Input Output Conformance Testing in the Unifying Theories of Programming

SNA-TR-2008-1P6

Martin Weiglhofer¹  Bernhard Aichernig²
Input Output Conformance Testing in the Unifying Theories of Programming

Martin Weiglhofer\textsuperscript{1} \hspace{1cm} Bernhard Aichernig\textsuperscript{2}

Abstract. Model-based conformance testing aims to assess the correctness of an implementation with respect to a specification. This raises the question of a proper conformance relation that should be established between implementations and specifications. One commonly used conformance relation is the so-called input output conformance (ioco), which is defined over labeled transition systems. In this paper we investigate a denotational semantics of the input output conformance relation over reactive processes. We formalize the underlying assumptions of the ioco relation in terms of formal healthiness conditions and by adopted choice operators. Finally, we show that our denotational version of ioco can be generalized in the same way as the original relation. Our work aims to provide a unification of input output conformance by lifting the definition from labeled transition systems to reactive processes.

\textsuperscript{1}Competence Network Softnet Austria, Institute for Softwatechnology, Graz University of Technology, Inffeldgasse 16c, A-8010 Graz, Austria. E-mail: weiglhofer@ist.tugraz.at

\textsuperscript{2}Institute for Softwatechnology, Graz University of Technology, Inffeldgasse 16b/II, A-8010 Graz, Austria. E-mail: aichernig@ist.tugraz.at

Copyright © 2009 by the authors
# Contents

1 Introduction 4

2 Input Output Conformance of Labeled Transition Systems 4

3 Input Output Conformance of Processes 8
   3.1 Reactive Processes 8
   3.2 IOCO Specifications 9
   3.3 IOCO Implementations 46
   3.4 Predicative Input Output Conformance Relation 60
   3.5 Test Cases, Test Processes, and Test Suites 62
   3.6 Testing Implementations 83

4 Conclusion and Future Work 84
List of Figures

1 Examples of input output labeled transition systems. ............................... 5
2 Examples of suspension automata. ..................................................... 6
3 Examples of input output transition systems (input-enabled by definition). .... 7
1 Introduction

Software development is a complex and error-prone task. Failures in safety-critical applications may be life-threatening. At least software failures cause high costs during and after the software development process. Therefore, software engineers need the support of tools, techniques, and theories in order to reduce the number of software failures.

Model-based black-box testing techniques aim to assess the correctness of a reactive system, i.e., the implementation under test (IUT), with respect to a given specification. The IUT is viewed as a black-box with an interface that accepts inputs and produces outputs. The goal of model-based black-box testing is to check if the observable behavior of the IUT conforms to a specification with respect to a particular conformance relation.

Industrial specifications are mostly incomplete and due to abstraction non-deterministic. Hence, a conformance relation being useful in industry needs to cope with incompleteness and non-determinism. One of the most popular of such conformance relations is the input output conformance ($ioco$) relation [Tre96].

Mature research prototypes (e.g. TORX [TB03], TGV [JJ05]) and successful industrial case studies (e.g. [dVBF02, APWW07]) have shown the usability of this conformance relation in practice. However, the used theory is given in an operational semantics and some of the underlying assumptions have been stated informally only. It is the contribution of this paper to redefine $ioco$ in the denotational predicative semantics of UTP. The benefits of this new theory can be summarized as follows: (1) Instead of describing the assumptions of $ioco$ informally, the UTP formalization presents the underlying assumptions as unambiguous healthiness conditions and by adopted choice operators over reactive processes; (2) A UTP formalization naturally relates $ioco$ and refinement in one theory; (3) The denotational version of $ioco$ enables formal, machine checkable, proofs. (4) Due to the predicative semantics of UTP, test case generation based on the presented theory can be seen as a satisfiability problem. This facilitates the use of modern sat modulo theory techniques (e.g. [DdM08]) for test case generation. (5) Finally, the UTP version of $ioco$ broadens the scope of $ioco$ to specification languages with similar UTP semantics, e.g. to generate test cases from Circus [OCW07] specifications. Hence our work enriches UTP’s reactive processes with a practical testing theory.

The rest of this paper is structured as follows. Section 2 reviews the input output conformance relation. Section 3 comprises the formalization of $ioco$ in the UTP-framework. Finally, we discuss our results and further research in Section 4.

2 Input Output Conformance of Labeled Transition Systems

This section reviews the $ioco$ relation [Tre96] which is defined over labeled transition system (LTS). When testing reactive systems one distinguishes between inputs and outputs. Thus, the alphabet of an LTS is partitioned into inputs and outputs.

Definition 1 (Labeled transition system with inputs and outputs) A labeled transition system is a tuple $M = (Q, A \cup \{\tau\}, \rightarrow, q_0)$, where $Q$ is a finite set of states, $A = A_I \cup A_O$ a finite alphabet partitioned into an input alphabet $A_I$ and an output alphabet $A_O$ where $A_I \cap A_O = \emptyset$. $\tau \notin A$ an unobservable action, $\rightarrow \subseteq Q \times (A \cup \{\tau\}) \times Q$ is the transition relation, and $q_0 \in Q$ is the initial state.

The class of labeled transition systems with inputs $A_I$ and outputs in $A_O$ is denoted by $\mathcal{LTS}(A_I, A_O)$ [Tre96]. We use the following common notations for LTSs:

Definition 2 Given a labeled transition system $M = (Q, A_I \cup A_O \cup \{\tau\}, \rightarrow, q_0)$ and let $q, q', q_i \in Q, a_i(\cdot) \in \mathbb{N}$, $a_i(\cdot)$
A_1 \cup A_O and \sigma \in (A_1 \cup A_O)^*.

\begin{align*}
q \xrightarrow{a} q' &=_{d_f} (q, a, q') \in \rightarrow \\
q \xrightarrow{a} &=_{d_f} \exists q' \bullet (q, a, q') \in \rightarrow \\
q \xrightarrow{\tau} &=_{d_f} \not\exists q' \bullet (q, a, q') \in \rightarrow \\
q \xrightarrow{?} q' &=_{d_f} (q = q') \lor \exists q_0, \ldots, q_n \bullet (q = q_0 \xrightarrow{\tau} q_1 \land \cdots \land q_n \xrightarrow{\tau} q_n = q') \\
q \xrightarrow{a} q' &=_{d_f} \exists q_1, q_2 \bullet q \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{q} q' \\
q \xrightarrow{a} \Rightarrow q' &=_{d_f} \exists q_0, \ldots, q_n \bullet q = q_0 \xrightarrow{a} q_1 \cdots q_n \xrightarrow{a} q_n = q' \\
q \xrightarrow{?} &=_{d_f} \exists q' \bullet q \xrightarrow{?} q'
\end{align*}

According to [Tre96], we use init(q) to denote the actions enabled in state q and traces(q) to denote the traces enabled in state q. Furthermore, we denote the states reachable by a particular trace \sigma by q after \sigma. More precisely,

**Definition 3** Given a labeled transition system \( M = (Q, A_I \cup A_O \cup \{\tau\}, \rightarrow, q_0) \) and let \( q \in Q, C \subseteq Q \) and \( \sigma \in (A_I \cup A_O)^* \).

\begin{align*}
\text{init}(q) &=_{d_f} \{ a \in A_I \cup A_O \cup \{\tau\} \mid q \xrightarrow{a} \} \\
q \text{ after } \sigma &=_{d_f} \{ q' \mid q \xrightarrow{\sigma} q' \} \\
\text{traces}(q) &=_{d_f} \{ \sigma \mid q \xrightarrow{\sigma} \} \\
C \text{ after } \sigma &=_{d_f} \bigcup_{q' \in C} \{ q' \text{ after } \sigma \}
\end{align*}

Note that we will not always distinguish between an LTS and its initial state and write \( M \Rightarrow \) instead of \( q_0 \Rightarrow \).

**Example 1** Figure 1 shows four labeled transition systems \( o, p, q, \) and \( r \). The input alphabet is given by \( A_I = \{1, 2\} \) and the output alphabet is \( A_O = \{c, t\} \). We denote input actions by the prefix "?", while output actions have the prefix "!". For example, \( p_0 \text{ after } ?1 = \{p_1\} \) while \( q_0 \text{ after } ?1 = \{q_1, q_2\} \).

The ioco conformance relation employs the idea of observable quiescence. That is, it is assumed that a special action, i.e. \( \theta \), is enabled in the case where the labeled transition system does not provide any output action. This \( \theta \)-labeled transitions allow to detect implementations that do not provide outputs while the specification requires some output (see Example 4: \( \neg(y \text{ ioco } s) \)). The input output conformance relation identifies quiescent states as follows: A state \( q \) of a labeled transition system is quiescent if neither an output action nor an internal action (\( \tau \)) is enabled in \( q \).

**Definition 4** Let \( M \) be a labeled transition system \( M = (Q, A_O \cup A_I \cup \{\tau\}, \rightarrow, q_0) \), then a state \( q \in Q \) is quiescent, denoted by \( \theta(q) \), if \( \forall a \in A_O \cup \{\tau\} \bullet q \xrightarrow{a} \).
Figure 2: Examples of suspension automata.

Usually, \( \delta \) is used as special action denoting quiescence. Because of a name clash with UTP's deadlock symbol \( \delta \) \cite{HH98} we use \( \theta \) for representing quiescence. By adding \( \theta \)-labeled transitions to LTSs the quiescence symbol can be used as any other action. By the use of suspension automata \( \theta \) becomes observable.

**Definition 5 (Suspension automata)** Let \( M = (Q, A_I \cup A_O \cup \{ \tau \}, \rightarrow, q_0) \) then the suspension automaton \( M_{\theta} \) is given by \( (Q, A_I \cup A_O \cup \{ \tau, \theta \}, \rightarrow \cup \theta, \rightarrow_{\theta}, q_0) \) where \( \rightarrow_{\theta} = \{(q, \theta) \mid q \in Q \land \theta(q)\} \).

The suspension traces of \( M_{\theta} \) are \( \text{Straces}(M_{\theta}) = \{ \sigma \in (A_I \cup A_O \cup \{ \theta \})^* \mid q_0 \xrightarrow{\sigma} q \} \).

Unless otherwise indicated, we use from now on \( M_{\theta} \) instead of \( M \), i.e. we usually include \( \theta \) in the transition relations.

**Example 2** Fig. 2 shows the suspension automata for the LTSs illustrated in Fig. 1. The transition systems depicted in Fig. 2 are equipped with \( \theta \) labels for each quiescent state. For example, the states \( u_0, u_2, u_3, \) and \( u_5 \) are quiescent states since they do not have outgoing edges labeled with an output nor with a \( \tau \) action. Among others, \( u \) comprises the suspension traces \( \langle ?1, !c, \theta \rangle \) and \( \langle \theta, ?1, \theta, ?1, !t, \theta \rangle \).

A major hypothesis of the input output conformance relation is that the implementation can be represented as a labeled transition system. It is not assumed that this LTS is known in advance, but only its existence is required. This is known as a testing hypothesis \cite{Ber91, Tre92}.

The models used for representing implementations are input output transition systems. Since implementations are not allowed to refuse inputs, their models obey to the same restriction. This means that implementations are assumed to be input-enabled and so are their models.

**Definition 6 (Input output transition system)** An input output transition system is an LTS \( M = (Q, A_I \cup A_O \cup \{ \tau \}, \rightarrow, q_0) \) where all input actions are enabled (possibly preceded by \( \tau \)-transitions) in all states: \( \forall a \in A_I, \forall q \in Q \cdot q \xrightarrow{a} \).

The class of input output transition systems with inputs \( A_I \) and outputs in \( A_O \) is given by \( \text{IOTS}(A_I, A_O) \subseteq \text{LTS}(A_I, A_O) \) \cite{Tre96}.

**Example 3** The IOTSs for the suspension automata of Fig. 2 are depicted in Fig. 3. Note that the reason for the \( \tau \) transitions in state \( z_4 \) is not input-enabledness but the restrictions on choices (see Section 3.3).

Before giving the definition of the \( \text{ioco} \) relation we need to define what are the outputs of a particular state and what are the outputs of a set of states.
Example 4  Consider the LTSs of Figure 2 to be specifications and let the IOTSs of Figure 3 be implementations. Then we have

Example 5  The IOTS $y$ and the IOTS $z$ of Fig. 3 serve to illustrate the differences between these conformance relations: $z \leq \text{ioco}_y$, $\neg(z \leq \text{ioco}_y)$, $z \text{ conf}_y$, $\neg(z \text{ conf}_y)$, $x \text{ conf}_w$, $x \text{ conf}_w$, $\neg(x \leq \text{ioco}_w)$ and $\neg(x \leq \text{ioco}_w)$. 

Figure 3: Examples of input output transition systems (input-enabled by definition).
By the use of a particular set of test cases one wants to test if a given implementation conforms to its specification. In the \textit{ioco} framework a test case is again a labeled transition system [Tre96]:

**Definition 10 (Test case)** A test case \( t \) is a labeled transition system \( t = (Q, A_I \cup A_O \cup \{\theta\}, \rightarrow, q_0) \) such that

- \( t \) is deterministic and has finite behavior
- \( Q \) contains terminal states pass and fail
- for any state \( q \in Q \) where \( q \neq \text{pass} \) and \( q \neq \text{fail} \), either \( \text{init}(q) = \{a\} \) for some \( a \in A_I \), or \( \text{init}(q) = A_O \cup \{\theta\} \)

Test cases are extracted from specifications by some algorithm (e.g. [Tre96, JJ05]). Basically, test cases consist of some trace of the specification where at each state allowed outputs lead to pass verdict states, while forbidden outputs lead to fail verdict states. Testing is then conducted by running a test case \( t \) in parallel with the implementation \( i \). A test run is a trace of the synchronous parallel composition \( t || i \) leading to a terminal state of \( t \).

