

# Optimization Rules Mathematical Proofs

Islam Abdelhalim and Steve Schneider

Department of Computing, University of Surrey

Technical Report Draft

## 1 Introduction

This report provides the detailed mathematical proofs of the theorems included in the paper [1]. Each theorem supports an optimization rule (Opti-Rule) to make sure that it does not eliminate important information from the CSP model that is required for a specific property checking (e.g., deadlock). The report includes also an additional theorem (Theorem(3) not included in [1]) which supports Opti-Rule(3) that is described in this report as well.

The incoming sections address Opti-Rule(1,2 and 3) and the proof of their theorems.

## 2 Opti-Rule(1): Removing passive processes

When checking deadlock, any passive process can be removed from the system without affecting the deadlock checking result. The passive process is defined as any process that never refusing any interaction (always willing to interact). Theorem(1) defines the formal representation of Opti-Rule(1).

**Theorem(1).** *If  $P_2$  is non-divergent and  $(s, X_{P_2}) \in \mathcal{F}(P_2) \Rightarrow X_{P_2} = \{\}$  (passive process), then for any process  $P_1$ :  $P_1$  is deadlock free  $\Leftrightarrow P_1 \parallel P_2$  is deadlock free.*

### **Proof**

$P_2 \Leftrightarrow RUN$                       *From process RUN definition in [2]*

$P_1 \parallel P_2 = P_1 \parallel RUN$

$= P_1$                       *From Low 3B in Section 2.2.1 of [3]*

$P_1$  is deadlock free  $\Leftrightarrow P_1 \parallel P_2$  is deadlock free                      *(as required)*

□ *Theorem(1)*

### 3 Opti-Rule(2): Removing abandoned events

When checking deadlock, any abandoned event can be removed from the system without affecting the deadlock checking result. The abandoned event is defined as any event that no other processes in the system synchronize on it. Theorem(2) defines the formal representation of Opti-Rule(2), nevertheless the following definitions and lemmas are required to proof this theorem.

**Definition(2).** The operator  $REMOVE_a(P)$ , where  $a$  is the abandoned event, is defined as follow:

$$REMOVE_a(STOP) = STOP$$

$$REMOVE_a(a \rightarrow P) = REMOVE_a(P)$$

$$REMOVE_a(a?x!y \rightarrow P(x)) = \prod_x REMOVE_a(P(x))$$

$$REMOVE_a(b \rightarrow P) = b \rightarrow REMOVE_a(P)$$

$$REMOVE_a(b?x!y \rightarrow P(x)) = b?x!y \rightarrow REMOVE_a(P(x))$$

$$REMOVE_a(P \sqcap Q) = REMOVE_a(P) \sqcap REMOVE_a(Q) \quad \text{From Section 1.2 of [2]}$$

$$REMOVE_a(P \square Q) = REMOVE_a(P) \square REMOVE_a(Q) \\ , \text{ if } a \notin \text{initials}(P) \wedge a \notin \text{initials}(Q)$$

**Lemma(2.1).** If  $a \notin \text{initials}(P) \wedge a \notin \text{initials}(Q)$ , then:

$$\underbrace{P \setminus \{a\} \square Q \setminus \{a\}}_{LHS} = \underbrace{(P \square Q) \setminus \{a\}}_{RHS}$$

**Proof**

To proof the equality in this lemma, we should proof that the LHS equals to the RHS, and vice versa. Thus, this proof is done on two stages (1 and 2). The proof below depends on the failures definition of the external choice and hiding in Section 8.2 of [4] illustrated below:

$$\mathcal{F}(P \square Q) = \{ (\langle \rangle, X) \mid (\langle \rangle, X) \in \mathcal{F}(P) \cap \mathcal{F}(Q) \} \\ \cup \\ \{ (s, X) \mid s \neq \langle \rangle \wedge (s, X) \in \mathcal{F}(P) \cup \mathcal{F}(Q) \}$$

$$\mathcal{F}(P \setminus \{a\}) = \{ (s \setminus \{a\}, X) \mid (s, X \cup \{a\}) \in \mathcal{F}(P) \}$$

