level synchronising devices in Meiji-SCCS. Theoretical Computer Science 37, 245-267, 1985.

J.F. Groote. Transition System Specifications with Negative Premises. Theoretical Computer Science 118, 263-299, 1993.

B. Bloom, S. Istrail and A. R. Meyer. Bisimulation can't be Traced. Journal of the ACM 42(1), 232-268, 1995.

A. Rensink, W. Vogler. Fair testing. Information and Computation, Volume 205, Issue 2, February 2007, Pages 125-198.

R.J. van Glabbeek, On Cool Congruence Formats for Weak Bisimulations. In D.V. Hung and M. wirsing, editors, Proceedings International Colloquium on Theoretical Aspects of Computing, LNCS 3722, page 331-346. Springer, 2005.

I. Ulidowski, Finite Axiom Systems for Testing Preorder and De Simone Process Languages. Theoretical computer Science, 239(1):97-139, 2000.

C.A.R. Hoare, Communicating Sequential Processes, Prentice-Hall, Englewood Cliffs, NJ, 1985.

R.J. van Glabbeek, The Meaning of Negative Premises in Transition System Specification II. The Journal of Logic and Algebraic Programming 60-61, pages 229-258, 2004.

B. Bloom. Structural operational semantics for weak bisimulations. Theoretical Computer Science 146, pages 27-68, 1995.

B. Bloom. Ready Simulation, Bisimulation, and the Semantics of CCS-Like Languages. PhD thesis, MIT, 1990.

R. Milner. Communicating and Mobile Systems: the  $\pi$ -Calculus. Cambridge University Press, 1990.