The United Nations
University

UNU/IIST

International Institute for
Software Technology

An Operational Approach to Communicating Processes

He Jifeng

May, 2003

An Operational Approach to Communicating Processes

He Jifeng

Abstract

Bisimulation and its asymmetric variant (simulation) are widely used in CCS to compare the behaviour of processes. In CSP, correctness issue is addressed by introducing an ordering relation between an implementation and a specification. This report presents a new operational semantics for CSP, where two closure transitions are added to give a calculus in which simulation and refinement are identical.

He Jifeng is a senior research-fellow of UNU/IIST, He is also a professor of computer science at East China Normal University and Shanghai Jiao Tong University. His research interests include the mathematical theory of programming and refined methods, the design techniques for the mixed software/hardware systems. Email: hjf@iist.unu.edu

Contents

1 Introduction

A communicating process is regarded as an agent which may communicate with its environment (which may itself regarded as a process) by performing certain atomic actions drawn from its alphabet. In CCS [5], an equivalent relation known as *bisimilarity* is introduced to compare processes. It is defined co-inductively on the structure of the behavioural graph of the two processes, where the arrows of the graph are the transitions in which the process may engage. To addressing the question of correctness, CSP defines an ordering relation known as *refinement* [2, 7], which holds between an implementation and a specification whenever all observations of any execution of the former are included in those allowed by the latter.

Bisimilarity is based on an operational semantics for the process calculus, it thus appeals directly to the operational intuition of programmers. It admits simple proofs using co-induction. In particular, proofs can often be replaced by mechanical checking. Refinement is based on a denotational semantics, which is close to that used more normally in mathematics. In the case of communicating processes, we can relate their definitions immediately to more or less direct observations of every execution of processes. A specification too is nothing but a description of the observations of the software product which the user will regard as acceptable. To be correct, a process must just be a subset of the observations permitted by the specification. The advantages of refinement are abstraction and expression power. It enables us to deal with program and specification in the same mathematical framework.

The technique of reconciliation of refinement with bisimularity was invented in [3] in dealing with a tiny subset of CCS, where three non-standard transitions are added to the operational semantics of CCS to give a calculus in which strong, weak, and two-thirds simulations correspond exactly to the trace, trace/divergence, and failure/divergence refinements. This report proposes two non-standard transitions to the operational semantics of CSP [1], with the result that two-thirds simulation [4] and failure-divergence refinement [1] are identical. The additional transitions are selected to characterise some algebraic properties of observations made during the execution of a process, thereby having an intuitive content.

We consider a tiny subset of CSP in this report, which contains deadlocked process, guarded choice, non-deterministic choice, general choice, parallel, hiding and recursion.

$$P ::= X \mid STOP \mid y : B \rightarrow P_y \mid P \sqcap Q \mid P \Box Q \mid P \| Q \mid \mu X.P$$

We use the notation $\alpha(P)$ to denote the set of all visible actions the process P may perform during its execution. It is assumed that $\alpha(P)$ is *finite* for every process P.

The operational semantics is defined by an appropriate set of **Conf** of configurations, representing the partial results or stages in an execution, and a set of *transition relations* between configurations, specifying how configurations can alter during execution of a process. Since we are dealing with an event-based model, it is natural to follow Plotkin [6] and use a *labelled transition system*, in which transitions are associated with the occurrence of events.

For the set of configurations we take the *closed terms* in this tiny language, that is, the terms in which no process identifier occur free. Let

$$\textbf{Conf} =_{df} \{P \mid P \text{ is a closed term}\}$$

The operational is defined by a family

$$\{\xrightarrow{x} \mid x \in \mathcal{A}^\tau\}$$

of labelled transition relations over configurations, where $\mathcal{A}$ is an *infinite* set of all externally visible actions, and

$$\mathcal{A}^\tau =_{df} \mathcal{A} \cup \{\tau\}$$

with τ a symbol not in $\mathcal{A}$. In later discussion, we will use a to denote a visible action, and let x to range over the set $\mathcal{A}^\tau$.

Transitions have one of two forms: a transition labelled with an action a,

$$P \xrightarrow{a} Q$$

can be interpreted "P can transform to process Q by performing an a-action"; this will be called a *visible* transition. An *invisible* or *silent* transition has the form

$$P \xrightarrow{\tau} Q$$

and indicates that P may evolve in a manner invisible to, and out with the control of, its environment into the process Q.

Section 2 enriches the standard transition system of CSP [1] by adding two closure transitions. Section 3 discusses the basic properties of the transition system, and revisits the problems of deadlock and livelock. In section 4 we rebuild a failure/divergence model for CSP, and show that the non-standard transitions keep the original denotational definition intact. Section 5 adopts the definition of two-third simulation of Larsen and Skou [4], and shows that it is a pre-congruence. In section 6 we demonstrate that the refinement ordering used to deal with the correctness issue in the standard failure/divergence model [1] corresponds exactly to the two-thirds simulation. Section 7 concludes the report, and proposes another technique to reconcile simulation and refinement.

2 Transition System

This section gives a set of transition axioms and inference rules for each syntactic construct.

2.1 Guarded Choice

Let B be a finite subset of $\mathcal{A}$, and P_y an expression defining a process for each action a in B. Assume that $B \subseteq \alpha(P_a)$ for every action a of B and $\alpha(P_{a1}) = \alpha(P_{a2})$ for all $a1, a2 \in B$, then $y : B \to P_y$ defines a process, which first offers a choice over any action a in B, and then behaves like P_a.

$$\frac{a \in B}{(y : B \to P_y) \xrightarrow{a} P_a}$$

The guarded choice has the same alphabet as its guarded processes

$$\alpha(y : B \to P_y) =_{df} \alpha(P_a), \quad a \in B$$

The notation $STOP$ represents a process which cannot perform any action. It can be seen as just a special case of the guarded choice notation with $B = \emptyset$.