3 Input Output Conformance of Processes

This section presents our denotational version of the \textit{ioco} conformance relation. We formulate \textit{ioco} over UTP’s reactive processes. As \textit{ioco}, the denotational version is applicable to incomplete specifications. That is, implementations may behave arbitrarily after unspecified inputs.

3.1 Reactive Processes

Basically, the process of testing is modelled as an interaction between two reactive processes, the implementation under test (IUT) and the test case. A reactive process with respect to the unified theories of programming is defined as follows:

**Definition 11 (Reactive process)** A reactive process \( P \) is one which satisfies the healthiness conditions \( R1, R2, R3 \) where

\[
R1(X) =_{df} X \land (tr \leq tr')
\]

\[
R2(X(tr, tr')) =_{df} \bigcap_s X(s, \hat{s} (tr' - tr))
\]

\[
R3(X) =_{df} I \ll wait \gg X
\]

The alphabet of \( P \) consists of the following:

- \( A \), the set of events in which it can potentially engage.
- \( tr : A^* \), the sequence of events which have happened up to the time of observation.
- \( ref : \mathcal{P}A \), the set of events refused by the process during its wait.
- \( wait : \text{Bool} \), which distinguishes its waiting states from its terminated states.
- \( ok, ok' : \text{Bool} \), indicating start and termination of a process.
The skip predicate (\(I\)) is defined as in [HH98]:
\[ I = \neg ok \land (tr \leq tr') \lor ok' \land (tr' = tr) \land \cdots \land (wait' = wait). \]
The input output conformance relation distinguishes between inputs and outputs. Outputs are actions that are initiated by and under control of an implementation under test, while input actions are initiated by and under control of the system's environment\(^1\). Hence, the alphabet \(A\) of a process consists of two disjoint sets \(A = A_{in} \cup A_{out}\). In addition, we will also differentiate between refused inputs \(ref_{in} = \neg ref \cap A_{in}\) and refused outputs \(ref_{out} = \neg ref \cap A_{out}\). Thus, also refusals form a partition: \(ref_{in} \cap ref_{out} = \emptyset\) and \(ref = ref_{in} \cup ref_{out}\). Note that we use ? and ! to indicate inputs and outputs for processes. For example, a process having as input alphabet \(A_{in} = \{1\}\) and as output alphabet \(A_{out} = \{c\}\) is written as \(do_A(?1); do_A(!c)\).

A process offering a single event \(a \in A\) for communication is expressed in terms of \(do_A(a)\), where
\[
\Phi(a \notin ref' \triangleleft wait' \triangleright tr' = tr \triangleright \langle a \rangle) =_d \Phi =_d R \circ and_B = and_B \circ R, \quad B =_d (tr' = tr) \land wait' \lor (tr < tr')
\]
\[
R =_d R1 \circ R2 \circ R3
\]
For sequential composition we rely on UTP’s standard sequential composition operator: \(P(v,v'); Q(v,v') =_d \exists v_0 \cdot (P(v,v_0) \land Q(v_0,v'))\).

### 3.2 IOCO Specifications
For technical reasons, that is the computability of particular sets during the test case generation, the reactive processes used in the ioco framework need to satisfy an additional healthiness condition. The processes need to be strongly responsive\(^2\), i.e. processes do not comprise livelocks. If there is a livelock a process may execute while it never offers communication. Hence, the healthiness condition for specifications excludes livelocks:

\[
\text{IOCO1: } P = P \land (ok \Rightarrow (wait' \lor ok'))
\]
As a function, \(\text{IOCO1}\) is defined as \(\text{IOCO1}(P) = P \land (ok \Rightarrow (wait' \lor ok'))\). It is an idempotent.

**Lemma 1 (IOCO1-idempotent)**

\[
\text{IOCO1} \circ \text{IOCO1} = \text{IOCO1}
\]

**Proof.**
\[
\begin{align*}
\text{IOCO1} & \circ \text{IOCO1}(P) = \\
& = \text{IOCO1}(P) \land (ok \Rightarrow (wait' \lor ok')) \quad \text{(def. of IOCO1)} \\
& = P \land (ok \Rightarrow (wait' \lor ok')) \land (ok \Rightarrow (wait' \lor ok')) \quad \text{(def. of IOCO1)} \\
& = P \land (ok \Rightarrow (wait' \lor ok')) \quad \text{(prop. calculus)} \\
& = P \land (ok \Rightarrow (wait' \lor ok')) \quad \text{(def. of IOCO1)} \\
& = \text{IOCO1}(P)
\end{align*}
\]
\(\exists\) is \(\text{IOCO1}\) healthy.

**Lemma 2 (I-IOCO1-healthy)**

\[
\text{IOCO1}(\exists) = \exists
\]
Proof.

\[ IOCO1(I) = \]
\[ = (¬ok ∧ (tr ≤ tr') ∨ ok' ∧ ...) ∧ (ok ⇒ (wait' ∨ ok')) \]
\[ = (¬ok ∧ (tr ≤ tr') ∧ (¬ok ∨ (wait' ∨ ok'))) ∨ (ok' ∧ ... ∧ (¬ok ∨ wait' ∨ ok')) \]
\[ = (¬ok ∧ (tr ≤ tr') ∧ ¬ok) ∨ (¬ok ∧ (tr ≤ tr') ∧ wait') ∨ (¬ok ∧ (tr ≤ tr') ∧ ok') \]
\[ = (¬ok ∧ (tr ≤ tr')) ∨ (ok ∧ ...) \]
\[ = I \]

The healthiness condition \( IOCO1 \) is independent from \( R1, R2, \) and \( R3 \), i.e. they commute.

Lemma 3 (commutativity-IOCO1-R1)

\[ IOCO1 ∘ R1 = R1 ∘ IOCO1 \]

Proof.

\[ IOCO1(R1(P)) = \]
\[ = IOCO1(P ∧ (tr ≤ tr')) \]
\[ = P ∧ (tr ≤ tr') ∧ (ok ⇒ (wait' ∨ ok')) \]
\[ = P ∧ (ok ⇒ (wait' ∨ ok')) ∧ (tr ≤ tr') \]
\[ = IOCO1(P) ∧ (tr ≤ tr') \]
\[ = R1(IOCO1(P)) \]

Lemma 4 (commutativity-IOCO1-R2)

\[ IOCO1 ∘ R2 = R2 ∘ IOCO1 \]

Proof.

\[ IOCO1(R2(P(tr, tr'))) = \]
\[ = IOCO1(P(⟨⟩, tr' − tr)) \]
\[ = P(⟨⟩, tr' − tr) ∧ (ok ⇒ (wait' ∨ ok')) \]
\[ = P ∧ (ok ⇒ (wait' ∨ ok'))(⟨⟩, tr' − tr) \]
\[ = IOCO1(P)(⟨⟩, tr' − tr) \]
\[ = R2(IOCO1(P)) \]

Lemma 5 (commutativity-IOCO1-R3)

\[ IOCO1 ∘ R3 = R3 ∘ IOCO1 \]

Proof.

\[ IOCO1(R3(P)) = \]
\[ = IOCO1(⟨⟩ ∼ wait ⊢ P) \]
\[ = (⟨⟩ ∼ wait ⊢ P) ∧ (ok ⇒ (wait' ∨ ok')) \]
\[ = (⟨⟩ ∧ (ok ⇒ (wait' ∨ ok'))) ∼ wait ⊢ (P ∧ (ok ⇒ (wait' ∨ ok')))) \]
\[ = IOCO1(⟨⟩ ∼ wait ⊢ IOCO1(P)) \]
\[ = I ∼ wait ⊢ IOCO1(P) \]
\[ = R3(IOCO1(P)) \]
Furthermore, composing two \textit{IOCO1} healthy processes using sequential composition, gives us another \textit{IOCO1} healthy process.

\textbf{Lemma 6 (closure-;\textit{-IOCO1})}

\[ \text{IOCO1}(P; Q) = P; Q \text{ provided } P \text{ and } Q \text{ are } \text{IOCO1} \text{ and } R3 \text{ healthy} \]
Proof.

$\text{IOCO1}(P; Q) =$ (assumption)

$= \text{IOCO1}(\text{IOCO1}(P); \text{IOCO1}(R3(Q)))$ (def. of R3)

$= \text{IOCO1}(\text{IOCO1}(P); \text{IOCO1}(\{ \langle wait \triangleright Q \rangle \}))$ (def. of ;)

$= \text{IOCO1}(\exists v_0 \bullet \text{IOCO1}(P)[[v_0/v']] \land \text{IOCO1}(\{ \langle wait \triangleright Q \rangle[v_0/v] \})$ (def. of \text{IOCO1})

$= \text{IOCO1}(\exists v_0 \bullet (P \land (ok \Rightarrow (wait' \lor ok')))[[v_0/v']] \land$

$((\{ \langle wait \triangleright Q \rangle \land (ok \Rightarrow (wait' \lor ok'))[[v_0/v]})$ (def. of if)

$= \text{IOCO1}(\exists v_0 \bullet (P \land (ok \Rightarrow (wait' \lor ok')))[[v_0/v']] \land$

$((\{ \langle wait \land \lnot wait \land Q \rangle \land (ok \Rightarrow (wait' \lor ok')))[[v_0/v]})$ (prop. calculus)

$= \text{IOCO1}(\exists v_0 \bullet (P \land (ok \Rightarrow (wait' \lor ok')))[[v_0/v']] \land$

$((\{ \langle wait \land \lnot wait \land Q \rangle \land (ok \Rightarrow (wait' \lor ok')))[[v_0/v]})$ (prop. calculus)

$= \exists v_0 \bullet (P[v_0/v'] \land I[v_0/v] \land wait_0 \land (ok \Rightarrow (wait_0 \lor ok_0)) \land (ok_0 \Rightarrow (wait' \lor ok')) \land (ok \Rightarrow (wait' \lor ok'))\lor$

$(P[v_0/v'] \land Q[v_0/v] \land \lnot wait_0 \land (ok \Rightarrow (wait_0 \lor ok_0)) \land (ok_0 \Rightarrow (wait' \lor ok')) \land (ok \Rightarrow (wait' \lor ok')))$ (prop. calculus)

$= \exists v_0 \bullet (P[v_0/v'] \land I[v_0/v] \land wait_0 \land (ok \Rightarrow (wait_0 \lor ok_0)) \land (ok_0 \Rightarrow (wait' \lor ok')) \land (ok \Rightarrow (wait' \lor ok'))\lor$

$(P[v_0/v'] \land Q[v_0/v] \land \lnot wait_0 \land (ok \Rightarrow (wait_0 \lor ok_0)) \land (ok_0 \Rightarrow (wait' \lor ok')))$ (prop. calculus)

$= \exists v_0 \bullet (P[v_0/v'] \land I[v_0/v] \land wait_0 \land (ok \Rightarrow (wait_0 \lor ok_0)) \land (ok_0 \Rightarrow (wait' \lor ok')) \land (ok \Rightarrow (wait' \lor ok'))\lor$

$(P[v_0/v'] \land Q[v_0/v] \land \lnot wait_0 \land (ok \Rightarrow (wait_0 \lor ok_0)) \land (ok_0 \Rightarrow (wait' \lor ok')))$ (def. of [])

$= \exists v_0 \bullet ((P \land (ok \Rightarrow (wait' \lor ok')))[v_0/v'] \land \{ \langle wait \land \lnot wait \land Q \rangle \land (Q \land (ok \Rightarrow (wait' \lor ok')))[[v_0/v] \land \lnot wait_0}) \lor$

$(P \land (ok \Rightarrow (wait' \lor ok')))[[v_0/v'] \land Q[v_0/v] \land \lnot wait_0)$ (def. \text{IOCO1})

$= \exists v_0 \bullet (\text{IOCO1}(P)[[v_0/v'] \land \text{IOCO1}(\{ I[v_0/v] \land wait_0 \})}$

$\land wait_0)$ (assumption)

$= \exists v_0 \bullet (P[v_0/v'] \land (\text{IOCO1}(\{ I[v_0/v] \land wait_0 \})) [v_0/v] \land wait_0)$ (Lemma 2)

$= \exists v_0 \bullet (P[v_0/v'] \land \{ I[v_0/v] \land wait_0 \}) \lor$

$(P[v_0/v'] \land Q[v_0/v] \land \lnot wait_0)$ (Lemma 2)

$= \exists v_0 \bullet (P[v_0/v'] \land \{ I[v_0/v] \land wait_0 \}) \lor$

$(P[v_0/v'] \land Q[v_0/v] \land \lnot wait_0)$ (prop. calculus)

$= \exists v_0 \bullet P[v_0/v'] \land ((\{ I[v_0/v] \land wait_0 \} \lor (Q[v_0/v] \land \lnot wait_0))$ (def. of if)

$= \exists v_0 \bullet P[v_0/v'] \land (\{ I[v_0/v] \land \langle wait_0 \triangleright Q[v_0/v] \})$ (def. of [])

$= \exists v_0 \bullet P[v_0/v'] \land \{ I \langle wait \triangleright Q \rangle[v_0/v] \}$ (assumption)

$= \exists v_0 \bullet P[v_0/v'] \land Q[v_0/v]$ (def. of ;)

$= P; Q \blacksquare$
Within Tretmans theory [Tre96] quiescence denotes the absence of outputs and the absence of internal actions. Quiescence is encoded by the presence of a particular action \( \theta \). Although, quiescence can be classified by \( \text{wait}' \) and \( \text{ref}' \) it is necessary to include \( \theta \) into the traces of processes (see Example 4 \( \sim \text{ ioco } u \)).

Since \text{ioco} uses traces containing quiescence we need to include \( \theta \) in the traces of our processes. Thus, we extend set of events for reactive processes \( \mathcal{A} \) by \( \theta \). In the sequel we use the following abbreviation \( \mathcal{A}_\theta = \mathcal{A} \cup \{ \theta \} \). A UTP process is quiescent after a particular trace iff either it has finished its execution or it refuses to do any output action

\[
\text{quiescence} = \neg \text{wait}' \lor \forall o \in A_{\text{out}} \rightarrow o \in \text{ref}'
\]

Quiescent communication is expressed in terms of \( do^\theta_A \), which adds quiescence (\( \theta \)) to the traces and to the refusal set of a process.