Therefore the failures definition of the LHS:

$$\mathcal{F}(P \setminus \{a\} \square Q \setminus \{a\}) = \{ (\langle \rangle, X) \mid (\langle \rangle, X) \in \mathcal{F}(P \setminus \{a\}) \cap \mathcal{F}(Q \setminus \{a\}) \} \\ \cup \\ \{ (s, X) \mid s \neq \langle \rangle \wedge (s, X) \in \mathcal{F}(P \setminus \{a\}) \cup \mathcal{F}(Q \setminus \{a\}) \}$$

And the failures definition of the RHS:

$$\mathcal{F}((P \square Q) \setminus \{a\}) = \{ (s \setminus \{a\}, X) \mid s = \langle \rangle \wedge (\langle \rangle, X \cup \{a\}) \in \mathcal{F}(P) \cap \mathcal{F}(Q) \\ \vee \\ s \neq \langle \rangle \wedge (s, X \cup \{a\}) \in \mathcal{F}(P) \cup \mathcal{F}(Q) \}$$

**Stage(1) proof (LHS = RHS)**

*Consider :*  $(s, X) \in \mathcal{F}(P \setminus \{a\} \square Q \setminus \{a\})$

*Proof :*  $(s, X) \in \mathcal{F}((P \square Q) \setminus \{a\})$

**Case(1.1) :**  $s = \langle \rangle$

$$(\langle \rangle, X) \in \mathcal{F}(P \setminus \{a\} \square Q \setminus \{a\})$$

$$(\langle \rangle, X) \in \mathcal{F}(P \setminus \{a\}) \cap \mathcal{F}(Q \setminus \{a\}) \quad \text{From the LHS failures definition}$$

$$\exists s_P \cdot (s_P, X \cup \{a\}) \in \mathcal{F}(P) \wedge s_P \setminus \{a\} = \langle \rangle \quad \text{From the hiding failures definition}$$

$$\exists s_Q \cdot (s_Q, X \cup \{a\}) \in \mathcal{F}(Q) \wedge s_Q \setminus \{a\} = \langle \rangle \quad \text{From the hiding failures definition}$$

**Case(1.1a) :**  $s_P \neq \langle \rangle$

$$(s_P, X \cup \{a\}) \in \mathcal{F}(P) \cup \mathcal{F}(Q)$$

$$(s_P \setminus \{a\}, X) \in \mathcal{F}((P \square Q) \setminus \{a\}) \quad \text{From the RHS failures definition}$$

$$(\langle \rangle, X) \in \mathcal{F}((P \square Q) \setminus \{a\}) \quad \text{Because, } s_P \setminus \{a\} = \langle \rangle$$

*as required*

**Case(1.1b) :**  $s_Q \neq \langle \rangle$

*Similar as Case(1.1a), but with  $s_Q$*

**Case(1.1c) :**  $s_P = \langle \rangle \wedge s_Q = \langle \rangle$

$$(\langle \rangle, X \cup \{a\}) \in \mathcal{F}(P) \cap \mathcal{F}(Q) \quad \text{From } s_P \text{ and } s_Q \text{ hypotheses}$$

$$(s \setminus \{a\}, X) \in \mathcal{F}((P \square Q) \setminus \{a\}) \quad \text{From the RHS failures definition}$$

$$(\langle \rangle, X) \in \mathcal{F}((P \square Q) \setminus \{a\}) \quad \text{as required}$$

**Case(1.2)** :  $s \neq \langle \rangle$

$$(s, X) \in \mathcal{F}(P \setminus \{a\} \sqcap Q \setminus \{a\})$$

$$(s, X) \in \mathcal{F}(P \setminus \{a\}) \cup \mathcal{F}(Q \setminus \{a\}) \quad \text{From the LHS failures definition}$$

**Case(1.2a)** :  $(s, X) \in \mathcal{F}(P \setminus \{a\})$

$$\exists s_P \cdot (s_P, X \cup \{a\}) \in \mathcal{F}(P) \wedge s_P \setminus \{a\} = s \quad \text{From the hiding failures definition}$$

$$(s_P, X \cup \{a\}) \in \mathcal{F}(P) \cup \mathcal{F}(Q)$$

$$(s_P \setminus \{a\}, X) \in \mathcal{F}((P \sqcap Q) \setminus \{a\}) \quad \text{From the RHS failures definition}$$

$$(s, X) \in \mathcal{F}((P \sqcap Q) \setminus \{a\}) \quad \text{as required}$$