2.2 Nondeterminism

If P and Q are processes of the same alphabet, then the notation

$$P \sqcap Q$$

represents a process which may behave, either like P or like Q, where the selection between them is made arbitrary, without the knowledge or control of the external environment.

$$(P \sqcap Q) \xrightarrow{\tau} P$$
$$(P \sqcap Q) \xrightarrow{\tau} Q$$

As usual

$$\alpha(P \sqcap Q) =_{df} \alpha(P) = \alpha(Q)$$

2.3 General Choice

Let P and Q be processes with $\alpha(P) = \alpha(Q)$. The notation

$$P \Box Q$$

represents a process, which may behave as P or Q and is such that the environment may influence the choice of which provided that such influence is exerted on the first occurrence of an action of the composite process.

$$\frac{P \xrightarrow{\tau} R}{(P \Box Q) \xrightarrow{\tau} (R \Box Q)} \qquad \frac{Q \xrightarrow{\tau} R}{(P \Box Q) \xrightarrow{\tau} (P \Box R)}$$

$$\frac{P \xrightarrow{a} R}{(P \Box Q) \xrightarrow{a} R} \qquad \frac{Q \xrightarrow{a} R}{(P \Box Q) \xrightarrow{a} R}$$

Define

$$\alpha(P \Box Q) \ =_{df} \ \alpha(P) \ = \ \alpha(Q)$$

2.4 Parallel Composition

The notation

$$P \| Q$$

denotes a process which behaves like the parallel system composed of processes P and Q with the restriction that any action performed by the composition must lie in $\alpha(P) \cup \alpha(Q) \cup \{\tau\}$, and the composition may perform an action x only if

(1) either $x \in \alpha(P) \cap \alpha(Q)$ and both P and Q may perform x,

(2) or $x \in \alpha(P)^\tau \setminus \alpha(Q)$ and P may perform x,

(3) or $x \in \alpha(Q)^\tau \setminus \alpha(P)$ and Q may perform x,

$$\frac{P \xrightarrow{a} R, \quad Q \xrightarrow{a} S, \quad a \in \alpha(P) \cap \alpha(Q)}{(P \,\|\, Q) \xrightarrow{a} (R \,\|\, S)}$$

$$\frac{P \xrightarrow{x} R, \quad x \in \alpha(P)^\tau \setminus \alpha(Q)}{(P \,\|\, Q) \xrightarrow{x} (R \,\|\, Q)}$$

$$\frac{Q \xrightarrow{x} S, \quad x \in \alpha(Q)^\tau \setminus \alpha(P)}{(P \,\|\, Q) \xrightarrow{x} (P \,\|\, S)}$$

The alphabet of $P\|Q$ is defined as the union of of the alphabets of its operands

$$\alpha(P\|Q) \ =_{df} \ \alpha(P) \cup \alpha(Q)$$

2.5 Hiding

Let B be a finite set of actions. $P\backslash B$ behaves like P except that all occurrences of the actions of B are rendered invisible to the environment.

$$\frac{P \xrightarrow{x} R, \quad x \notin B}{P\backslash B \xrightarrow{x} R\backslash B}$$

$$\frac{P \xrightarrow{a} R, \quad a \in B}{P\backslash B \xrightarrow{\tau} R\backslash B}$$

We define

$$\alpha(P\backslash B) \ =_{df} \ \alpha(P) \setminus B$$

2.6 Recursion

Let $P(X)$ be an expression defining a process for each process X. The notation

$$\mu X.P(X)$$

denotes a recursive process, which executes its body P.

$$\mu X.P \xrightarrow{\tau} P[\mu X.P/X]$$

We use the notation $CHAOS$ to represent the recursive process $\mu X.X$ hereafter.

2.7 Closure Rules

If P can evolve into both R_1 and R_2 by performing an a-action, then the selection between these two transitions is made internally. This non-deterministic choice can remain open after the a-transition.

$$\frac{P \xrightarrow{a} R_i \;\; \text{for } i = 1, 2}{P \xrightarrow{a} (R_1 \sqcap R_2)} \qquad (closure - 1)$$

This non-standard transition corresponds to the following algebraic law in CSP [2, 7]

$$(a \to Q) \sqcap (a \to Q) \;=\; a \to (P \sqcap Q)$$

If P can make a hidden transition, then the implementation may make it nondeterministically. However, it the process P had previously offered a choice to its environment, it would be kinder to keep this offer open even after the τ transition.

$$\frac{P \xrightarrow{\tau} R}{P \xrightarrow{\tau} (P \Box R)} \qquad (closure - 2)$$

which has a counterpart in the algebraic laws of CSP

$$P \sqcap (P \Box Q \Box R) \;=\; P \sqcap (P \Box Q) \sqcap (P \Box Q \Box R)$$

3 Properties of the Transition System

This section investigates properties of the transition system defined in the previous section, and relates transitions with observations one can make during execution of processes.

3.1 Finitary

Theorem 3.1.1

For any process P, the set of visible actions it can perform at the very beginning, defined formally by

$$\mathcal{H}(P) =_{df} \{a \mid a \in \mathcal{A} \wedge \exists R \bullet P \xrightarrow{a} R\}$$

is finite.

Proof The conclusion follows from the following facts

$$\begin{array}{rcl}
\mathcal{H}(y : B \rightarrow P_y) & = & B \\
\mathcal{H}(STOP) & = & \emptyset \\
\mathcal{H}(P \sqcap Q) & = & \emptyset \\
\mathcal{H}(P \Box Q) & = & \mathcal{H}(P) \cup \mathcal{H}(Q) \\
\mathcal{H}(P \| Q) & = & \mathcal{H}(P) \setminus \alpha(Q) \; \cup \; \mathcal{H}(Q) \setminus \alpha(P) \; \cup \; \mathcal{H}(P) \cap \mathcal{H}(Q) \\
\mathcal{H}(P \backslash B) & = & \mathcal{H}(P) \setminus B \\
\mathcal{H}(\mu X.P) & = & \emptyset
\end{array}$$

Theorem 3.1.2

For any process P and any action a, the set $\{R \mid P \xrightarrow{a} R$ where no closure transition is involved$\}$ is finite,

Proof Let $\mathcal{S}_a(P) =_{df} \{R \mid P \xrightarrow{a} R$ where no closure transition is involved$\}$. From the transition axioms and inference rules of Section 2 we conclude that

$$\begin{array}{rcl}
\mathcal{S}_a(y : B \rightarrow P_y) & = & \{P_a\} \lhd a \in B \rhd \emptyset \\
\mathcal{S}_a(STOP) & = & \emptyset \\
\mathcal{S}_a(P \sqcap Q) & = & \emptyset \\
\mathcal{S}_a(P \Box Q) & = & \mathcal{S}_a(P) \cup \mathcal{S}_a(Q) \\
\mathcal{S}_a(P \| Q) & = & \left\{ \begin{array}{ll} \{R \| S \mid R \in \mathcal{S}_a(P) \wedge S \in \mathcal{S}_a(Q)\} & \text{if } a \in \alpha(P) \cap \alpha(Q) \\ \{R \| Q \mid R \in \mathcal{S}_a(P)\} & \text{if } a \in \alpha(P) \setminus \alpha(Q) \\ \{P \| S \mid S \in \mathcal{S}_a(Q)\} & \text{if } a \in \alpha(Q) \setminus \alpha(P) \end{array} \right. \\
\mathcal{S}_a(P \setminus B) & = & \{R \backslash B \mid R \in \mathcal{S}_a(P)\} \lhd a \notin B \rhd \emptyset \\
\mathcal{S}_a(\mu X.P) & = & \emptyset
\end{array}$$