**Definition 12 (Quiescent communication)** Let \( a \in \mathcal{A} \) be an action of a process’ alphabet, then

\[
do^\theta_A(a) =\begin{cases} 
\Phi^i(do_A(a)) & \text{if } a \in A_{\text{out}} \\
\Phi^i(\{\theta,a\} \notin \text{ref}' \land tr' - tr \in \theta^* \land \text{wait}') \\
tr' - tr \in \theta^* (a) & \text{if } a \in A_{\text{in}} 
\end{cases}
\]

where \( \Phi^i = \text{ioco1} \circ R \circ \text{and}_B \)

Consider the case where \( a \) is an input action, i.e., \( a \in A_{\text{in}} \): In the case of \( \text{wait}' \) the process \( do^\theta_A(a) \) allows an arbitrary number of \( \theta \) events (see the \( \theta \)-loops in Fig. 2). After termination the event \( a \) has happened preceded by an arbitrary - possible empty sequence - of quiescence events, i.e. \( tr' - tr \in \theta^*(a) \). For the sake of simplicity we sometimes write \( \theta^* \) instead of \( \{\theta\}^* \). Note that \( \theta^*(a) \) denotes the set of events where every element of \( \theta^* \) is concatenated with \( (a) \).

The possible occurrence of \( \theta \) events is formalized as follows:

**IOCO2**\[ P = P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref}' \Rightarrow \text{quiescence}))) \]

**IOCO3**\[ P = P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref}' \Rightarrow \exists s \cdot tr' - tr = s \theta^*))) \]

The antecedence \( \neg \text{wait} \) is necessary due to the same reasons as in \text{R3}.

As functions, **IOCO2** and **IOCO3** can be defined as **IOCO2**\( (P) = P \land (\text{wait}' \Rightarrow (\theta \notin \text{ref}' \Leftrightarrow \text{quiescence})) \) and **IOCO3**\( (P) = P \land (\text{wait}' \Rightarrow (\theta \notin \text{ref}' \Rightarrow \exists s \cdot tr' - tr \in s \theta^*)) \), respectively. Both functions, **IOCO2** and **IOCO3** are idempotent.

**Lemma 7 (IOCO2-idempotent)**

\[
\text{IOCO2 \circ IOCO2} = \text{IOCO2}
\]

**Proof.**

\[
\text{IOCO2( IOCO2(P)) = } \\
= \text{IOCO2}(P) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref}' \Rightarrow \text{quiescence}))) \] (def. of IOCO2)

\[
= P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref}' \Rightarrow \text{quiescence}))) \land \\
(\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref}' \Rightarrow \text{quiescence}))) \] (prop. calculus)

\[
= P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref}' \Rightarrow \text{quiescence}))) \] (def. of IOCO2)

\[
= \text{IOCO2}(P) \]

\[
\square
\]

**Lemma 8 (IOCO3-idempotent)**

\[
\text{IOCO3 \circ IOCO3} = \text{IOCO3}
\]
Proof.

\[ \text{IOCO3}(\text{IOCO3}(P)) = \] (def. of IOCO3)
\[ = \text{IOCO3}(P) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref'} \Rightarrow \exists s \cdot \text{tr'} - \text{tr} = s \hat{\theta}'))) \] (def. of IOCO3)
\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref'} \Rightarrow \exists s \cdot \text{tr'} - \text{tr} = s \hat{\theta}'))) \land \] (prop. calculus)
\[ (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref'} \Rightarrow \exists s \cdot \text{tr'} - \text{tr} = s \hat{\theta}'))) \] (prop. calculus)
\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref'} \Rightarrow \exists s \cdot \text{tr'} - \text{tr} = s \hat{\theta}'))) \land \] (prop. calculus)
\[ = \text{IOCO3}(P) \] (def. of IOCO3)
\[ \Box \]

**Lemma 9** (commutativity-IOCO2-IOCO3)

\text{IOCO2} \circ \text{IOCO3} = \text{IOCO3} \circ \text{IOCO2}

Proof.

\[ \text{IOCO2}(\text{IOCO3}(P)) = \] (def. of IOCO3)
\[ = \text{IOCO2}(P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref'} \Rightarrow \exists s \cdot \text{tr'} - \text{tr} = s \hat{\theta}'))) \] (def. of IOCO2)
\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref'} \Rightarrow \exists s \cdot \text{tr'} - \text{tr} = s \hat{\theta}'))) \land \] (prop. calculus)
\[ (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref'} \Rightarrow \exists s \cdot \text{tr'} - \text{tr} = s \hat{\theta}'))) \] (prop. calculus)
\[ = \text{IOCO2}(P) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref'} \Rightarrow \exists s \cdot \text{tr'} - \text{tr} = s \hat{\theta}'))) \land (\text{tr} \leq \text{tr}') \] (def. of IOCO2)
\[ = \text{R1}(\text{IOCO2}(P)) \] (def. of R1)
\[ \Box \]

In addition, both IOCO2 and IOCO3 are independent from R1, R2 and IOCO1.

**Lemma 10** (commutativity-IOCO2-R1) \text{IOCO2} \circ R1 = R1 \circ \text{IOCO2}

Proof.

\[ \text{IOCO2}(\text{R1}(P)) = \] (def. of R1)
\[ = \text{IOCO2}(P \land (\text{tr} \leq \text{tr}')) \] (def. of IOCO2)
\[ = P \land (\text{tr} \leq \text{tr}') \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref'} \Rightarrow \text{quiescence}))), \text{tr} \leq \text{tr'} \] (prop. calculus)
\[ = \text{IOCO2}(P) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref'} \Rightarrow \text{quiescence}))), \text{tr} \leq \text{tr'} \] (def. of IOCO2)
\[ = \text{R1}(\text{IOCO2}(P)) \] (def. of R1)
\[ \Box \]

**Lemma 11** (commutativity-IOCO2-R2) \text{IOCO2} \circ R2 = R2 \circ \text{IOCO2}

Proof.

\[ \text{IOCO2}(\text{R2}(\text{P}(\text{tr}, \text{tr}')))) = \] (def. of R2)
\[ = \text{IOCO2}(P \land (\text{tr} \leq \text{tr}')) \] (def. of IOCO2)
\[ = P \land (\text{tr} \leq \text{tr'}) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref'} \Rightarrow \text{quiescence}))), \text{tr}, \text{tr}' \text{ are not used in IOCO2} \] (prop. calculus)
\[ = (P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref'} \Rightarrow \text{quiescence}))))((\text{tr}, \text{tr}'), \text{tr} \leq \text{tr'}) \] (def. of IOCO2)
\[ = \text{IOCO2}(P)((\text{tr}, \text{tr}'), \text{tr} \leq \text{tr'}) \] (def. of R2)
\[ = \text{R2}(\text{IOCO2}(P)) \] (def. of R2)
\[ \Box \]
Lemma 12 (commutativity-IOCO2-IOCO1) \( \text{IOCO}_2 \circ \text{IOCO}_1 = \text{IOCO}_1 \circ \text{IOCO}_2 \)

Proof.

\[
\text{IOCO}_2(\text{IOCO}_1(P)) = \\
= \text{IOCO}_2(P \land (\text{ok} \Rightarrow (\text{wait'} \lor \text{ok}'))) \\
= P \land (\text{ok} \Rightarrow (\text{wait'} \lor \text{ok}')) \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (\theta \not\in \text{ref'} \Rightarrow \text{quiescence}))) \land (\text{ok} \Rightarrow (\text{wait'} \lor \text{ok}')) \\
= \text{IOCO}_2(P) \land (\text{ok} \Rightarrow (\text{wait'} \lor \text{ok}'))
\]

(\text{IOCO}_1\text{I}(\text{IOCO}_2(P)))

Lemma 13 (commutativity-IOCO3-R1) \( \text{IOCO}_3 \circ \text{R}_1 = \text{R}_1 \circ \text{IOCO}_3 \)

Proof.

\[
\text{IOCO}_3(\text{R}_1(P)) = \\
= \text{IOCO}_3(P \land (\text{tr} \leq \text{tr'})) \\
= P \land (\text{tr} \leq \text{tr'}) \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (\theta \not\in \text{ref'} \Rightarrow \exists s \bullet \text{tr'} - \text{tr} = \text{s}\theta'))) \\
= \text{IOCO}_3(P) \land (\text{tr} \leq \text{tr'}) \\
= \text{R}_1(\text{IOCO}_3(P))
\]

Lemma 14 (commutativity-IOCO3-R2) \( \text{IOCO}_3 \circ \text{R}_2 = \text{R}_2 \circ \text{IOCO}_3 \)

Proof.

\[
\text{IOCO}_3(\text{R}_2(P(\text{tr}, \text{tr'}))) = \\
= \text{IOCO}_3(P(\text{tr}, \text{tr'})) \\
= P(\text{tr}, \text{tr'} - \text{tr}) \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (\theta \not\in \text{ref'} \Rightarrow \exists s \bullet \text{tr'} - \text{tr} = \text{s}\theta'))) \\
= \text{R}_2(\text{IOCO}_3(P))
\]

Lemma 15 (commutativity-IOCO3-IOCO1) \( \text{IOCO}_3 \circ \text{IOCO}_1 = \text{IOCO}_1 \circ \text{IOCO}_3 \)

Proof.

\[
\text{IOCO}_3(\text{IOCO}_1(P)) = \\
= \text{IOCO}_3(P \land (\text{ok} \Rightarrow (\text{wait'} \lor \text{ok}'))) \\
= P \land (\text{ok} \Rightarrow (\text{wait'} \lor \text{ok}')) \land \\
\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (\theta \not\in \text{ref'} \Rightarrow \exists s \bullet \text{tr'} - \text{tr} = \text{s}\theta'))) \\
= P \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (\theta \not\in \text{ref'} \Rightarrow \exists s \bullet \text{tr'} - \text{tr} = \text{s}\theta'))) \land (\text{ok} \Rightarrow (\text{wait'} \lor \text{ok}')) \\
= \text{IOCO}_3(P) \land (\text{ok} \Rightarrow (\text{wait'} \lor \text{ok}'))
\]

(\text{IOCO}_1\text{I}(\text{IOCO}_3(P)))

Lemma 16 (commutativity-IOCO3-IOCO2) \( \text{IOCO}_3 \circ \text{IOCO}_2 = \text{IOCO}_2 \circ \text{IOCO}_3 \)
Proof.

\[
\text{IICO3}(\text{IICO2}(P)) = (\text{def. of IICO2}) \\
= \text{IICO3}(P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \text{quiescence})))) (\text{def. of IICO3}) \\
= P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \text{quiescence}))) \land \\
(\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \exists s \cdot \text{tr}' - \text{tr} = s \cdot \theta^*))) (\text{prop. calculus}) \\
= P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \exists s \cdot \text{tr}' - \text{tr} = s \cdot \theta^*))) \land \\
(\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \text{quiescence}))) (\text{def. of IICO3}) \\
= \text{IICO3}(P) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \text{quiescence}))) (\text{def. of IICO2}) \\
= \text{IICO3}(P) \\
\]

By introducing the observability of quiescence we need to change the definition of the skip (I) element: Processes always need to respect the properties of quiescence. Even in the case of divergence θ can be observed if and only if there is no output. This leads to \( I^0 \), which is defined as follows.

\[
I^0 = \exists (\neg \text{ok} \land (\text{tr} \leq \text{tr}')) \land (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \text{quiescence})) \land \\
(\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \exists s \cdot \text{tr}' - \text{tr} = s \cdot \theta^*)) \\
\lor (\text{ok}' \land (v' = v))
\]

In the above definition the variables \( v \) and \( v' \) denote the observation vectors, i.e., \( v = \{ \text{ok}, \text{wait}, \text{tr}, \text{ref} \} \) and \( v' = \{ \text{ok}', \text{wait}', \text{tr}', \text{ref}' \} \), respectively.

By introducing a new skip element we also need to change the definition of the healthiness condition \( R3 \). We will denote this modified healthiness condition as \( R3^0 \).