**Case(1.2b)** :  $(s, X) \in \mathcal{F}(Q \setminus \{a\})$

*Similar as Case(1.2a), but with  $s_Q$ .*

**Stage(2) proof (RHS = LHS)**

*Consider* :  $(s, X) \in \mathcal{F}((P \sqcap Q) \setminus \{a\})$

*Proof* :  $(s, X) \in \mathcal{F}(P \setminus \{a\} \sqcap Q \setminus \{a\})$

**Case(2.1)** :  $s = \langle \rangle$

$$(\langle \rangle, X) \in \mathcal{F}((P \sqcap Q) \setminus \{a\})$$

$$(s \setminus \{a\}, X \cup \{a\}) \in \mathcal{F}(P) \cap \mathcal{F}(Q) \quad \text{From the RHS failures definition}$$

*The following sub – proof to show that :  $s \setminus \{a\} = \langle \rangle$*

Let,  $(s, X) \in \mathcal{F}(R \setminus A) \Rightarrow \exists s' \cdot s' \setminus A = s \wedge (s', X \cup A) \in \mathcal{F}(R)$  *From the hiding failures definition*

Having,  $R = P \square Q$ ,  $A = \{a\}$ , and  $S' \setminus A = \langle \rangle$  (sub-proof target)

Then,  $s' \setminus \{a\} = \langle \rangle \wedge (s', X \cup \{a\}) \in \mathcal{F}(P \square Q)$

Using contradiction to proof that  $s' = \langle \rangle$ :

If  $s' \neq \langle \rangle$  then  $s' = \langle a \rangle \hat{\ } s'' \wedge (s', X \cup \{a\}) \in \mathcal{F}(P) \vee (s', X \cup \{a\}) \in \mathcal{F}(Q)$

$\Rightarrow \langle a \rangle \hat{\ } s'' \in \text{traces}(P) \vee \langle a \rangle \hat{\ } s'' \in \text{traces}(Q)$

Since this result contradict with Lemma(XX) of [XX], therefore  $s' = \langle \rangle$

Then,  $s' \setminus \{a\} = \langle \rangle$  as required

$(\langle \rangle, X \cup \{a\}) \in \mathcal{F}(P) \cap \mathcal{F}(Q)$  *From the previous sub-proof*

$(\langle \rangle, X) \in \mathcal{F}(P \setminus \{a\}) \cap \mathcal{F}(Q \setminus \{a\})$  *From the hiding definition*

$(\langle \rangle, X) \in \mathcal{F}(P \setminus \{a\}) \square \mathcal{F}(Q \setminus \{a\})$  *From the LHS failures definition*

as required

**Case(2.2)** :  $s \neq \langle \rangle$

$(s, X) \in \mathcal{F}((P \square Q) \setminus \{a\})$

Let,  $(s, X) \in \mathcal{F}(P \square Q \setminus \{a\}) \Rightarrow \exists s' \cdot s' \setminus \{a\} = s \wedge (s', X \cup \{a\}) \in \mathcal{F}(P \square Q)$   
*From the hiding failures definition*

Then,  $(s', X \cup \{a\}) \in \mathcal{F}(P) \cup \mathcal{F}(Q)$  *From the RHS failures definition &  $s' \setminus \{a\} = s$*

$(s' \setminus \{a\}, X) \in \mathcal{F}(P \setminus a) \cup \mathcal{F}(Q \setminus \{a\})$  *From the hiding failures definition*

$(s, X) \in \mathcal{F}(P \setminus \{a\}) \cup \mathcal{F}(Q \setminus \{a\})$  *Because,  $s' \setminus \{a\} = s$*