Theorem 3.1.3

For any process P, the set $\{R \mid P \xrightarrow{\tau} R$ where no closure transition is involved$\}$ is finite,

Proof Let $\mathcal{S}_\tau(P) =_{df} \underline{minimal}_{\sqcap}\{R \mid P \xrightarrow{a} R\}$. The conclusion follows from the fact that

$$\begin{array}{rcl}
\mathcal{S}_\tau(y : B \rightarrow P_y) & = & \emptyset \\
\mathcal{S}_\tau(STOP) & = & \emptyset \\
\mathcal{S}_\tau(P \sqcap Q) & = & \{P, Q\}
\end{array}$$

$$\begin{aligned}
\mathcal{S}_\tau(P\Box Q) &= \{R\Box Q \mid R \in \mathcal{S}_\tau(P)\} \cup \{P\Box S \mid S \in \mathcal{S}_\tau(Q)\}\\
\mathcal{S}_\tau(P\|Q) &= \{R\|Q \mid R \in \mathcal{S}_\tau(P)\} \cup \{P\|S \mid S \in \mathcal{S}_\tau(Q)\}\\
\mathcal{S}_\tau(P \setminus B) &= \{R\backslash B \mid R \in \bigcup_{x\in B^\tau} \mathcal{S}_x(P)\}\\
\mathcal{S}_\tau(\mu X.P) &= \{P[\mu X.P/X]\}
\end{aligned}$$

3.2 Divergence

The symbol τ is used to stand for an event that is internal to the process, in the sense that its occurrence cannot be influenced by the external environment. Its introduction brings up the question of divergence, also known as livelock. Divergence can occur when there is an infinite series of τ transitions which a process can make.

Definition 3.2.1

A process P *can diverge* if it is infinitely often capable of τ transition in absence of the transition $(closure-2)$.

$$P \xrightarrow{\tau} P_0 \xrightarrow{\tau} ... \xrightarrow{\tau} P_n \xrightarrow{\tau} ...$$

Theorem 3.2.1

If P can diverge, then it is infinitely often capable of τ transition in absence of the transitions $(closure-1)$ and $(closure-2)$.

Proof Because P can diverge, one has

$$P \xrightarrow{\tau} P_0 \xrightarrow{\tau} ... \xrightarrow{\tau} P_n \xrightarrow{\tau} ...$$

where the rule $(closure-2)$ is not involved in the transitions. We are going to find an infinite set $\{X_i \mid i \in Nat\}$ of processes such that

$$P \xrightarrow{\tau} X_0 \xrightarrow{\tau} ... \xrightarrow{\tau} X_n \xrightarrow{\tau} ...$$

where neither $(closure-1)$ nor $(closure-2)$ is used.

Case 1. $P = Q \sqcap R$: In this case the first transition

$$P \xrightarrow{\tau} P_0$$

ensures that either $P_0 = Q$ or $P_0 = R$. Consequently, we can set $X_0 = P_0$.

Case 2. $P = \mu X.F$: In this case the first transition $P \xrightarrow{\tau} P_0$ has to be $\mu X \xrightarrow{\tau} F[\mu X.F/X]$. Like Case 1, we can set $X_0 = P_0$.

Case 3. $P = Q\backslash B$: There can be two cases:

(1a) $P_0 = R\backslash B$ and $R \in \underline{after}_b(Q)^{\sqcap}$ with $b \in B$. where

$$\underline{after}_b(Q)^{\sqcap} = \bigcup_{n \in Nat} PP^n$$

and $PP^0 =_{df} \underline{after}_b(Q)$ and $PP^{n+1} =_{df} PP^n \cup \{R \sqcap S \mid R, S \in PP^n\}$.

If $R \in PP^0 = \underline{after}_b(Q)$ then the transition

$$P = Q\backslash B \xrightarrow{\tau} P_0 = R\backslash B$$

does not use $(closure - 1)$, and we can set $X_0 = P_0$.

If $R \in PP^1 \setminus PP^0$ then there exist U and W in $\underline{after}_b(Q)$ such that $R = U \sqcap W$. As a result the transition

$$P_0 = (U \sqcap W)\backslash B \xrightarrow{\tau} P_1$$

implies that either $P_1 = U\backslash B$ or $P_1 = W\backslash B$, and we can simply set $X_0 = P_1$.

In general if $R \in PP^{n+1} \setminus PP^n$, then the existence of the transitions

$$P_0 = R\backslash B \xrightarrow{\tau} ... \xrightarrow{\tau} P_{n+1}$$

ensures that there are processes $R_1 \in PP^n \setminus PP^{n-1}$,.., $R_{n+1} \in PP^0$ such that

$$P_i = R_i\backslash B \ \text{ for } 1 \le i \le n+1$$

Consequently we have

$$P = Q\backslash B \xrightarrow{\tau} P_{n+1} = R_{n+1}\backslash B$$

without use of $(closure - 2)$, and can set $X_0 = P_{n+1}$

(1b) $P_0 = R\backslash B$ and $Q \xrightarrow{\tau} R$.

If the transition $Q \xrightarrow{\tau} R$ does not employ $(closure - 1)$, we can set $X_0 = P_0$.

If $Q = U\backslash C$, taking a similar argument as in (1a) we can derive

$$P = Q\backslash B \xrightarrow{\tau} P_k$$

in absence of the closure rules, and set $X_0 = P_k$.

Case 4. $P = Q\Box R$. From the transition rules of $\Box$ we can show that either Q can diverge, or R can diverge. The conclusion follows from the induction hypothesis.

Case 5. $P = Q\|R$: similar to Case 4.

Clearly P cannot be a guarded choice nor the deadlocked process $STOP$. By induction we can follow the above procedure to construct a sequence $\{X_i \mid i \in Nat\}$ of processes from $\{P_i \mid i \in Nat\}$ as required. $\Box$

Lemma 3.2.2

Assume that Q is the unique process such that $P \xrightarrow{\tau} Q$, then if P can diverge, so can Q.

Corollary

If $\mu X.P$ can diverge, so can the process $P[\mu X.P/X]$.