Lemma 17 (commutativity-IICO2-R3^0)

\[
\text{IICO2} \circ R3^0 = R3^0 \circ \text{IICO2}
\]

Proof.

\[
\text{IICO2}(R3^0(P)) = (\text{def. of R3^0}) \\
= \text{IICO2}(I^0 \triangleleft \text{wait} \triangleright P) (\text{def. of IICO2}) \\
= (I^0 \triangleleft \text{wait} \triangleright P) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \text{quiescence}))) (\text{prop. if and ~wait}) \\
= (I^0 \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \text{quiescence})))) < \text{wait} > \\
(P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \text{quiescence})))) \lor (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \text{quiescence}))) (\text{def. of IOCO2}) \\
= I^0 \triangleleft \text{wait} \triangleright (P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \text{quiescence})))) \lor (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \text{quiescence}))) (\text{def. of IOCO2}) \\
= I^0 \triangleleft \text{wait} \triangleright I\text{CO2}(P)) \lor (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \text{quiescence}))) (\text{def. of R3^0}) \\
= R3^0(I\text{CO2}(P)) (\text{def. of IICO2}) \\
\]

Lemma 18 (commutativity-IICO3-R3^0)

\[
\text{IICO3} \circ R3^0 = R3^0 \circ \text{IICO3}
\]
Proof.

\begin{align*}
\text{IOCO3(R3}\theta(P)) &= \quad \text{(def. of R3}\theta) \\
= \text{IOCO3}(1^\theta \triangleright wait \triangleright P) \quad \text{(def. of IOCO3)} \\
= (1^\theta \triangleright wait \triangleright P) \land \left(\neg\text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \exists s \cdot \text{tr}' - \text{tr} = \hat{s} \theta))\right) \quad \text{(\land\text{-if-distr)}} \\
= (1^\theta \land (\neg\text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \exists s \cdot \text{tr}' - \text{tr} = \hat{s} \theta)))) \triangleright \text{wait} \triangleright \\
(P \land (\neg\text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \exists s \cdot \text{tr}' - \text{tr} = \hat{s} \theta)))) \quad \text{(def. of if and \neg\text{wait})} \\
= 1^\theta \triangleright \neg wait \triangleright (P \land (\neg\text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \exists s \cdot \text{tr}' - \text{tr} = \hat{s} \theta)))) \quad \text{(def. of IOCO3)} \\
= 1^\theta \triangleright \text{wait} \triangleright \text{IOCO3}(P) \quad \text{(def. of R3}\theta) \\
= R3^\theta(\text{IOCO3}(P)) \\
\end{align*}

\text{IOCO2 and IOCO3 form a closure with respect to sequential composition. That is, composing two healthy processes using sequential composition gives again an healthy process.}

Lemma 19 (closure-\text{-};-\text{IOCO2})

\text{IOCO2}(P;Q) = P;Q \text{ provided P and Q are IOCO2 and R3}\theta \text{ healthy}
Proof.

\[ \text{IOCO2}(P; Q) = \]

\[ \text{IOCO2}(P; (1^0 \triangleleft \text{wait} \triangleright Q)) \] (assumption and def. of R3^0)

\[ \text{IOCO2}(\exists v_0 \cdot P[v_0/v'] \land (1^0 \triangleleft \text{wait} \triangleright Q)[v_0/v]) \] (def. of \( \exists \))

\[ \exists v_0 \cdot P[v_0/v'] \land (1^0[v_0/v] \land \text{wait}_0 \lor \neg \text{wait}_0 \land Q[v_0/v]) \land (\text{wait} \lor \neg \text{wait}' \lor (\theta \not\in \text{ref}' \Rightarrow \text{quiescence})) \] (def. of \( \exists \) and prop. calculus)

\[ \exists v_0 \cdot \left( P[v_0/v'] \land \text{wait}_0 \land \neg \text{ok}_0 \land \cdots \land (\neg \text{wait}' \lor (\theta \not\in \text{ref}' \Rightarrow \text{quiescence}) \lor \text{wait} \lor \neg \text{wait}' \lor (\theta \not\in \text{ref}' \Rightarrow \text{quiescence})) \right) \] (substitution and prop. calculus)

\[ \exists v_0 \cdot \left( P[v_0/v'] \land \text{wait}_0 \land \neg \text{ok}_0 \land \cdots \land (\neg \text{wait}' \lor (\theta \not\in \text{ref}' \Rightarrow \text{quiescence}) \lor \text{wait} \lor \neg \text{wait}' \lor (\theta \not\in \text{ref}' \Rightarrow \text{quiescence})) \right) \] (def. of \( \exists \) and assumption)

\[ \exists v_0 \cdot \left( P[v_0/v'] \land \text{wait}_0 \land \neg \text{ok}_0 \land \cdots \land (\neg \text{wait}' \lor (\theta \not\in \text{ref}' \Rightarrow \text{quiescence}) \lor \text{wait} \lor \neg \text{wait}' \lor (\theta \not\in \text{ref}' \Rightarrow \text{quiescence})) \right) \] (prop. calculus and def. of \( \exists \) and assumption)

\[ \exists v_0 \cdot \left( P[v_0/v'] \land \text{wait}_0 \land \neg \text{ok}_0 \land \cdots \land (\neg \text{wait}' \lor (\theta \not\in \text{ref}' \Rightarrow \text{quiescence}) \lor \text{wait} \lor \neg \text{wait}' \lor (\theta \not\in \text{ref}' \Rightarrow \text{quiescence})) \right) \] (def. of \( \exists \) and renaming)

\[ \exists v_0 \cdot \left( P[v_0/v'] \land \text{wait}_0 \land \neg \text{ok}_0 \land \cdots \land (\neg \text{wait}' \lor (\theta \not\in \text{ref}' \Rightarrow \text{quiescence}) \lor \text{wait} \lor \neg \text{wait}' \lor (\theta \not\in \text{ref}' \Rightarrow \text{quiescence})) \right) \] (prop. calculus)

\[ \exists v_0 \cdot \left( P[v_0/v'] \land \text{wait}_0 \land \neg \text{ok}_0 \land \cdots \land (\neg \text{wait}' \lor (\theta \not\in \text{ref}' \Rightarrow \text{quiescence}) \lor \text{wait} \lor \neg \text{wait}' \lor (\theta \not\in \text{ref}' \Rightarrow \text{quiescence})) \right) \] (prop. calculus)

\[ \exists v_0 \cdot \left( P[v_0/v'] \land \text{wait}_0 \land \neg \text{ok}_0 \land \cdots \land (\neg \text{wait}' \lor (\theta \not\in \text{ref}' \Rightarrow \text{quiescence}) \lor \text{wait} \lor \neg \text{wait}' \lor (\theta \not\in \text{ref}' \Rightarrow \text{quiescence})) \right) \] (substitution and def. of \( \exists \))

\[ \exists v_0 \cdot P[v_0/v'] \land \text{wait}_0 \lor \neg \text{wait}_0 \lor \text{IOCO2}(Q)[v_0/v] \] (assumption and def. of R3^0)

\[ \exists v_0 \cdot P[v_0/v'] \land R3^0(Q)[v_0/v] \] (assumption and def. of \( \exists \))

\[ P; Q \]

\[ \square \]
Lemma 20 (closure-;-IOCO3)

\[ IOCO3(P; Q) = P; Q \text{ provided } P \text{ and } Q \text{ are } IOCO3 \text{ and } R3 \text{ healthy} \]

**Proof.** Similar to the proof of Lemma 19.

Since the quiescence event \( \theta \) encodes the absence of output events it may occur at any time. This is even true for the deadlock process.

**Definition 13 (Quiescent deadlock)**

\[ \delta^\theta = \text{def } R3^\theta(tr' - tr \in \theta^* \land \text{wait}') \]

Consequently, the classical deadlock process indicating absolute inactivity does not exist within the ioco theory.

As the deadlock, also the quiescent deadlock is a left zero for sequential composition:

**Lemma 21 (\( \delta^\theta \)-left-zero)**

\[ \delta^\theta; P = \delta^\theta \]

**Proof.**

\[
\begin{align*}
\delta^\theta; P &= & \text{(closure of } R3^\theta) \\
= \tilde{\text{I}}^\theta < \text{wait} \triangleright (\delta^\theta; P) & \quad \text{(def. of } \delta^\theta) \\
= \tilde{\text{I}}^\theta < \text{wait} \triangleright ((\tilde{\text{I}}^\theta < \text{wait} \triangleright (tr' - tr \in \theta^* \land \text{wait}'); P) & \quad \text{(def. of } \triangleright) \\
= \tilde{\text{I}}^\theta < \text{wait} \triangleright ((tr' - tr \in \theta^* \land \text{wait}'); \tilde{\text{I}}^\theta < \text{wait} \triangleright P) & \quad \text{(P meets } R3) \\
= \tilde{\text{I}}^\theta < \text{wait} \triangleright (\tilde{\text{I}}^\theta < \text{wait} \triangleright P) & \quad \text{(P meets } R3) \\
= \tilde{\text{I}}^\theta < \text{wait} \triangleright ((tr' - tr \in \theta^* \land \text{wait}')); \tilde{\text{I}}^\theta & \quad \text{(; unit)} \\
= \tilde{\text{I}}^\theta < \text{wait} \triangleright ((tr' - tr \in \theta^* \land \text{wait}')) & \quad \text{(def. of } \delta^\theta) \\
= \delta^\theta & \quad \square
\end{align*}
\]

Although quiescence is preserved by sequential composition, we need to redefine internal and external choices in order to preserve the properties of quiescence. Basically, the composition of processes that start with input actions (i.e. the processes are quiescent initially) is quiescent. If one of the two composed processes is not quiescent initially, the composition is not quiescent either.

For our quiescence preserving composition operators (\( \cap^\theta, +^\theta \)) we use an approach similar to parallel by merge [HH98]. The idea of parallel by merge is to run two processes independently and merge their results afterwards. In order to express independent execution we need a relabeling function. Given an output alphabet \( \{v'_1, v'_2, \ldots, v'_n\} \), \( U_l \) is defined as follows

**Definition 14 (Relabelling)**

\[
\begin{align*}
\alpha U_l(\{v'_1, v'_2, \ldots, v'_n\}) &= \text{def } \{v_1, v_2, \ldots, v_n, l.v'_1, l.v'_2, \ldots, l.v'_n\} \\
U_l(\{v'_1, v'_2, \ldots, v'_n\}) &= \text{def } (l.v'_1 = v_1) \land (l.v'_2 = v_2) \land \cdots \land (l.v'_n = v_n)
\end{align*}
\]

Independent execution of \( P \) and \( Q \) is now expressed by relabeling:

**Definition 15 (Independent execution)**

\[ P \downarrow Q = \text{def } P; U_0(\text{out } \alpha P) \land Q; U_1(\text{out } \alpha Q) \]
Given the independent execution of two processes and a merge relation we can unfold the definition as follows:

**Lemma 22 (unfolded-independent-execution)**

\[
(P \triangleleft Q); M = \exists v_0, v_1 \cdot P[v_0/v'] \land Q[v_1/v'] \land M[v_0, v_1/0.v, 1.v]
\]

**Proof.**

\[
(P \triangleleft Q); M =
\begin{align*}
&= (P; U_0(outaP) \land Q; U_1(outaQ)); M \\
&= ((\exists v_0 \cdot P[v_0/v'] \land (0.v' = v_0)) \land (\exists v_1 \cdot Q[v_1/v'] \land (1.v' = v_1))); \\
&= (P[0.v'/v'] \land Q[1.v'/v']); M \\
&= \exists v_0, v_1 \cdot (P[0.v'/v'] \land Q[1.v'/v'])[v_0, v_1/0.v, 1.v] \land M[v_0, v_1/0.v, 1.v]
\end{align*}
\]

An internal choice, which takes care of \( \theta \) within the resulting process, can be defined as follows:

**Definition 16 (Quiescence preserving internal choice)**

\[
P \; \triangleright^\theta \; Q =_d (P \triangleleft Q); M \Gamma \; with 
M \Gamma \; =_d M^[\theta] \Gamma \; < \delta^\theta \; > M^{\sim \delta^\theta}
\]

As this definition illustrates we need two merge relations for the quiescence preserving internal choice: \( M^[\theta] \) and \( M^{\sim \delta^\theta} \). \( M^[\theta] \) merges the very beginning of the two processes \( P \) and \( Q \). After that, \( M^{\sim \delta^\theta} \) takes care that \( P \triangleright^\theta Q \) behaves like \( P \) or \( Q \).

\( M^[\theta] \) and \( M^{\sim \delta^\theta} \) share some common properties, formalized by \( M^\theta \): (1) Parts of \( P \) and \( Q \) are only merged if their \( wait' \) values are equal, and (2) potentially the initial \( \theta \) has to be removed from all traces.

**Definition 17 (Internal choice - Common merge)**

\[
M^\theta =_d (0.wait \iff 1.wait) \land wait' = 0.wait \land (ok' = (0.ok \land 1.ok)) \land
((-initQuiet(0.tr - tr) \lor \neg initQuiet(1.tr - tr)) \implies \neg initQuiet(tr' - tr))
\]

where \( initQuiet(t) =_d t \not\in \{s|t|s \in A \land u \in A^*_0 \} \cup \{()\} \)

The merge relation \( M^\theta \) is symmetric in its variables, i.e. in \( 0.v = \{0.wait, 0.ok, 0.ref, 0.tr\} \) and in \( 1.v = \{1.wait, 1.ok, 1.ref, 1.tr\} \):

**Lemma 23 (symmetric-M^\theta)**

\[
M^\theta[0.v, 1.v/1.v, 0.v] = M^\theta
\]

**Proof.**

\[
M^\theta[0.v, 1.v/1.v, 0.v] =
\begin{align*}
&= \left( (0.wait \iff 1.wait) \land wait' = 0.wait \land (ok' = (0.ok \land 1.ok)) \land
((-initQuiet(0.tr - tr) \lor \neg initQuiet(1.tr - tr)) \implies \neg initQuiet(tr' - tr)) \right) [0.v, 1.v/1.v, 0.v] \\
&= (0.wait \iff 0.wait) \land wait' = 0.wait \land (ok' = (1.ok \land 0.ok)) \land
((-initQuiet(1.tr - tr) \lor \neg initQuiet(0.tr - tr)) \implies \neg initQuiet(tr' - tr)) \\
&= (0.wait \iff 1.wait) \land wait' = 0.wait \land (ok' = (0.ok \land 1.ok)) \land
((-initQuiet(0.tr - tr) \lor \neg initQuiet(1.tr - tr)) \implies \neg initQuiet(tr' - tr)) \\
&= M^\theta
\end{align*}
\]

Furthermore, the common merge relation \( M^\theta \) is equivalent to having \( wait' = (0.wait \land 1.wait) \) and \( ok' = (0.ok \land 1.ok) \) in conjunction with the merge relation itself:
Lemma 24 (wait-and-ok-$M^\theta$)

$$M_\theta = (\text{0.wait} \Leftrightarrow \text{1.wait}) \land (\text{wait}' = (\text{0.wait} \lor \text{1.wait})) \land (\text{ok}' = (\text{0.ok} \land \text{1.ok})) \land M^\theta$$

Proof.

$$M^\theta =$$

\[
= (\text{0.wait} \Leftrightarrow \text{1.wait}) \land (\text{wait}' = \text{0.wait} \land (\text{ok}' = (\text{0.ok} \land \text{1.ok}))) \land \\
(\neg \text{initQuiet}(0.\text{tr} - \text{tr}) \lor \neg \text{initQuiet}(1.\text{tr} - \text{tr}) \Rightarrow \neg \text{initQuiet}(\text{tr}' - \text{tr})) \quad \text{(def. of } M^\theta) \\
= (\text{0.wait} \Leftrightarrow \text{1.wait}) \land (\text{wait}' = \text{0.wait} \land (\text{ok}' = (\text{0.ok} \land \text{1.ok}))) \land \\
(\neg \text{initQuiet}(0.\text{tr} - \text{tr}) \lor \neg \text{initQuiet}(1.\text{tr} - \text{tr}) \Rightarrow \neg \text{initQuiet}(\text{tr}' - \text{tr})) \quad \text{(prop. calculus)}
\]

$$M^\theta_{\cap}$$ can now be defined by the use of $M^\theta$. In addition to the common merge relation, traces and refusal sets of $P$ and $Q$ are merged into new traces and new refusal sets.