$(s, X) \in \mathcal{F}(P \setminus \{a\}) \square \mathcal{F}(Q \setminus \{a\})$  *From the LHS failures definition*

as required

□ *Lemma(2.1)*

**Lemma(2.2).** If  $P \setminus \{a\}$  is non-divergent,  $REMOVE_a(P)$  is defined, and  $P ::= a \rightarrow P \mid c?x!y \rightarrow P(x) \mid P \square Q \mid P \sqcap Q$ , then:

$$REMOVE_a(P) =_{FD} P \setminus \{a\}$$

**Proof** (by induction)

**Case(1)** :  $a \rightarrow P$

$$REMOVE_a(a \rightarrow P) =$$

$$REMOVE_a(P) = \quad \text{From Definition(2)}$$

$$P \setminus \{a\}$$

**Case(2)** :  $a?x!y \rightarrow P(x)$

$$REMOVE_a(a?x!y \rightarrow P(x)) =$$

$$\sqcap_x REMOVE_a(P(x)) = \quad \text{From Definition(2)}$$

$$P(x) \setminus \{a\}$$

**Case(3)** :  $b \rightarrow P$ , where  $b \neq a$

$$REMOVE_a(b \rightarrow P) =$$

$$b \rightarrow REMOVE_a(P) = \quad \text{From Definition(2)}$$

$$b \rightarrow P \setminus \{a\}$$

**Case(4)** :  $b?x!y \rightarrow P(x)$ , where  $b \neq a$

$$REMOVE_a(b?x!y \rightarrow P(x)) =$$

$$b?x!y \rightarrow REMOVE_a(P(x)) = \quad \text{From Definition(2)}$$

$$b?x!y \rightarrow P(x) \setminus \{a\}$$

**Case(5)** :  $P \sqcap Q$

$$REMOVE_a(P \sqcap Q) =$$

$$REMOVE_a(P) \sqcap REMOVE_a(Q) = \quad \text{From Definition(2)}$$

$$P \setminus \{a\} \sqcap Q \setminus \{a\} =$$

$$(P \sqcap Q) \setminus \{a\} \quad \text{From Lemma(8.3.3) of [2]}$$

**Case(6)** :  $P \sqcup Q$ , where  $a \notin \text{initials}(P) \wedge a \notin \text{initials}(Q)$

$$REMOVE_a(P \sqcup Q) =$$

$$REMOVE_a(P) \sqcup REMOVE_a(Q) = \quad \text{From Definition(2)}$$

$$P \setminus \{a\} \sqcup Q \setminus \{a\} =$$

$$(P \square Q) \setminus \{a\} \quad \text{From Lemma(2.1)}$$

□ Lemma(2.2)

**Theorem(2).** *If  $a \in \text{alpha}(P_1)$ ,  $a \notin \text{alpha}(P_2)$ ,  $P_1 \setminus \{a\}$  is non-divergent,  $\text{REMOVE}_a(P_1)$  is defined and  $P_1$  as defined in Lemma(2.2) then:  $\text{REMOVE}_a(P_1) \parallel P_2$  is deadlock free  $\Leftrightarrow P_1 \parallel P_2$  is deadlock free.*

**Proof**

$$\text{REMOVE}_a(P_1) \parallel P_2$$

$$= P_1 \setminus \{a\} \parallel P_2 \quad \text{From Lemma(2.2)}$$

$$= (P_1 \parallel P_2) \setminus \{a\} \quad \text{From Low 6 in Section 3.5.1 of [3]}$$

$$\text{REMOVE}_a(P_1) \parallel P_2 \text{ is deadlock free} \Leftrightarrow P_1 \parallel P_2 \text{ is deadlock free} \quad \text{From Section 13.1.1 of [2]}$$

(as required)

□ Theorem(2)

## 4 Opti-Rule(3): Removing internal choices

Unlike the first two Opti-Rules, this one needs human interaction to be performed. It also does not lead to an optimized version of the original model. Alternatively, it splits the original model into chunks that are easier to be checked separately using FDR2. This kind of Opti-Rules are very useful when analyzing big models, as it will allow the modeller to focus on certain parts of the model (chunk) at a time (a bounded approach to find and solve the model's problems). Another benefit is that when the model is too big to be analyzed by FDR2, this Opti-Rule can be used to analyze the system on different stages, each stage is an analysis of one of the model's chunk.