Theorem 3.2.3

(1) $y : B \rightarrow P_y$ cannot diverge.

(2) $STOP$ cannot diverge.

(3) $(P\Box Q)$ can diverge iff either P or Q can diverge.

(4) $(P \sqcap Q)$ can diverge iff either P or Q can diverge.

(5) $(P \parallel Q)$ can diverge iff either P or Q can diverge.

(6) $CHAOS$ can diverge.

(7) $P\backslash B$ can diverge iff either P can diverge or it can perform the actions of B indefinitely, i.e., there exists a set $\{P_i \mid i \in Nat\}$ such that

$$\exists x \in B^\tau (P \xrightarrow{x} P_0) \ \wedge \ \forall i \in Nat \exists x \in B^\tau \bullet (P_i \xrightarrow{x} P_{i+1})$$

hold in absence of the closure rules.

Proof (1) and (2) follow from the fact that $\mathcal{S}_\tau(y : B \rightarrow P_y) = \mathcal{S}_\tau(STOP) = \emptyset$

(3) $(P \Box Q)$ can diverge

$\Leftrightarrow$ {Definition 3.1}

$\exists \{R_k \mid k \in Nat\} \bullet (P \Box Q) \xrightarrow{\tau} R_0 \wedge \forall i \in Nat \bullet R_i \xrightarrow{\tau} R_{i+1}$

$\Leftrightarrow$ $\{\tau -$ transition rules for general choice$\}$

$\exists \{R_k \mid k \in Nat\}, \exists \{P_k \mid k \in Nat\}, \exists \{Q_k \mid k \in Nat\} \bullet \forall i \in Nat \bullet (R_i = P_i \Box Q_i) \wedge$

$\exists N_1, N_2 \bullet N_1 \cup N_2 = Nat \wedge N_1 \cap N_2 = \emptyset \wedge$

$(P \xrightarrow{\tau} P_0 \wedge Q = Q_0) \vee (P = P_0 \wedge Q \xrightarrow{\tau} Q_0) \wedge$

$\forall i \in N_1 \bullet (P_i \xrightarrow{\tau} P_{i+1} \wedge Q_i = Q_{i+1}) \wedge \forall i \in N_2 \bullet (P_i = P_{i+1} \wedge Q_i \xrightarrow{\tau} Q_{i+1})$

$\Leftrightarrow$ {either N_1 or N_2 is infinite}

either P or Q can diverge

(4) The conclusion follows the fact that $\mathcal{S}_\tau(P \sqcap Q) = \{P, Q\}$.

(5) Similar to the proof of (3).

(6) From the transition rule for recursion we obtain $CHAOS \xrightarrow{\tau} CHAOS$.

(7) From the transition rules for the hiding operator and Theorem 3.2.1 it follows that there are $\{P_i \mid i \in Nat\}$ such that of the closure rules

$$(P \backslash B \xrightarrow{\tau} P_0 \backslash B) \text{ and } \forall i \in Nat \bullet (P_i \backslash B \xrightarrow{\tau} P_{i+1} \backslash B)$$

in absence of the closure rules. The conclusion follows directly from the τ-rule for the hiding operator. □

3.3 Transitive Closure

This section will define a new kind of transition

$$P \overset{a}{\Longrightarrow} Q$$

to indicate that P can make zero or more τ-transitions before perform a visible event a. If P can diverge, our definition allows it to make transition $P \overset{a}{\Longrightarrow}$ for any a whatsoever. This choice is motivated by correctness concerns in CSP, where livelock is regarded as the worst behaviour of a process [1].

Definition 3.3.1

$P \xrightarrow{*} P' =_{df} P$ can diverge or $P(\xrightarrow{\tau})^* P'$

$P \overset{a}{\Longrightarrow} P' =_{df} \exists R \bullet P \xrightarrow{*} R \xrightarrow{a} P'$ □

Theorem 3.3.1

If $P \overset{a}{\Longrightarrow} S_i$ for $i = 1$, 2, then

$$P \overset{a}{\Longrightarrow} (S_1 \sqcap S_2)$$

Proof If P can diverge then the conclusion follows directly from the definition of the binary relation $\overset{a}{\Longrightarrow}$. If P cannot diverge, we are going to investigate the following cases:

Case 1. $P \xrightarrow{a} S_i$ for $i = 1, 2$: the conclusion follows from $(closure - 1)$.

Case 2. $P \xrightarrow{\tau} R \overset{a}{\Longrightarrow} S_1$: in this case following $(closure - 2)$ we have

$$P \xrightarrow{\tau} (P \Box R)$$

The conclusion follows from the τ rule for general choice and the rule $(closure - 1)$.

Case 3. $P \xrightarrow{\tau} R \overset{a}{\Longrightarrow} S_2$: the proof is similar to Case 2. □

Theorem 3.3.2

If P cannot diverge, then for any action $a \in \mathcal{A}$ the set

$$\underline{after}_a(P) =_{df} \{Q \mid P \overset{a}{\Longrightarrow} Q \text{ where no closure rule is involved}\}$$

is finite.

Proof From Theorems 3.1.2 and 3.1.3. □

Corollary

If P cannot diverge, then $\{R \mid P \overset{a}{\Longrightarrow} R\}$ is the least fixed point of the recursive equation

$$\mathcal{PP} \ = \ \underline{after}_a(P) \cup \{Q \sqcap R \mid Q, R \in \mathcal{PP}\}$$

Proof From the closure rules (1) and (2). □

Theorem 3.3.3

(1) $\underline{after}_a(P \sqcap Q) \ = \ \underline{after}_a(P) \cup \underline{after}_a(Q)$

(2) $\underline{after}_a(P \Box Q) \ = \ \underline{after}_a(P) \cup \underline{after}_a(Q)$

(3) $\underline{after}_a(P \| Q) \ = \ \{U \| W \mid U \in \underline{after}_a(P) \wedge W \in \underline{after}_a(Q)\}$ provided that $a \in \alpha(P) \cap \alpha(Q)$

(4) $\underline{after}_a(P \| Q) \ = \ \{U \| W \mid U \in \underline{after}_a(P) \wedge Q(\xrightarrow{\tau})^* W\}$ if $a \in \alpha(P) \setminus \alpha(Q)$

(5) $\underline{after}_a(P \| Q) \ = \ \{U \| W \mid P(\xrightarrow{\tau})^* U \ \wedge W \in \underline{after}_a(Q)\}$ provided that $a \in \alpha(Q) \cap \alpha(P)$ □

Lemma 3.3.4

If $P \xrightarrow{x} R$, then $x \in \alpha(P)^\tau$ and $\alpha(P) = \alpha(R)$.