Definition 18 (Internal choice - Initial merge)

$$M^\theta_{\cap} =_{sj} M^\theta \land M^\theta_{\cap}^{\text{init}}$$

$$M^\theta_{\cap}^{\text{init}} =_{sj} ((\text{tr}' = 0.\text{tr} \land \text{ref}' = (\text{0.ref} \setminus \{\theta\}) \cup (\{\theta\} \cap (\text{0.ref} \cup \text{1.ref}))) \lor \\
(\text{tr}' = 1.\text{tr} \land \text{ref}' = (\text{1.ref} \setminus \{\theta\}) \cup (\{\theta\} \cap (\text{0.ref} \cup \text{1.ref}))))$$

The merge relation $M^\theta_{\cap}$ is symmetric in its variables, i.e. in $0.v = \{\text{0.wait}, \text{0.ok}, \text{0.ref}, \text{0.tr}\}$ and in $1.v = \{\text{1.wait}, \text{1.ok}, \text{1.ref}, \text{1.tr}\}$.

Lemma 25 (symmetric-$M^\theta_{\cap}$)

$$M^\theta_{\cap}^{\theta} [0.v, 1.v/1.v, 0.v] = M^\theta_{\cap}$$

Proof.

$$M^\theta_{\cap}^{\theta} [0.v, 1.v/1.v, 0.v] =$$

\[
= (M^\theta \land M^\theta_{\cap}^{\text{init}})[0.v, 1.v/1.v, 0.v] \quad \text{(def. of } M^\theta_{\cap}^{\theta}) \\
= (M^\theta \land (\text{0.wait} \land (\text{0.ok} \land \text{1.ok}))) \land (\neg \text{initQuiet}(0.\text{tr} - \text{tr}) \lor \neg \text{initQuiet}(1.\text{tr} - \text{tr}) \Rightarrow \neg \text{initQuiet}(\text{tr}' - \text{tr})) \quad \text{(def. of } M^\theta_{\cap}^{\theta}) \\
= M^\theta \land (\text{0.wait} \land (\text{0.ok} \land \text{1.ok})) \land (\neg \text{initQuiet}(0.\text{tr} - \text{tr}) \lor \neg \text{initQuiet}(1.\text{tr} - \text{tr}) \Rightarrow \neg \text{initQuiet}(\text{tr}' - \text{tr})) \quad \text{(def. of } M^\theta_{\cap}^{\theta}) \\
= M^\theta \land (\text{0.wait} \land (\text{0.ok} \land \text{1.ok})) \land (\neg \text{initQuiet}(0.\text{tr} - \text{tr}) \lor \neg \text{initQuiet}(1.\text{tr} - \text{tr}) \Rightarrow \neg \text{initQuiet}(\text{tr}' - \text{tr})) \quad \text{(def. of } M^\theta_{\cap}^{\theta}) \\
= M^\theta \land M^\theta_{\cap}^{\text{init}} \quad \text{(def. of } M^\theta_{\cap}) \\
= M^\theta_{\cap} \quad \text{(def. of } M^\theta_{\cap})
\]

By adding $\{\theta\} \cap (\text{0.ref} \cup \text{1.ref})$ to the set of refused actions the new process refuses $\theta$ only if one of the two processes refuse to exhibit a $\theta$ event. In other words, only if both processes do not refuse $\theta$, i.e., $\theta \notin \text{0.ref} \land \theta \notin \text{1.ref}$, the resulting process does not refuse $\theta$ as well, i.e., $\theta \notin \text{ref}'$. 


\( M^{\delta} \) takes care that finally \( P \sqcap^\theta Q \) behaves like \( P \) or \( Q \). Additionally, \( M^\theta \) is applied in order to potentially remove \( \theta \) from the traces.

**Definition 19 (Internal choice - Terminal merge)**
\[
M^{\delta} =_{df} M^\theta \land M^{term},
\]
\[
M^{term} =_{df} ((tr' = 0.\text{tr} \land \text{ref}' = 0.\text{ref}) \lor (tr' = 1.\text{tr} \land \text{ref}' = 1.\text{ref})).
\]

The merge relation \( M^{\delta} \) is symmetric in its variables, i.e. in \( 0.v = \{0.\text{wait}, 0.\text{ok}, 0.\text{ref}, 0.\text{tr}\} \) and in \( 1.v = \{1.\text{wait}, 1.\text{ok}, 1.\text{ref}, 1.\text{tr}\} \):

**Lemma 26 (symmetric-\( M^{\delta} \))**
\[
M^{\delta}[0.v, 1.v/1.v, 0.v] = M^{\delta}\]

**Proof.**
\[
M^{\delta}[0.v, 1.v/1.v, 0.v] = (M^\theta \land M^{term})[0.v, 1.v/1.v, 0.v] \quad \text{(def. of } M^{\delta} \text{)}
\]
\[
= (M^\theta \land ((tr' = 0.\text{tr} \land \text{ref}' = 0.\text{ref}) \lor (tr' = 1.\text{tr} \land \text{ref}' = 1.\text{ref})))[0.v, 1.v/1.v, 0.v] \quad \text{(def. of } \llbracket \text{ } \rrbracket \text{)}
\]
\[
= M^\theta[0.v, 1.v/1.v, 0.v] \land ((tr' = 1.\text{tr} \land \text{ref}' = 1.\text{ref}) \lor (tr' = 0.\text{tr} \land \text{ref}' = 0.\text{ref})) \quad \text{(Lemma 23)}
\]
\[
= M^\theta \land ((tr' = 1.\text{tr} \land \text{ref}' = 1.\text{ref}) \lor (tr' = 0.\text{tr} \land \text{ref}' = 0.\text{ref})) \quad \text{(prop. calculus)}
\]
\[
= M^\theta \land ((tr' = 0.\text{tr} \land \text{ref}' = 0.\text{ref}) \lor (tr' = 1.\text{tr} \land \text{ref}' = 1.\text{ref})) \quad \text{(def. of } M^{term} \text{)}
\]
\[
= M^\theta \land M^{term} \quad \text{(def. of } M^{\delta} \text{)}
\]
\[
= M^{\delta} \quad \Box
\]

Our internal choice’s merge relation is equal to setting the value of \( tr' \) to either \( 0.\text{tr} \) or \( 1.\text{tr} \) and the merge relation itself, i.e.,

**Lemma 27 (tr-\( M_\gamma \))**
\[
M_\gamma = (tr' = 0.\text{tr} \lor tr' = 1.\text{tr}) \land M_\gamma
\]
Proof.

\[ M_\cap = (\text{def. of } M_\cap) \]

\[ = M_\cap^\theta < \delta^\theta > M^{-\delta^\theta} \]

\[ = (M^\theta \land M_\cap^{\init}) < \delta^\theta > (M^\theta \land M^{\term}) \]

\[ = \left( M^\theta \land \left( \begin{array}{l}
tr' = 0.tr \land (ref' = \ldots) \lor \\
tr' = 1.tr \land (ref' = \ldots )
\end{array} \right) \right) < \delta^\theta > \left( M^\theta \land \left( \begin{array}{l}
tr' = 0.tr \land (ref' = 0.ref) \lor \\
tr' = 1.tr \land (ref' = 1.ref)
\end{array} \right) \right) \]

\[ = \left( (tr' = 0.tr \lor tr' = 1.tr) \land \\
M^\theta \land (tr' = 0.tr \land (ref' = \ldots)) \lor \\
tr' = 1.tr \land (ref' = \ldots ) \right) < \delta^\theta > \left( \begin{array}{l}
M^\theta \land \\
M^\theta \land (tr' = 0.tr \land (ref' = 0.ref) \lor \\
tr' = 1.tr \land (ref' = 1.ref)
\end{array} \right) \]

\[ = (tr' = 0.tr \lor tr' = 1.tr) \land \left( M^\theta \land M_\cap^{\init} \right) < \delta^\theta > ((tr' = 0.tr \lor tr' = 1.tr) \land M^\theta \land M^{\term}) \]

Furthermore, the merge relation \( M_\cap \) is equivalent to having \( \text{wait}' = (0.\text{wait} \land 1.\text{wait}) \) and \( \text{ok}' = (0.\text{ok} \land 1.\text{ok}) \) in conjunction with the merge relation itself:

**Lemma 28 (wait-and-ok-M_\cap)**

\[ M_\cap = (0.\text{wait} \iff 1.\text{wait}) \land (\text{wait}' = (0.\text{wait} \lor 1.\text{wait})) \land (\text{ok}' = (0.\text{ok} \land 1.\text{ok})) \land M_\cap \]

Proof.

\[ M_\cap = (\text{def. of } M_\cap) \]

\[ = M_\cap^\theta < \delta^\theta > M^{-\delta^\theta} \]

\[ = (M^\theta \land M_\cap^{\init}) < \delta^\theta > M^\theta \land M^{\term} \]

\[ = \left( \begin{array}{l}
(\text{wait} = 1.\text{wait}) \land \\
(\text{ok}' = (0.\text{ok} \land 1.\text{ok})) \land
\end{array} \right) \land \left( \begin{array}{l}
(\text{wait}' = (0.\text{wait} \lor 1.\text{wait})) \land \\
(\text{ok}' = (0.\text{ok} \land 1.\text{ok})) \land
\end{array} \right) \]

\[ = (0.\text{wait} \iff 1.\text{wait}) \land (\text{wait}' = (0.\text{wait} \lor 1.\text{wait})) \land (\text{ok}' = (0.\text{ok} \land 1.\text{ok})) \land M_\cap \]

We can also extract the calculation of the refusal sets from the merge relation:
Lemma 29 (wait-and-ref-M_r)

\[ M_r = M_r \land (wait' = (0.wait \land 1.wait)) \land \]

\[
\left( \begin{array}{l}
tr' = 0.tr \land ref_{in}' = 0.ref_{in} \land ref_{out}' = 0.ref_{out} \\
ref' = (0.ref \setminus \{\theta\}) \cup (\{\theta\} \cap (0.ref \cup 1.ref)) \forall \\
\end{array} \right) < \delta^0 \implies \left( \begin{array}{l}
tr' = 1.tr \land ref_{in}' = 1.ref_{in} \land ref_{out}' = 1.ref_{out} \\
ref' = (1.ref \setminus \{\theta\}) \cup (\{\theta\} \cap (0.ref \cup 1.ref)) \forall \\
\end{array} \right)
\]

Proof.

\[ M_r = \]

\[ M_r^\theta \land \delta^0 \triangleright M_r^{-\theta} \]

\[ (M^\theta \land M_r^{init}) \land \delta^0 \triangleright (M^\theta \land M^{term}) \]

\[ M^\theta \land (M_r^{init} \land \delta^0 \triangleright M^{term}) \]

\[ (0.wait \iff 1.wait) \land (wait' = (0.wait \lor 1.wait)) \land (ok' = (0.ok \land 1.ok)) \land \]

\[ M^\theta \land (M_r^{init} \land \delta^0 \triangleright M^{term}) \]

\[ (0.wait \iff 1.wait) \land (wait' = (0.wait \lor 1.wait)) \land (wait' = (0.wait \land 1.wait)) \land (ok' = (0.ok \land 1.ok)) \land \]

\[ M^\theta \land (M_r^{init} \land \delta^0 \triangleright M^{term}) \]

\[ (0.wait \iff 1.wait) \land (wait' = (0.wait \land 1.wait)) \land (wait' = (0.wait \land 1.wait)) \land (ok' = (0.ok \land 1.ok)) \land \]

\[ M^\theta \land (M_r^{init} \land \delta^0 \triangleright M^{term}) \]

\[ (0.wait \iff 1.wait) \land (wait' = (0.wait \land 1.wait)) \land (ok' = (0.ok \land 1.ok)) \land \]

\[ M^\theta \land (M_r^{init} \land \delta^0 \triangleright M^{term}) \]

\[ (0.wait \iff 1.wait) \land (wait' = (0.wait \land 1.wait)) \land (ok' = (0.ok \land 1.ok)) \land \]

\[ M^\theta \land (M_r^{init} \land \delta^0 \triangleright M^{term}) \]

\[ (0.wait \iff 1.wait) \land (wait' = (0.wait \land 1.wait)) \land (ok' = (0.ok \land 1.ok)) \land \]

\[ M^\theta \land (M_r^{init} \land \delta^0 \triangleright M^{term}) \]

\[ (0.wait \iff 1.wait) \land (wait' = (0.wait \land 1.wait)) \land (ok' = (0.ok \land 1.ok)) \land \]

\[ M^\theta \land (M_r^{init} \land \delta^0 \triangleright M^{term}) \]
The merge relation $M_\cap$ is symmetric in its variables, i.e. in $0.v = \{0.wait, 0.ok, 0.ref, 0.tr\}$ and in $1.v = \{1.wait, 1.ok, 1.ref, 1.tr\}$:

**Lemma 30 (symmetric-$M_\cap$) $M_\cap[0.v,1.v/1.v,0.v] = M_\cap$**

**Proof.**

\[
M_\cap[0.v,1.v/1.v,0.v] = (\text{def. of } M_\cap) \\
= (M_\cap^\theta \triangleleft \delta^\theta \triangleright M^{-\delta^\theta})[0.v,1.v/1.v,0.v] (\text{no } 1.v \text{ and } 0.v \text{ in } \delta^\theta) \\
= M_\cap^\theta[0.v,1.v/1.v,0.v] < \delta^\theta \triangleright M^{-\delta^\theta}[0.v,1.v/1.v,0.v] (\text{Lemma 25}) \\
= M_\cap^\theta < \delta^\theta \triangleright M^{-\delta^\theta}[0.v,1.v/1.v,0.v] (\text{Lemma 26}) \\
= M_\cap^\theta < \delta^\theta \triangleright M^{-\delta^\theta} (\text{def. of } M_\cap) \\
= M_\cap \tag*{\square}
\]

If the values for $0.v$ and $1.v$ in $M_\cap$ are the same, than $M_\cap$ does not change anything, i.e. it reduces to the relational skip [CW04] $I_{rel} = (wait' = wait) \land (ok' = ok) \land (ref' = ref) \land (tr' = tr)$.