Opti-Rule(3) can be summarized as follow: When checking deadlock, if all the chunks are deadlock free, then the original model is deadlock free as well. A chunk can be generated by disabling an internal choice in the model and replacing it with one (or more) of the internal choice branches.

To apply this rule, the Optimization Advisor scans the fUML model for the decision nodes that will be translated to internal choices in the CSP model. If the number of the internal choices exceeds a certain limit (threshold), the

Optimization Advisor will advise the modeller to divide the model checking on several stages, each stage includes subset of the internal choices. Consequently, the modeller will choose this subset based on his understanding of the model, and then start the model checking process (first stage). The selected choices will be passed to the Model Optimizer to apply Opti-Rule(3) based on the modeller selection. The process should be repeated to cover all the internal choices in the original model.

Illustrated below Theorem(3), which proofs that the accumulative deadlock checking (on several stages) is equivalent to the original model deadlock checking (as a whole).

**Theorem(3).** *If  $SYS = P_1 \sqcap P_2$ ,  $SYS' = P_1$  (after splitting  $SYS$  and choosing the first branch), and  $SYS'' = P_2$  (after splitting  $SYS$  and choosing the second branch), then:*

$$\neg(SYS \text{ Deadlock Free}) \Leftrightarrow \neg(SYS' \text{ Deadlock Free}) \vee \neg(SYS'' \text{ Deadlock Free})$$

### ***Proof***

Let  $SYS$  is an LTS (Label Transition System) that consists of:

- Set of nodes  $\{P_0, P_1, \dots, P_n\}$ , each represents the current status of the system.
- Binary transitions relations between those nodes  $\{\xrightarrow{x} \mid x \in A^+\}$ , where  $A$  is the set of all possible external events in the system and  $A^+ = A \cup \{\tau\}$ .

Let  $P_0$  is one of  $SYS$  nodes (states) that holds a non-deterministic (internal) choice ( $\sqcap$ ), so that:

$$\begin{array}{l} P_0 \xrightarrow{\tau} P_1 \quad \dots \dots \quad \text{first choice, and} \\ P_0 \xrightarrow{\tau} P_2 \quad \dots \dots \quad \text{second choice.} \end{array}$$

Let  $SYS'$  represents  $SYS$ , but with removing the non-deterministic choice and continuing with the first choice, so that:

$$SYS' = SYS \setminus \{P_0 \xrightarrow{\tau} P_1\}$$

Let  $SYS''$  represents  $SYS$ , but with removing the non-deterministic choice and continuing with the second choice, so that:

$$SYS'' = SYS \setminus \{P_0 \xrightarrow{\tau} P_2\}$$

Let  $SYS$  contains a deadlock state such that:

$$\begin{array}{l} P \xrightarrow{\mu_1} P' \xrightarrow{\mu_2} P'' \xrightarrow{\mu_3} \dots \xrightarrow{\mu_n} P^n \\ \text{Where, } P^n \not\rightarrow \quad \quad \quad (\text{i.e., there are no transitions after } P^n) \end{array}$$

If  $P_0$  is one of the in between (between  $P$  and  $P^n$ ) states and  $P_0$  appears at most once, then the deadlock  $P \xrightarrow{\mu_1 \dots \mu_n} P^n$  exists in  $SYS'$  or  $SYS''$

Which can be formed as follow:

$$\neg(SYS \text{ Deadlock Free}) \Leftrightarrow \neg(SYS' \text{ Deadlock Free}) \vee \neg(SYS'' \text{ Deadlock Free})$$

as required.

□ *Theorem(3)*

## References

1. Abdelhalim, I., Schneider, S.A.: An Optimization Approach for Effective Formalized fUML Model Checking. In: Integrated Formal Methods - 9th International Conference, IFM 2012 - To Submit. Lecture Notes in Computer Science, Springer (2012)
2. Roscoe, A.W.: The Theory and Practice of Concurrency. Prentice Hall (1998)
3. Hoare, C.A.R.: Communicating Sequential Processes. Prentice Hall International (2004)
4. Schneider, S.: Concurrent and Real-Time Systems: the CSP Approach. Wiley (1999)