Theorem 3.3.5

If $P \overset{a}{\Longrightarrow}$ for all a, then P can diverge, and vice-versa.

Proof Assume that P cannot diverge and $P \overset{a}{\Longrightarrow}$ for all a. Let $b \in \mathcal{A} \setminus \alpha(P)$.

$$\begin{array}{ll} & P \overset{b}{\Longrightarrow} R \\ \Leftrightarrow & \{P \text{ cannot diverge}\} \\ & \exists \{Q_i \mid 1 \leq i \leq n\} \bullet P \xrightarrow{\tau} Q_1 \xrightarrow{\tau} ... \xrightarrow{\tau} Q_n \xrightarrow{b} R \\ \Rightarrow & \{\text{Lemma 3.3.4}\} \\ & b \in \alpha(P) \end{array}$$

which directly contradicts the assumption $b \notin \alpha(P)$. □

Corollary

If $\{a : \mathcal{A} \mid P \overset{a}{\Longrightarrow}\}$ is infinite, then P can diverge, and vice-versa. □

To deal with the problem of deadlock, CSP introduces an explicit observation of the fact a process can on occasion refuse to engage in some event, even if the environment is ready to select that event. Deadlock is not possible while a process is still capable of further internal transition. A process will actually refuse an event only when it has reached a *stable* state.

Definition 3.3.2

A process is *stable* if it cannot perform τ-transition.

Theorem 3.3.6

(1) $(y : B \rightarrow P_y)$ is stable.

(2) $P \Box Q$ is stable iff both P and Q are stable.

(3) $P \sqcap Q$ is NOT stable.

(4) $P \| Q$ is stable iff both P and Q are stable.

(5) $P \backslash B$ is stable iff P is stable and cannot perform any action of B.

(6) $\mu X.P$ is NOT stable.

Proof (1) follows from the fact that $y : B \rightarrow P_y$ can only perform the actions of B at the very beginning.

(2) From the τ transition rule for general choice it follows that $P \Box Q$ can make a τ-move if and only if either P or Q can perform a τ transition.

(3) From the fact that $P \sqcap Q$ resolves its choice by performing a τ transition at the very beginning.

(4) Similar to the proof of (2).

(5) From the τ rule for hiding.

(6) From the τ rule for recursion.

Definition 3.3.3

Let $a \in \alpha(P)$. P refuses action a if it is stable, but cannot perform a.

Let $X \subseteq \alpha(P)$. P refuses X if it refuses every action of X.

Theorem 3.3.7

(1) $y : B \to P_y$ refuses every set X with $X \cap B = \emptyset$

(2) $P \Box Q$ refuses X iff both P and Q refuse X.

(3) $P \parallel Q$ refuses X iff there exist X_1 and X_2 such that $X = X_1 \cup X_2$ and P refuses X_1 and Q refuses X_2.

(4) $P \backslash B$ refuses X iff P refuses $B \cup X$.

Proof The conclusion (1) follows from Theorem 3.3.6(1) and the fact that the guarded choice $y : B \to P_y$ can only perform the actions of B at the very beginning.

(2) From Theorem 3.3.6(2) and the fact that $P \Box Q$ can perform action a iff either P or Q can do it.

(3) From Theorem 3.3.6(4) it follows that $P \| Q$ is stable iff both P and Q are stable. The transition rules for $\|$ lead to the following observations

(i) $P \| Q$ cannot perform an action of $\alpha(P) \setminus \alpha(Q)$ iff P can not do it.

(ii) $P \| Q$ can not perform an action of $\alpha(P) \cap \alpha(Q)$ iff either P or Q can not engage in it.

(iii) $P \| Q$ cannot perform an action of $\alpha(Q) \setminus \alpha(P)$ iff Q cannot do it.

from which follows the conclusion.

(4) The conclusion follows Theorem 3.3.4(5) and the visible transition rules for hiding. $\Box$

Definition 3.3.4

P can refuse $X =_{df} \exists R \bullet (P \xrightarrow{*} R$ and R refuses $X)$.

Theorem 3.3.8

If P can refuse X, then it remains so after removal of the closure rules $(closure - 1)$ and $(closure - 2)$.

Proof We consider the case where P cannot diverge. Define

$$\underline{after}_{\tau}(P) =_{df} \{Q \mid P(\xrightarrow{\tau})^* Q \text{ where no closure rule is used}\}$$

From the closure rules it follows that $\{R \mid P \xrightarrow{*} R\}$ is the least fixed point of the recursive equation

$$\mathcal{PP} = \underline{after}_{\tau}(P) \cup \{U \Box W \mid U, W \in \mathcal{PP}\}$$

which in together with Theorem 3.3.7(2) implies the conclusion. $\Box$

Theorem 3.3.9

(1) P can refuse $\emptyset$

(2) If P can refuse X, then it can also refuse every subset of X.

(3) If P can refuse X, and for any $a \in Y$ there is no process R such that $P \overset{a}{\Longrightarrow} R$, then P can also refuse $X \cup Y$.

Proof of (3). Case 1: P can diverge. The conclusion follows from the fact that $P \overset{*}{\Longrightarrow} STOP$.

Case 2: P cannot diverge. From the assumption it follows that there exists a process S which refuses X, and

$$P \overset{*}{\Longrightarrow} S$$

Clearly, S cannot perform any action of Y, otherwise it leads to a contradiction with the assumption that for any $a \in Y$ there is no process R such that $P \overset{a}{\Longrightarrow} R$, Consequently, S cannot execute any action of $X \cup Y$. □

Theorem 3.3.10

$P \sqcap Q$ can refuse X iff either P or Q can refuse X.

Proof Case 1. $P \sqcap Q$ can diverge: the conclusion follows from Theorem 3.2.3(3).

Case 2. $P \sqcap Q$ cannot diverge. From Theorem 3.3.8 it follows that $P \sqcap Q$ can also refuse X in absence of the closure rules. The conclusion follows from the fact that $P \sqcap Q$ can only evolve to either P or Q as its first transition. □

4 Observations

Definition 4.1

A trace of a process P is a sequence of events, which is a possible record of all the actions occurring in the execution of P up to some point in time.

$\mathcal{T}(P) \ =_{df} \ \{<>\} \ \cup \ \{< a > s \mid \exists R \bullet P \overset{a}{\Longrightarrow} R \ \wedge \ s \in \mathcal{T}(R)\}$

A divergence is also represented by a sequence of actions whose occurrences can lead P to livelock.