**Lemma 31 ($M_\cap$-reduces-to-skip) $M_\cap[v_0,v_0/1.v,0.v] = I_{rel}[v_0/v]$**
Proof.

\[ M_{\gamma}[v_0, v_0/v, 0, v, 1.v] = \]

\[(M_{\gamma}^\theta \triangleleft \delta^\theta \triangleright M_{\gamma}^{-\theta})[v_0, v_0/0.v, 1.v] \]

\[(M^\theta \land M_{\gamma}^{init} \triangleleft \delta^\theta \triangleright M^\theta \land M^{term})[v_0, v_0/0.v, 1.v] \]

(\text{def. of } M_{\gamma}^\theta \text{ and of } M^{-\theta})

(\text{def. of merge relations})

\[
\begin{align*}
\left(0.wait \Rightarrow 1.wait \right) \land \left(0.wait \land ok' = (0.ok \land 1.ok)\right) \\
\left(\neg \text{initQuiet}(0.tr - tr) \lor \neg \text{initQuiet}(1.tr - tr) \Rightarrow \neg \text{initQuiet}(tr' - tr)\right) \\
\left((tr' = 0.tr \land ref' = (0.ref \setminus \{\theta\}) \cup (\{\theta\} \cap (0.ref \cup 1.ref)))\right) \\
\left((tr' = 1.tr \land ref' = (1.ref \setminus \{\theta\}) \cup (\{\theta\} \cap (0.ref \cup 1.ref)))\right)
\end{align*}
\]

(\text{def. of } \left[ v_0, v_0/v, 0, v, 1.v \right])

(\text{def. of } [])

\[
\begin{align*}
\left(\text{wait}_0 \Leftrightarrow \text{wait}_0 \right) \land \left(\text{wait}' = \text{wait}_0 \land ok' = (0.ok \land 0.ok)\right) \\
\left(\neg \text{initQuiet}(tr_0 - tr) \lor \neg \text{initQuiet}(tr_0 - tr) \Rightarrow \neg \text{initQuiet}(tr' - tr)\right) \\
\left((tr' = tr_0 \land ref' = (ref_0 \setminus \{\theta\}) \cup (\{\theta\} \cap (ref_0 \cup ref_0)))\right) \\
\left((tr' = tr_0 \land ref' = (ref_0 \setminus \{\theta\}) \cup (\{\theta\} \cap (ref_0 \cup ref_0)))\right)
\end{align*}
\]

(\text{def. of } \left[ v_0, v_0/0, v, 1.v \right]\delta^\theta)

(\text{prop. calculus})

\[
\begin{align*}
\left(\text{wait}' = \text{wait}_0 \land ok' = 0.ok\right) \land \\
\left(\neg \text{initQuiet}(tr_0 - tr) \Rightarrow \neg \text{initQuiet}(tr' - tr)\right) \\
\left((tr' = tr_0 \land ref' = ref_0)\right)
\end{align*}
\]

(\text{prop. calculus})

\[
\begin{align*}
\left(\text{wait}' = \text{wait}_0 \land ok' = 0.ok\right) \land \\
\left(\neg \text{initQuiet}(tr_0 - tr) \Rightarrow \neg \text{initQuiet}(tr' - tr)\right) \\
\left((tr' = tr_0 \land ref' = ref_0)\right)
\end{align*}
\]

(\text{cond. idemp})

\[
= \text{wait}' = \text{wait}_0 \land ok' = 0.ok \land (\neg \text{initQuiet}(tr_0 - tr) \Rightarrow \neg \text{initQuiet}(tr' - tr)) \land (tr' = tr_0 \land ref' = ref_0)
\]

(\text{prop. calculus})

\[
= \text{wait}' = \text{wait}_0 \land ok' = 0.ok \land (\neg \text{initQuiet}(tr_0 - tr) \Rightarrow \neg \text{initQuiet}(tr' - tr)) \land (tr' = tr_0 \land ref' = ref_0)
\]

(\text{prop. calculus})

\[
= \text{wait}' = \text{wait}_0 \land ok' = 0.ok \land tr' = tr_0 \land ref' = ref_0
\]

(\text{def. of } \mathbb{I}_{\text{rel}})

\[
= \mathbb{I}_{\text{rel}}[v_0/v]
\]

(\text{def. of } \mathbb{I}_{\text{rel}})

\[
\square
\]

Our quiescence preserving internal choices \( \cap^\theta \) obey the usual laws for choices, i.e. quiescence preserving internal choices are idempotent and commutative.

Lemma 32 (idempotent-\( \cap^\theta \))

\[ P \cap^\theta P = P \]
Proof.

\[ P \cap^\theta P = \]  
\[ = (P \cap P); M \gamma \]  
\[ = \exists v_0, v_1 \cdot P[v_0/v'] \land P[v_1/v'] \land M \gamma[v_0, v_1/0.v, 1.v] \]  
\[ = \exists v_0 \cdot P[v_0/v'] \land P[v_0/v'] \land M \gamma[v_0, v_0/0.v, 1.v] \]  
\[ = \exists v_0 \cdot P[v_0/v'] \land M \gamma[v_0, v_0/0.v, 1.v] \]  
\[ = \exists v_0 \cdot P[v_0/v'] \land \mathbb{I}_{rel}[v_0, v] \]  
\[ = P; \mathbb{I}_{rel} \]  
\[ = P \]

Lemma 33 (commutative-\( \cap^\theta \))

\[ P \cap^\theta Q = Q \cap^\theta P \]

Proof.

\[ P \cap^\theta Q = \]  
\[ = (P \cap Q); M \gamma \]  
\[ = \exists v_0, v_1 \cdot P[v_0/v'] \land Q[v_1/v'] \land M \gamma[v_0, v_1/0.v, 1.v] \]  
\[ = \exists v_0, v_1 \cdot P[v_1/v'] \land Q[v_0/v'] \land M \gamma[v_1, v_0/0.v, 1.v] \]  
\[ = \exists v_0 \cdot P[v_1/v'] \land Q[v_0/v'] \land (M \gamma[0.v, 1.v/1.v, 0.v])[v_1, v_0/0.v, 1.v] \]  
\[ = \exists v_0, v_1 \cdot P[v_1/v'] \land Q[v_0/v'] \land M \gamma[v_1, v_0/1.v, 0.v] \]  
\[ = \exists v_0, v_1 \cdot P[v_1/v'] \land Q[v_0/v'] \land M \gamma[v_0, v_1/0.v, 1.v] \]  
\[ = \exists v_0, v_1 \cdot Q[v_0/v'] \land P[v_1/v'] \land M \gamma[v_0, v_1/0.v, 1.v] \]  
\[ = (Q \cap P); M \gamma \]  
\[ = Q \cap^\theta P \]

Furthermore, our \( \cap^\theta \) operator preserves healthiness with respect to \( R1, R2, R3, IOCO1, IOCO2, \) and \( IOCO3. \)

Lemma 34 (closure-\( \cap^\theta \)-R1)

\[ R1(P \cap^\theta Q) = P \cap^\theta Q \text{ provided } P \text{ and } Q \text{ are } R1 \text{ healthy} \]
Lemma 35 (closure-\( GAP \))

\[ \text{R2}(P \ \triangledown \ Q) = P \ \triangledown \ Q \] provided \( P \) and \( Q \) are \( R2 \) healthy
Proof.

\[ R2(P(tr, tr')) \vdash Q(tr, tr') = \]  
(def of R2)

\[ = (P(tr, tr')) \vdash Q(tr, tr')(\emptyset , tr', tr) \]  
(def of \( \vdash \))

\[ = ((P(tr, tr') \land Q(tr, tr'))(\emptyset , tr', tr)) \]  
(Lemma 22)

\[ = (\exists v_0, v_1 \cdot P(tr, tr')[v_0/v'] \land Q(tr, tr')[v_1/v'] \land M_1[v_0, v_1/0,v,1,v](tr, tr')(\emptyset , tr', tr)) \]  
(assumption)

\[ = (\exists v_0, v_1 \cdot R2(P(tr, tr'))[v_0/v'] \land R2(Q)(tr, tr')[v_1/v'] \land M_1[v_0, v_1/0,v,1,v](tr, tr')(\emptyset , tr', tr)) \]  
(def of R2)

\[ = (\exists v_0, v_1 \cdot P(\emptyset , tr', tr)[v_0/v'] \land Q(\emptyset , tr', tr)[v_1/v'] \land M_1[v_0, v_1/0,v,1,v](tr, tr')(\emptyset , tr', tr)) \]  
(substitution)

\[ = (\exists v_0, v_1 \cdot P[v_0/v'](\emptyset , tr0 - tr) \land Q[v_1/v'](\emptyset , tr1 - tr) \land \]  
{def of \( M_1 \) and substitution)

\[ \big( (\text{wait}0 \leftrightarrow \text{wait}') = \text{wait}0 \land (\text{ok}' = (\text{ok}0 \land \text{ok}1)) \land 
\neg \text{initQuiet}(\text{tr}0 - tr) \land \neg \text{initQuiet}(\text{tr}1 - tr) \Rightarrow \neg \text{initQuiet}(\text{tr}' - tr) \land 
((\text{tr}' = \text{tr}0 \land \text{ref}' = \ldots) \lor (\text{tr}' = \text{tr}1 \land \text{ref}' = \text{dofs}))) \big) < \delta^0 > \]

\[ = (\exists v_0, v_1 \cdot P[v_0/v'](\emptyset , tr0 - tr) \land Q[v_1/v'](\emptyset , tr1 - tr) \land \]  
{substitution}

\[ \big( (\text{wait}0 \leftrightarrow \text{wait}') = \text{wait}0 \land (\text{ok}' = (\text{ok}0 \land \text{ok}1)) \land 
\neg \text{initQuiet}(\text{tr}0 - tr - \emptyset) \lor \neg \text{initQuiet}(\text{tr}1 - tr - \emptyset) \Rightarrow \neg \text{initQuiet}(\text{tr}' - tr - \emptyset) \land 
((\text{tr}' = \text{tr}0 \land \text{ref}' = \ldots) \lor (\text{tr}' = \text{tr}1 \land \text{ref}' = \text{dofs}))) \big) < \delta^0 > \]

\[ = (\exists v_0, v_1 \cdot P[v_0/v'](\emptyset , tr0 - tr) \land Q[v_1/v'](\emptyset , tr1 - tr) \land \]  
{prop. calculus}

\[ \big( (\text{wait}0 \leftrightarrow \text{wait}') = \text{wait}0 \land (\text{ok}' = (\text{ok}0 \land \text{ok}1)) \land 
\neg \text{initQuiet}(\text{tr}0 - tr - \emptyset) \lor \neg \text{initQuiet}(\text{tr}1 - tr - \emptyset) \Rightarrow \neg \text{initQuiet}(\text{tr}' - tr - \emptyset) \land 
((\text{tr}' = \text{tr}0 \land \text{ref}' = \ldots) \lor (\text{tr}' = \text{tr}1 \land \text{ref}' = \text{dofs}))) \big) < \delta^0 > \]

\[ = (\exists v_0, v_1 \cdot P[v_0/v'](\emptyset , tr0 - tr) \land Q[v_1/v'](\emptyset , tr1 - tr) \land M_1[v_0, v_1/0,v,1,v] \]  
{def of \( M_1 \) and substitution)

\[ = (\exists v_0, v_1 \cdot P(\emptyset , tr', tr)[v_0/v'] \land Q(\emptyset , tr', tr)[v_1/v'] \land M_1[v_0, v_1/0,v,1,v] \]  
(substitution)

\[ = (\exists v_0, v_1 \cdot R2(P(tr, tr'))[v_0/v'] \land R2(Q)(tr, tr')[v_1/v'] \land M_1[v_0, v_1/0,v,1,v] \]  
(def of R2)

\[ = (\exists v_0, v_1 \cdot P(tr, tr')[v_0/v'] \land Q(tr, tr')[v_1/v'] \land M_1[v_0, v_1/0,v,1,v] \]  
(Lemma 22)

\[ = (P(tr, tr') \land Q(tr, tr')) \land M_1[v_0, v_1/0,v,1,v] \]  
(Lemma 22)

\[ = P(tr, tr') \vdash Q(tr, tr') \]  
(def of \( \vdash \))

\[ \square \]

Lemma 36 (closure-\( \vdash \)-R3\( \delta \))

\[ R3^\delta (P \vdash Q) = P \vdash Q \]  
provided \( P \) and \( Q \) are \( R3^\delta \) healthy
Proof.