$\mathcal{D}(P) \ =_{df} \ \{<> \ \mid P \text{ can diverge}\} \cup \{< a > s \mid \exists R \bullet P \overset{a}{\Longrightarrow} R \wedge s \in \mathcal{D}(R)\}$

A failure of a process P is defined as a pair (s, X), where s is a trace and X a set that can be refused by P after it has engaged in all of the sequence of events of s.

$\mathcal{F}(P) \ =_{df} \ \{(<>, X) \mid P \text{ can refuse } X\} \ \cup \ \{(< a > s, X) \mid \exists R \bullet P \overset{a}{\Longrightarrow} R \ \wedge \ (s, X) \in \mathcal{F}(R)\}$□

Theorem 4.1

The presence of the closure rules $(closure-1)$ and $(closure-2)$ has no effect on the construction of $\mathcal{D}(P)$, $\mathcal{T}(P)$ and $\mathcal{F}(P)$.

Proof From Theorems 3.2.1 and 3.3.8, and Corollary of Theorem 3.3.2. □

Theorem 4.2

$\mathcal{T}(P) \ = \ \{s \mid (s, \emptyset) \in \mathcal{F}(P)\}$

Proof By induction on the length of s.

$$\begin{array}{lll}
\text{Base case :} & & <>\in \mathcal{T}(P) \\
& \Leftrightarrow & \{\text{Theorem 3.3.7}\} \\
& & (<>, \emptyset) \in \mathcal{F}(P) \\
\text{Inductive step :} & & < a > s \in \mathcal{T}(P) \\
& \Leftrightarrow & \{\text{Definition 4.1}\} \\
& & \exists R \bullet P \overset{a}{\Longrightarrow} R \wedge s \in \mathcal{T}(R) \\
& \Leftrightarrow & \{\text{induction hypothesis}\} \\
& & \exists R \bullet P \overset{a}{\Longrightarrow} R \wedge (s, \emptyset) \in \mathcal{F}(R) \\
& \Leftrightarrow & \{\text{Definition 4.1}\} \\
& & (< a > s, \emptyset) \in \mathcal{F}(P)
\end{array}$$

Theorem 4.3

If P cannot diverge, and $P \overset{a}{\Longrightarrow} R$, then $\mathcal{F}(R) \subseteq \bigcup_{Q \in \underline{after}_a(P)} \mathcal{F}(Q)$.

$$\begin{array}{lll}
\textbf{Proof} & & (s, X) \in \mathcal{F}(R) \\
& \Rightarrow & \{\text{Definition of } \mathcal{F}(P)\} \\
& & < a > s \in \mathcal{F}(P) \\
& \Rightarrow & \{\text{Theorem 4.1}\} \\
& & \exists Q \bullet (Q \in \underline{after}_a(P) \wedge (s, X) \in \mathcal{F}(Q))
\end{array}$$

Theorem 4.4

If $s \in \mathcal{D}(P)$, then $s < a >\in \mathcal{D}(P)$ for all a.

Proof From Theorem 3.3.5. □

Lemma 4.5

$\mathcal{D}(P)$ is a subset of $\mathcal{T}(P)$.

$$\begin{array}{lll}
\textbf{Proof}\text{. Base case} & & <>\in \mathcal{D}(P) \\
& \Rightarrow & \{\text{Definition of } \mathcal{T}(P)\} \\
& & <>\in \mathcal{T}(P)
\end{array}$$

Inductive step $< a > s \in \mathcal{D}(P)$

$\Rightarrow$ {Definition of $\mathcal{D}(P)$}

$\exists a \bullet P \overset{a}{\Longrightarrow} R \wedge s \in \mathcal{D}(R)$

$\Rightarrow$ {induction hypothesis}

$\exists a \bullet P \overset{a}{\Longrightarrow} R \wedge s \in \mathcal{T}(R)$

$\Rightarrow$ {Definition of $\mathcal{T}$}

$< a > s \in \mathcal{T}(P)$

Lemma 4.6

If $s < a > \in \mathcal{T}(P)$ for all action a of an infinite set, then $s \in \mathcal{D}(P)$.

Proof From Corollary of Theorem 3.3.5. □

Theorem 4.7

$\mathcal{D}(P) = \{s \mid s \in \mathcal{T}(P) \wedge \forall a \in \mathcal{A} \bullet s \cdot < a > \in \mathcal{T}(P)\}$

Proof. Base case : $<> \in \mathcal{D}(P)$

$\Leftrightarrow$ {Definition 4.1}

P can diverge

$\Leftrightarrow$ {Theorem 3.3.5}

$\forall a \in \mathcal{A} \bullet (P \overset{a}{\Longrightarrow} R)$

$\Leftrightarrow$ {Definition 4.1}

$\forall a \in \mathcal{A} \bullet < a > \in \mathcal{T}(P)$

Inductive step : $< b > s \in \mathcal{D}(P)$

$\Leftrightarrow$ {Definition 4.1}

$\exists R \bullet P \overset{b}{\Longrightarrow} R \wedge s \in \mathcal{D}(R)$

$\Leftrightarrow$ {induction hypothesis}

$\exists R \bullet P \overset{b}{\Longrightarrow} R \wedge s \in \mathcal{T}(R) \wedge \forall a \in \mathcal{A} \bullet s < a > \in \mathcal{T}(R)$

$\Rightarrow$ {Definition 4.1}

$\forall a \in \mathcal{A} \bullet < b > s < a > \in \mathcal{T}(P)$

$\Rightarrow$ {Theorem 3.3.2}

$\exists R, \exists X \bullet X$ is infinite $\wedge\ P \overset{b}{\Longrightarrow} R \wedge \forall a \in X \bullet s < a > \in \mathcal{T}(R)$

$\Rightarrow$ {Lemma 4.6}

$\exists R \bullet P \overset{b}{\Longrightarrow} R \wedge s \in \mathcal{D}(R)$

$\Rightarrow$ {Definition of $\mathcal{D}(P)$}

$< b > s \in \mathcal{D}(P)$

5 Simulation

The definition of simulation we will use is called two-third simulation, introduced by Larson and Skou. It deals with both traces and refusals.

Definition 5.1

A simulation is defined as a binary relation $\mathcal{R}$ between processes such that

(1) If $P\,\mathcal{R}\,Q$ and $P \overset{a}{\Longrightarrow} U$, then there exists a process W such that

$$Q \overset{a}{\Longrightarrow} W \text{ and } U\,\mathcal{R}\,W$$

(2) if $P\,\mathcal{R}\,Q$ and P can refuse X, so can Q. $\square$

We use $\leq$ to denote the largest simulation relation.

Theorem 5.1

(1) $\leq$ is transitive.

(2) If $P \xrightarrow{\tau} Q$, then $Q \leq P$.

Proof of (2). Let Id denote the identity relation. Define $\mathcal{R} \;=_{df}\; Id \cup \{(Q,\, P)\}$.

From the fact that $Q \overset{a}{\Longrightarrow} S$ implies $P \overset{a}{\Longrightarrow} S$ we conclude that $\mathcal{R}$ is a simulation.

Theorem 5.2

If $P \;\leq\; Q$ then

(1) $(a \rightarrow P) \;\leq\; (a \rightarrow Q)$

(2) $(P \Box R) \leq (Q \Box R)$

(3) $(P \sqcap R) \leq (Q \sqcap R)$

(4) $(P \| R) \leq (Q \| R)$

(5) $(P \backslash B) \leq (Q \backslash B)$

(6) $\mu X.P \;\leq \mu X.Q$

Proof

(1) It is easy to show $\mathcal{R} =_{df} \leq \cup \{(a \rightarrow P,\, a \rightarrow Q)\}$ is a simulation.

(2) From the fact that $\mathcal{R} =_{df} \leq\; \cup \{(P \Box W,\, Q \Box W) \mid P \leq Q\} \cup \{P \sqcap W,\, Q \sqcap W) \mid P \leq Q\}$ is a simulation.

(3) Similar to (2) we can show $\mathcal{R} =_{df} \leq \cup \{(P \sqcap W,\, Q \sqcap W) \mid P \leq Q\}$ is a simulation.

(4) From the fact that

$$\mathcal{R} =_{df} \; \{(P \| W,\, Q \| W) \mid P \leq Q\} \cup \{(X_1 \sqcap X_2,\, Y_1 \sqcap Y_2) \mid (X_1,\, Y_1),\, (X_2,\, Y_2) \in \mathcal{R}\}$$

is a simulation.

(5) From the fact that $\mathcal{R} =_{df} \{(P\backslash B, Q\backslash B) \mid P \leq Q\} \cup \{(X_1 \sqcap X_2, Y_1 \sqcap Y_2) \mid (X_1, Y_1), (X_2, Y_2) \in \mathcal{R}\}$ is a simulation.

(6) Define $\mathcal{R} =_{df} \leq \cup \{(\mu X.P, \mu X.Q) \mid P \leq Q\}$ which can be shown a simulation. □

Theorem 5.3

(1) $P \leq (P \sqcap P)$

(2) $(P \sqcap P) \leq P$

(3) $P \sqcap Q \leq Q \sqcap P$

(4) $P \sqcap (Q \sqcap R) \leq (P \sqcap Q) \sqcap R$

(5) $(P \sqcap Q) \sqcap R \leq P \sqcap (Q \sqcap R)$

Proof (1) Let $\mathcal{R} =_{df} Id \cup \{(P, P \sqcap P)\}$. From Theorem 3.3.3(1) and Corollary of Theorem 3.3.2 one has

(i) $\{R \mid P \overset{a}{\Longrightarrow} R\}$ is the least fixed point of the recursive equation

$$PP = \underline{after}_a(P) \cup \{X \sqcap Y \mid X, Y \in PP\}$$

(ii) $\underline{after}_a(P \sqcap P) = \underline{after}_a(P) \cup \underline{after}_a(P) = \underline{after}_a(P)$

which imply that $\mathcal{R}$ is a simulation.

(2) Similar to (1).

(3) Define $\mathcal{R} =_{df} Id \cup \{(P \sqcap Q, Q \sqcap P)\}$ which can be shown to be a simulation from Theorem 3.3.3(1) and Corollary of Theorem 3.3.2.

(4) Define $\mathcal{R} =_{df} Id \cup \{P \sqcap (Q \sqcap R), (P \sqcap Q) \sqcap R\}$ which can be shown to be a simulation from Corollary of Theorem 3.3.2 and the fact that

$$\underline{after}_a(P \sqcap (Q \sqcap R)) = \underline{after}_a(P) \cup \underline{after}_a(Q) \cup \underline{after}_a(R) = \underline{after}_a((P \sqcap Q) \sqcap R)$$

(5) Similar to (2). □

Definition 5.2

Define $P \sim Q =_{df} (P \leq Q) \wedge (Q \leq P)$.

Clearly, $\sim$ is an equivalence relation. □

Theorem 5.4

(1) $P \sim P \sqcap P$

(2) $P \sqcap Q \sim Q \sqcap P$

(3) $P \sqcap (Q \sqcap R) \sim (P \sqcap Q) \sqcap R$

Proof From Theorem 5.3. □

The previous theorem enables us to define a non-deterministic choice over a finite set of processes

$$\sqcap\{Q_1, .., Q_n\} \;=_{df}\; Q_1 \sqcap Q_2 \sqcap ... \sqcap Q_n$$

6 Failure/Divergence Refinement

A process P can be seen as a correct implementation of a process Q if all observations of the behaviour of every execution of P are allowed by Q. This paper proposes a variant of the failure-divergence model [1] for CSP, identifies each process with the set of failures one can record during its execution.

Definition 6.1

P refines Q if $\mathcal{F}(P) \subseteq \mathcal{F}(Q)$.

Theorem 6.1

If P refines Q, then $\mathcal{D}(P) \subseteq \mathcal{D}(Q)$.

Proof From Theorem 4.7. □

Theorem 6.2

If $P \leq Q$, then P refines Q.

Proof Induction on the length of trace s.

$$\begin{array}{lll}
\text{Base case :} & & (<>, X) \in \mathcal{F}(P) \\
 & \Rightarrow & \{\text{Definition 4.1}\} \\
 & & P \text{ can refuse } X \\
 & \Rightarrow & \{P \leq Q\} \\
 & & Q \text{ can refuse } X \\
 & \Rightarrow & \{\text{Definition 4.1}\} \\
 & & (<>, X) \in \mathcal{F}(Q)
\end{array}$$

$$\begin{array}{lll}
\text{Inductive step :} & & (<a>s,\, X) \in \mathcal{F}(P) \\
& \Rightarrow & \{\text{Definition 4.1}\} \\
& & \exists R \bullet P \overset{a}{\Longrightarrow} R \wedge (s,\, X) \in \mathcal{F}(R) \\
& \Rightarrow & \{P \leq Q\} \\
& & \exists S \bullet Q \overset{a}{\Longrightarrow} S \wedge R \leq S \wedge (s,\, X) \in \mathcal{F}(R) \\
& \Rightarrow & \{\text{induction hypothesis}\} \\
& & \exists S \bullet Q \overset{a}{\Longrightarrow} S \wedge (s,\, X) \in \mathcal{F}(S) \\
& \Rightarrow & \{\text{Definition 4.1}\} \\
& & (<a>s,\, X) \in \mathcal{F}(Q)
\end{array}$$

Definition 6.2 (Derivative)

Let P be a process. Assume that there exists R such that $P \overset{a}{\Longrightarrow} R$, we define the a-derivative of P by

$$P/a \;=_{df}\; \begin{cases} \sqcap \underline{after}_a(P) & \text{if } P \text{ cannot diverge} \\ CHAOS & \text{if } P \text{ can diverge} \end{cases} \qquad \Box$$

Theorem 6.3

(1) $P \overset{a}{\Longrightarrow} (P/a)$

(2) $\mathcal{F}(P/a) \;=\; \{(s,\, X) \mid (<a>\cdot s,\, X) \in \mathcal{F}(P)\}$

Proof (1) The conclusion follows from Theorem 3.3.1.

(2) If P can diverge, the conclusion is trivial. Let us examine the case when P cannot diverge.

$$\begin{array}{ll}
& \mathcal{F}(P) \\
= & \{\text{Definition 4.1}\} \\
& \{(<>,\, X) \mid P \text{ can refuse } X\} \;\cup\; \{(<a>s,\, X) \mid \exists Q \bullet P \overset{a}{\Longrightarrow} Q \;\wedge\; (s,\, X) \in \mathcal{F}(Q)\} \\
= & \{\text{Theorems 4.3}\} \\
& \{(<>,\, X) \mid P \text{ can refuse } X\} \;\cup \\
& \{(<a>s,\, X) \mid \exists Q \in \underline{after}_a(P) \bullet P \overset{a}{\Longrightarrow} Q \;\wedge\; (s,\, X) \in \mathcal{F}(Q)\} \\
= & \{\mathcal{F}(P \sqcap Q) \;=\; \mathcal{F}(P) \cup \mathcal{F}(Q)\} \\
& \{(<>,\, X) \mid P \text{ can refuse } X\} \;\cup\; \{(<a>s,\, X) \mid (s,\, X) \in \mathcal{F}(P/a)\}
\end{array}$$

which leads to the conclusion (2).

Corollary If $P \overset{a}{\Longrightarrow} R$ then R refines (P/a).

Proof From Theorem 4.3, and the fact that $\mathcal{F}(P \sqcap Q) = \mathcal{F}(P) \cup \mathcal{F}(Q)$. $\Box$

Theorem 6.4

If P refines Q, and $P \overset{a}{\Longrightarrow} R$, then

(1) there exists S such that $Q \overset{a}{\Longrightarrow} S$

(2) P/a refines Q/a.

Proof The conclusion (1) follows from the fact that $(< a >, \emptyset) \in \mathcal{F}(P) \subseteq \mathcal{F}(Q)$.

$$\begin{array}{lll}
(2) & & (s, X) \in \mathcal{F}(P/a) \\
& \Rightarrow & \{\text{Theorem 6.3(2)}\} \\
& & (< a > \cdot s, X) \in \mathcal{F}(P) \\
& \Rightarrow & \{P \text{ refines } Q\} \\
& & (< a > \cdot s, X) \in \mathcal{F}(Q) \\
& \Rightarrow & \{\text{Theorem 6.3(2)}\} \\
& & (s, X) \in \mathcal{F}(Q/a)
\end{array}$$

Theorem 6.5 (Refinement implies simulation)

If P refines Q, then $P \leq Q$.

Proof Define $\mathcal{R} \ =_{df} \ \{(P, Q) \mid \mathcal{F}(P) \subseteq \mathcal{F}(Q)\}$. We are going to show that $\mathcal{R}$ meets the conditions of a simulation.

(1) Assume that $P \overset{a}{\Longrightarrow} S$. From Corollary of Theorem 6.3 and Theorem 6.4 one concludes

$$\mathcal{F}(S) \subseteq \mathcal{F}(P/a) \subseteq \mathcal{F}(Q/a)$$

which implies that $S \,\mathcal{R}\, (Q/a)$.

(2) If P can refuse X, then $(<>, X) \in \mathcal{F}(P) \subseteq \mathcal{F}(Q)$ which indicates Q can also refuse X. $\Box$

7 Conclusion

This paper illustrates how to reconcile simulation and refinement by adding two non-standard transitions to the transition system of CSP [1]. The technique adopts the definition of two-thirds simulation of Larsen and Skou. Another way to achieve the reconciliation is to leave the transition system of CSP intact, and to revise the definition of the two-thirds simulation by replace its first condition

(1) If $P \,\mathcal{R}\, Q$ and $P \overset{a}{\Longrightarrow} U$, then there exists a process W such that

$$Q \overset{a}{\Longrightarrow} W \text{ and } U \,\mathcal{R}\, W$$

by

(1') If $P \,\mathcal{R}\, Q$ and $P \overset{a}{\Longrightarrow} U$, then there exist processes $W_1, .., W_n$ such that $Q \overset{a}{\Longrightarrow} W_i$ for all i, and

$$U\,\mathcal{R}\,(W_1 \sqcap (W_2 \sqcap ..(W_{n-1} \sqcap W_n))$$

Consequently we can establish the following facts:

(1) The revised two-third simulation implies refinement.

(2) The failure/divergence refinement is a simulation.

References

[1] S.D. Brookes, C.A.R. Hoare and A.W. Roscoe. *A theory of communicating sequential processes.* Journal of the ACM, Vol 31: 560-599, (1984).

[2] C.A.R. Hoare. *Communicating Sequential Processes* Prentice Hall, (1985)

[3] C.A.R. Hoare, et al. *Bisimulation and refinement reconciled.* (2003).

[4] K.G. Larsen and A. Skou. *Bisimulation through Probabilistic Testing.* Information and Control 94, 1 (1991).

[5] R. Milner. *Communication and Concurrency.* Prentice Hall, (1989).

[6] G.D. Plotkin. *A structural approach to operational semantics* Technical Report, DAIMI-FN-19, Aarhus University, Denmark, (1981).

[7] A,W, Roscoe. *The theory and practice of concurrency.* Prentice Hall, (1998).