\[ R^\theta(P \cap^\theta Q) = \]
\[ = \emptyset \triangleleft \text{wait} \triangleright (P \cap^\theta Q) \] (def. of \( R^\theta \))
\[ = (\emptyset \cap^\theta \emptyset) \triangleleft \text{wait} \triangleright (P \cap^\theta Q) \] (def. of \( \cap^\theta \))
\[ = (\emptyset \triangleleft \emptyset); M_\emptyset \triangleleft \text{wait} \triangleright (P \triangleleft Q); M_\emptyset \] (Lemma 22)
\[ = (\exists v_0, v_1 \cdot \emptyset[v_0/v'] \land \emptyset[v_1/v'] \land M_\emptyset[v_0, v_1/0.v, 1.v]) \triangleleft \text{wait}\] (wait not quantified)
\[ = \exists v_0, v_1 \cdot ((\emptyset[v_0/v'] \land \emptyset[v_1/v'] \land M_\emptyset[v_0, v_1/0.v, 1.v]) \triangleleft \text{wait}\] (\&-if distr.)
\[ = \exists v_0, v_1 \cdot ((\emptyset[v_0/v'] \land \emptyset[v_1/v'] \land M_\emptyset[v_0, v_1/0.v, 1.v]) \land M_\emptyset[v_0, v_1/0.v, 1.v]) \land (2.1.2 \text{ of } [HH98])
\[ = \exists v_0, v_1 \cdot (\emptyset[v_0/v'] \triangleleft \text{wait} \triangleright P[v_0/v']) \land (\emptyset[v_1/v'] \triangleleft \text{wait} \triangleright Q[v_1/v']) \land M_\emptyset[v_0, v_1/0.v, 1.v] \land \text{distr. of } [\text{]}]
\[ = \exists v_0, v_1 \cdot (\emptyset \triangleleft \text{wait} \triangleright P[v_0/v'] \land \emptyset \triangleleft \text{wait} \triangleright Q[v_1/v'] \land M_\emptyset[v_0, v_1/0.v, 1.v]) \land \text{def. of } R^\theta\)
\[ = \exists v_0, v_1 \cdot R^\theta P[v_0/v'] \land R^\theta Q[v_1/v'] M_\emptyset[v_0, v_1/0.v, 1.v] \land \text{assumption}
\[ = \exists v_0, v_1 \cdot P[v_0/v'] \land Q[v_1/v'] M_\emptyset[v_0, v_1/0.v, 1.v] \land \text{Lemma 22}
\[ = (P \triangleleft Q); M_\emptyset \land \text{def. of } \cap^\theta\)
\[ = P \cap^\theta Q \]

\[ \square \]

Lemma 37 (closure-\( \cap^\theta \)-IOCO1)

\[ \text{IOCO1}(P \cap^\theta Q) = P \cap^\theta Q \text{ provided } P \text{ and } Q \text{ are IOCO1 healthy} \]
Proof.

\[ \text{Lemma 38} \ (\text{closure-}\land \theta \ - \text{ICO2}) \]

\[ \text{IOCO1}(P \land \theta \ Q) = \]
\[ = \text{IOCO1}(P \land \theta \ Q; M_{\land}) \]
\[ = \text{IOCO1}(\exists v_0, v_1 \cdot P[v_0/v'] \land Q[v_1/v'] \land M_{\land}[v_0, v_1/0.v, 1.v]) \]
\[ = \exists v_0, v_1 \cdot P[v_0/v'] \land Q[v_1/v'] \land M_{\land}[v_0, v_1/0.v, 1.v] \land (ok \Rightarrow (wait' \lor ok')) \]
\[ = \exists v_0, v_1 \cdot P[v_0/v'] \land Q[v_1/v'] \land (0.wait \Leftrightarrow 1.wait) \land (wait' = (0.wait \lor 1.wait)) \land (ok' = (0.ok \land 1.ok)) \land M_{\land}[v_0, v_1/0.v, 1.v] \]
\[ = \exists v_0, v_1 \cdot P[v_0/v'] \land Q[v_1/v'] \land M_{\land}[v_0, v_1/0.v, 1.v] \land (wait \Leftrightarrow wait) \land (wait' = (wait \lor wait)) \land (ok' = (ok \land ok)) \land (ok \Rightarrow (ok \lor wait)) \land (ok \Rightarrow (ok \lor wait)) \land (ok \Rightarrow (ok \lor wait)) \land (ok \Rightarrow (ok \lor wait)) \]
\[ = \exists v_0, v_1 \cdot P[v_0/v'] \land Q[v_1/v'] \land M_{\land}[v_0, v_1/0.v, 1.v] \land (0.wait \Leftrightarrow 1.wait) \land (wait' = (0.wait \lor 1.wait)) \land (ok' = (0.ok \land 1.ok)) \land M_{\land}[v_0, v_1/0.v, 1.v] \land (ok \Rightarrow (ok \lor wait)) \land (ok \Rightarrow (ok \lor wait)) \land (ok \Rightarrow (ok \lor wait)) \land (ok \Rightarrow (ok \lor wait)) \]
\[ = \exists v_0, v_1 \cdot P[v_0/v'] \land Q[v_1/v'] \land M_{\land}[v_0, v_1/0.v, 1.v] \land (Q \land (ok \Rightarrow (ok \lor wait'))) \land M_{\land}[v_0, v_1/0.v, 1.v] \land \theta \] (Lemma 28)

\[ \text{Lemma 38} \ (\text{closure-}\land \theta \ - \text{ICO2}) \]

\[ \text{IOCO2}(P \land \theta \ Q) = P \land \theta \ Q \text{ provided } P \text{ and } Q \text{ are } \text{IOCO2} \text{ healthy} \]
Proof.

\[ \text{IOCO2}(P \vdash^\theta Q) = \]
\[ = \text{IOCO2}(\exists \theta_v, \exists \theta_v \cdot P[v_0/v'] \land Q[v_1/v'] \land M_{\theta_v}[v_0, v_1/0.v, 1.v]) \quad \text{(def. of \( \text{IOCO2} \))} \]
\[ = \exists \theta_v, \exists \theta_v \cdot P[v_0/v'] \land Q[v_1/v'] \land M_{\theta_v}[v_0, v_1/0.v, 1.v] \land (\neg \text{wait} \Rightarrow (\neg \text{wait} \Rightarrow (\theta \not\in \text{ref} \Rightarrow \text{quiescence}))) \quad \text{(assumption)} \]
\[ = \exists \theta_v, \exists \theta_v \cdot \text{IOCO2}(P)[v_0/v'] \land \text{IOCO2}(Q)[v_1/v'] \land M_{\theta_v}[v_0, v_1/0.v, 1.v] \land \]
\[ = \exists \theta_v, \exists \theta_v \cdot (\text{ref} \Rightarrow (\theta \not\in \text{ref} \Rightarrow \text{quiescence})) \quad \text{(def. \text{IOCO2} and substitution)} \]
\[ = \exists \theta_v, \exists \theta_v \cdot P[v_0/v'] \land (\neg \text{wait} \Rightarrow (\neg \text{wait} \Rightarrow (\theta \not\in \text{ref} \Rightarrow \text{quiescence}))) \land Q[v_1/v'] \land \]
\[ = \exists \theta_v, \exists \theta_v \cdot (\text{ref} \Rightarrow (\theta \not\in \text{ref} \Rightarrow \text{quiescence})) \land M_{\theta_v}[v_0, v_1/0.v, 1.v] \land \]
\[ = \exists \theta_v, \exists \theta_v \cdot (\text{ref} \Rightarrow (\theta \not\in \text{ref} \Rightarrow \text{quiescence}))) \land \]
\[ = \exists \theta_v, \exists \theta_v \cdot P[v_0/v'] \land \text{IOCO2}(Q)[v_1/v'] \land M_{\theta_v}[v_0, v_1/0.v, 1.v] \land \]
\[ = \text{IOCO2}(P \land Q); M_{\theta_v} \quad \text{(substitution and Lemma 29)} \]
\[ = (P \land Q); M_{\theta_v} \quad \text{(substitution and def. of \text{IOCO2})} \]
\[ = (P \land Q); M_{\theta_v} \quad \text{(def. of \( \vdash^\theta \))} \]
\[ \square \]

Lemma 39 (closure-\( \vdash^\theta \)-IOCO3)

\[ \text{IOCO3}(P \vdash^\theta Q) = P \vdash^\theta Q \quad \text{provided \( P \) and \( Q \) are \text{IOCO3} \text{ healthy}} \]

\(^3\text{The validity of this step was automatically checked using the sat-modulo-theories-solver yices [Int08]. The proof-script can be found at http://www.ist.tugraz.at/staff/weiglhofer}\)
Proof.

\[ \text{IOCO3}(P \ 	riangledown^\theta \ Q) = \] 

(assumption)

\[ \text{IOCO3}(P \wedge Q); M_\triangledown \] 

(Lemma 22)

\[ = \text{IOCO3}(\exists v_0, v_1 \cdot P[v_0/v'] \land Q[v_1/v'] \land M_\triangledown[v_0, v_1/0.v.1.v]) \] 

(def of IOCO3)

\[ = \exists v_0, v_1 \cdot P[v_0/v'] \land Q[v_1/v'] \land M_\triangledown[v_0, v_1/0.v.1.v] \land (\neg wait \Rightarrow (wait' \Rightarrow (\theta \notin ref' \Rightarrow \exists s \cdot tr' - tr = s \cdot \theta'))) \]

(def of \text{IOCO3} and substitution)

\[ = \exists v_0, v_1 \cdot P[v_0/v'] \land \text{IOCO3}(Q)[v_1/v'] \land M_\triangledown[v_0, v_1/0.v.1.v] \land \] 

\[ \neg wait \Rightarrow (wait' \Rightarrow (\theta \notin ref' \Rightarrow \exists s \cdot tr' - tr = s \cdot \theta')) \] 

(prop. calculus\(^4\))

\[ = \exists v_0, v_1 \cdot P[v_0/v'] \land (\neg wait \Rightarrow (wait_0 \Rightarrow (\theta \notin ref_0 \Rightarrow \exists s \cdot tr_0 - tr \in s \cdot \theta'\theta))) \land Q[v_1/v'] \land \] 

(substitution and def. of IOCO3)

\[ = \exists v_0, v_1 \cdot \text{IOCO3}(P)[v_0/v'] \land \text{IOCO3}(Q)[v_1/v'] \land M_\triangledown[v_0, v_1/0.v.1.v] \land \] 

(assumption)

\[ = \exists v_0, v_1 \cdot P[v_0/v'] \land Q[v_1/v'] \land M_\triangledown[v_0, v_1/0.v.1.v] \] 

(lemma)

\[ = (P \wedge Q); M_\triangledown \] 

(def of \triangledown^\theta)

\[ = P \ 	riangledown^\theta \ Q \]

A quiescence preserving external choice operator is given by

**Definition 20 (Quiescence preserving external choice)**

\[ P \ 	riangledown^\theta \ Q =_d (P \wedge Q); M_+ \] with \[ M_+ =_d M_+^\theta < \delta \triangleright M^-\delta^\theta \]

Our external choice’s merge relation is equal to setting the value of \( tr' \) to either \( 0.tr \) or \( 1.tr \) and the merge relation itself, i.e.,

\(^4\)The validity of this step was automatically checked using the sat-modulo-theories-solver yices [Int08]. The proof-script can be found at http://www.ist.tugraz.at/staff/weighhofer
Lemma 41 (tr-\(M_+\))

\[ M_+ = (\text{tr}' = 0.\text{tr} \lor \text{tr}' = 1.\text{tr}) \land M_+ \]

Proof.
\[ M_+ = \]
\[ = M_+^{\theta} < \delta^{\theta} \triangleright M^{-\theta} \]
\[ = (\theta \land M_+^{\text{init}}) < \delta^{\theta} \triangleright (\theta \land M^{\text{term}}) \]
\[ = \left( M^{\theta} \land \left( \begin{array}{c} (\text{ref}' = \ldots) \land \\
\text{tr}' = 0.\text{tr} \lor \text{tr}' = 1.\text{tr} \end{array} \right) \right) < \delta^{\theta} \triangleright \left( M^{\theta} \land \left( \text{tr}' = 0.\text{tr} \land (\text{ref}' = 0.\text{ref}) \lor \\
\text{tr}' = 1.\text{tr} \land (\text{ref}' = 1.\text{ref}) \right) \right) \]  
(prop. calculus)
\[ = \left( M^{\theta} \land \left( \begin{array}{c} (\text{tr}' = 0.\text{tr} \lor \text{tr}' = 1.\text{tr}) \land \\
\text{ref}' = \ldots \land \\
\text{tr}' = 0.\text{tr} \lor \text{tr}' = 1.\text{tr} \end{array} \right) \right) < \delta^{\theta} \triangleright \left( M^{\theta} \land \left( \text{tr}' = 0.\text{tr} \land (\text{ref}' = 0.\text{ref}) \lor \\
\text{tr}' = 1.\text{tr} \land (\text{ref}' = 1.\text{ref}) \right) \right) \]  
(prop. calculus)
\[ = \left( (\text{tr}' = 0.\text{tr} \lor \text{tr}' = 1.\text{tr}) \land \theta \land M^{\text{init}} \right) < \delta^{\theta} \triangleright ((\text{tr}' = 0.\text{tr} \lor \text{tr}' = 1.\text{tr}) \land M^{\theta} \land M^{\text{term}}) \]  
(def. of \(M_+^{\theta}\) and of \(M^{\theta}\))
\[ = \left( (\text{tr}' = 0.\text{tr} \lor \text{tr}' = 1.\text{tr}) \land M_+^{\theta} \right) < \delta^{\theta} \triangleright ((\text{tr}' = 0.\text{tr} \lor \text{tr}' = 1.\text{tr}) \land M^{-\theta}) \]  
(distr. of \(\land\) over if)
\[ = (\text{tr}' = 0.\text{tr} \lor \text{tr}' = 1.\text{tr}) \land (M_+^{\theta} \land \theta) \land M_+ \]  
(def. of \(M_+\))
\[ = (\text{tr}' = 0.\text{tr} \lor \text{tr}' = 1.\text{tr}) \land M_+ \]

Furthermore, the merge relation \(M_+\) is equivalent to having \(\text{wait}' = (0.\text{wait} \land 1.\text{wait})\) and \(\text{ok}' = (0.\text{ok} \land 1.\text{ok})\) in conjunction with the merge relation itself:

Lemma 41 (wait-and-ok-\(M_+\))

\[ M_+ = (0.\text{wait} \equiv 1.\text{wait}) \land (\text{wait}' = (0.\text{wait} \lor 1.\text{wait})) \land (\text{ok}' = (0.\text{ok} \land 1.\text{ok})) \land M_+ \]

Proof.
\[ M_+ = \]
\[ = M_+^{\theta} < \delta^{\theta} \triangleright M^{-\theta} \]
\[ = (\theta \land M_+^{\text{init}}) < \delta^{\theta} \triangleright (\theta \land M^{\text{term}}) \]
\[ = \left( \text{wait} \equiv 1.\text{wait} \land \left( \begin{array}{c} \text{wait}' = (0.\text{wait} \lor 1.\text{wait}) \land \\
\text{ok}' = (0.\text{ok} \land 1.\text{ok}) \end{array} \right) \right) \land \left( M^{\theta} \land M_+^{\text{init}} \right) < \delta^{\theta} \triangleright (\theta \land M^{\text{term}}) \]  
(Lemma 24)
\[ = \left( \text{wait} \equiv 1.\text{wait} \land \left( \begin{array}{c} \text{wait}' = (0.\text{wait} \lor 1.\text{wait}) \land \\
\text{ok}' = (0.\text{ok} \land 1.\text{ok}) \end{array} \right) \right) \land (M^{\theta} \land M_+^{\text{init}}) < \delta^{\theta} \triangleright (\theta \land M^{\text{term}}) \]  
(def. of \(M_+^{\theta}\) and of \(M^{-\theta}\))
\[ = (\text{wait} \equiv 1.\text{wait} \land \left( \begin{array}{c} \text{wait}' = (0.\text{wait} \lor 1.\text{wait}) \land \\
\text{ok}' = (0.\text{ok} \land 1.\text{ok}) \end{array} \right) \right) \land (M^{\theta} \land M_+^{\text{init}}) < \delta^{\theta} \triangleright (\theta \land M^{\text{term}}) \]  
(def. of \(M_+\))
\[ = (\text{wait} \equiv 1.\text{wait} \land (\text{wait}' = (0.\text{wait} \lor 1.\text{wait})) \land (\text{ok}' = (0.\text{ok} \land 1.\text{ok})) \land M_+ \]

As for the internal choice initial merge relation we can extract the calculation of the refusals for \(M_+\) as well.
Lemma 42 (wait-and-ref-$M_+$)

$M_+ = M_+ \land (\text{wait}' = (0.\text{wait} \land 1.\text{wait}))$\land
\[
\left(\begin{array}{c}
tr' = 0.tr \land ref'_\text{in} = 0.ref_\text{in} \land ref'_\text{out} = 0.ref_\text{out} \\
ref' = ((0.ref \land 1.ref) \setminus \{\theta\}) \cup (\{\theta\} \land (0.ref \cup 1.ref))\lor \\
tr' = 1.tr \land ref'_\text{in} = 1.ref_\text{in} \land ref'_\text{out} = 1.ref_\text{out} \\
ref' = ((0.ref \land 1.ref) \setminus \{\theta\}) \cup (\{\theta\} \land (0.ref \cup 1.ref))
\end{array}\right) < \delta^2 >
\]
\[
\left(\begin{array}{c}
tr' = 0.tr \land ref' = 0.ref \\
ref'_\text{in} = 0.ref'_\text{in} \land ref'_\text{out} = 0.ref'_\text{out} \lor \\
tr' = 1.tr \land ref' = 1.ref \\
ref'_\text{in} = 1.ref'_\text{in} \land ref'_\text{out} = 1.ref'_\text{out}
\end{array}\right)
\]
Lemma 43 (symmetric-M+)

\[ M_+[0,v,1,v/1,v,0,v] = M_+ \]
Definition 21 (External choice merge relation)

Lemma 45 (the relational skip beginning of $P$)

Proof.

\[
M_+ [0.v, 1.v/1.v, 0.v] = \quad \text{(def. of $M_+$)}
\]

\[
= (M_+^{\delta^g} \circ \delta^g \triangleright M^{-\delta^g}) [0.v, 1.v/1.v, 0.v] \quad \text{(no 1.v and 0.v in $\delta^g$)}
\]

\[
= M_+^{\delta^g} [0.v, 1.v/1.v, 0.v] \quad \text{(Lemma 25)}
\]

\[
= M_+^{\delta^g} \circ \delta^g \triangleright M^{-\delta^g} [0.v, 1.v/1.v, 0.v] \quad \text{(Lemma 26)}
\]

\[
= M_+^{\delta^g} \circ \delta^g \triangleright M^{-\delta^g}
\]

\[
= M_+
\]

\[
\square
\]

Except the merge relation $M_+^{\delta^g}$ the external choice is equivalent $\cap^\theta$. The difference is how the very beginning of $P$ and $Q$ is combined to form $P \ +^\theta Q$.

Definition 21 (External choice merge relation)

\[
M_+^{\delta^g} = M^{\theta} \land M^{\text{init}}
\]

\[
M^{\text{init}} = \theta \land (\text{ref}' = ((0.\text{ref} \cap 1.\text{ref}) \setminus \{\theta\}) \cup (\{\theta\} \cap (0.\text{ref} \cup 1.\text{ref}))) \land (tr' = 0.tr \lor tr' = 1.tr)
\]

The merge relation $M_+^{\delta^g}$ is symmetric in its variables, i.e. in $0.v = \{0.wait, 0.ok, 0.\text{ref}, 0.tr\}$ and in $1.v = \{1.wait, 1.ok, 1.\text{ref}, 1.tr\}$:

Lemma 44 (symmetric-$M_+^{\delta^g}$)

\[
M_+^{\delta^g} [0.v, 1.v/1.v, 0.v] = M_+^{\delta^g}
\]

Proof.

\[
M_+^{\delta^g} [0.v, 1.v/1.v, 0.v] = \quad \text{(def. of $M_+^{\delta^g}$)}
\]

\[
= (M^{\theta} \land M^{\text{init}}) [0.v, 1.v/1.v, 0.v] \quad \text{(def. of $M_+^{\text{init}}$)}
\]

\[
= \left( M^{\theta} \land \left( (\text{ref}' = ((0.\text{ref} \cap 1.\text{ref}) \setminus \{\theta\}) \cup (\{\theta\} \cap (0.\text{ref} \cup 1.\text{ref}))) \land (tr' = 0.tr \lor tr' = 1.tr) \right) \right) [0.v, 1.v/1.v, 0.v] \quad \text{(def. of $[]$)}
\]

\[
= M^{\theta} [0.v, 1.v/1.v, 0.v] \land \left( (\text{ref}' = ((1.\text{ref} \cap 0.\text{ref}) \setminus \{\theta\}) \cup (\{\theta\} \cap (1.\text{ref} \cup 0.\text{ref}))) \land (tr' = 1.tr \lor tr' = 0.tr) \right) \quad \text{(Lemma 23)}
\]

\[
= M^{\theta} \land \left( (\text{ref}' = ((1.\text{ref} \cap 0.\text{ref}) \setminus \{\theta\}) \cup (\{\theta\} \cap (1.\text{ref} \cup 0.\text{ref}))) \land (tr' = 1.tr \lor tr' = 0.tr) \right) \quad \text{(symmetry of $\cap$, $\lor$)}
\]

\[
= M^{\theta} \land \left( (\text{ref}' = ((0.\text{ref} \cap 1.\text{ref}) \setminus \{\theta\}) \cup (\{\theta\} \cap (0.\text{ref} \cup 1.\text{ref}))) \land (tr' = 0.tr \lor tr' = 1.tr) \right) \quad \text{(def. of $M_+^{\text{init}}$)}
\]

\[
= M^{\theta} \land M^{\text{init}} \quad \text{(def. of $M_+^{\delta^g}$)}
\]

\[
= M_+^{\delta^g}
\]

\[
\square
\]

As $M_\cap$ $M_+$ does not change anything if the values for $0.v$ and $1.v$ in $M_\cap$ are the same, i.e. it reduces to the relational skip $\mathbb{I}_{rel}$.

Lemma 45 ($M_+\text{-reduces-to-skip}$)

\[
M_+ [v_0, v_0/1.v, 0.v] = \mathbb{I}_{rel}[v_0/v]
\]
Proof.

\[
M_+[v_0,v_0/0.v,1.v] = (M_+^g < \delta^g \triangleright M_+^g)[v_0,v_0/0.v,1.v] \quad \text{(def. of } M_+) \\
= (M^g \land M_+^{init} < \delta^g \triangleright M^g \land M^{term})[v_0,v_0/0.v,1.v] \quad \text{(def. of } M_+^g \text{ and of } M^{-\delta^g}) \\
= \begin{pmatrix}
(0.wait \Leftrightarrow 1.wait) \land wait' = 0.wait \land ok' = (0.ok \land 1.ok)\land \\
(\neg initQuiet(tr_0 - tr) \lor \neg initQuiet(tr_0 - tr) \Rightarrow \neg initQuiet(tr' - tr))\land \\
(tr' = tr_0 \lor tr' = tr_0) \land ref' = ((ref_0 \cap ref_0) \land \emptyset) \cup ((\emptyset \cap (ref_0 \cup ref_0)))
\end{pmatrix} \triangleright \delta^g[v_0,v_0/0.v,1.v] \\
\quad \text{(def. of [])}
\]

\[
= \begin{pmatrix}
wait_0 \Leftrightarrow wait_0) \land wait' = wait_0 \land ok' = (0.ok \land 0.ok)\land \\
(\neg initQuiet(tr_0 - tr) \lor \neg initQuiet(tr_0 - tr) \Rightarrow \neg initQuiet(tr' - tr))\land \\
(tr' = tr_0 \lor tr' = tr_0) \land ref' = ((ref_0 \cap ref_0) \land \emptyset) \cup ((\emptyset \cap (ref_0 \cup ref_0)))
\end{pmatrix} \triangleright \delta^g[v_0,v_0/0.v,1.v] \\
\quad \text{(prop. calculus)}
\]

\[
= \begin{pmatrix}
wait' = wait_0 \land ok' = 0.ok\land \\
(\neg initQuiet(tr_0 - tr) \Rightarrow \neg initQuiet(tr' - tr))\land \\
(tr' = tr_0 \land ref' = ref_0)
\end{pmatrix} \triangleright \delta^g[v_0,v_0/0.v,1.v] \\
\quad \text{(cond. idemp)}
\]

\[
= wait' = wait_0 \land ok' = 0.ok \land (\neg initQuiet(tr_0 - tr) \Rightarrow \neg initQuiet(tr' - tr)) \land (tr' = tr_0 \land ref' = ref_0) \\
= wait' = wait_0 \land ok' = 0.ok \land (\neg initQuiet(tr_0 - tr) \Rightarrow \neg initQuiet(tr_0 - tr)) \land (tr' = tr_0 \land ref' = ref_0) \quad \text{(def. of } \llbracket \rrbracket_{def}) \\
= wait' = wait_0 \land ok' = 0.ok \land tr' = tr_0 \land ref' = ref_0 \quad \text{(def. of } \llbracket \rrbracket_{def}) \\
\]

Also, our quiescence preserving external choices +^g obey the laws for quiescent preserving internal choices.

**Lemma 46 (idempotent-+^g)**

\[
P \ +^g \ P = P
\]
Proof.

\[ P +^\theta P = \]  
\[ = (P \triangleleft P) ; M_+ \]  
(lem. of \( +^\theta \))

\[ = \exists v_0, v_1 \bullet P[v_0/v'] \land P[v_1/v] \land M_+[v_0, v_1/0.v, 1.v] \]  
(rename variables)

\[ = \exists v_0 \bullet P[v_0/v'] \land P[v_0/v] \land M_+[v_0, v_0/0.v, 1.v] \]  
(prop. calculus)

\[ = \exists v_0 \bullet P[v_0/v'] \land M_+[v_0, v_0/0.v, 1.v] \]  
(lem. of \( +^\theta \))

\[ = P ; I_{rel} \]  
(\( ; \) unit)

\[ = P \]

Lemma 47 (commutative-\(+^\theta\))

\[ P +^\theta Q = Q +^\theta P \]

Proof.

\[ P +^\theta Q = \]  
\[ = (P \triangleleft Q) ; M_+ \]  
(lem. of \( +^\theta \))

\[ = \exists v_0, v_1 \bullet P[v_0/v'] \land Q[v_1/v'] \land M_+[v_0, v_1/0.v, 1.v] \]  
(rename variables)

\[ = \exists v_0, v_1 \bullet P[v_1/v'] \land Q[v_1/v'] \land M_+[v_1, v_0/0.v, 1.v] \]  
(lem. of \( +^\theta \))

\[ = \exists v_0, v_1 \bullet P[v_1/v'] \land Q[v_1/v'] \land (M_+[0.v, 1.v/1.v, 0.v])[v_1, v_0/0.v, 1.v] \]  
(prop. calculus)

\[ = \exists v_0, v_1 \bullet Q[v_0/v'] \land P[v_1/v'] \land M_+[v_0, v_1/0.v, 1.v] \]  
(lem. of \( +^\theta \))

\[ = (Q \triangleleft P) ; M_+ \]  
(lem. of \( +^\theta \))

\[ = Q +^\theta P \]

Our healthiness conditions are preserved by \( +^\theta \).

Lemma 48 (closure-\(+^\theta\)-R1)

\[ R_1(P +^\theta Q) = P +^\theta Q \text{ provided } P \text{ and } Q \text{ are } R_1 \text{ healthy} \]
Proof.

R1(P +^θ Q) = (assumption)
= R1(R1(P) +^θ R1(Q)) = (def. of R1)
= ((P ∧ (tr ≤ tr')) +^θ (Q ∧ (tr ≤ tr'))) ∧ (tr ≤ tr')
= (((P ∧ (tr ≤ tr')) ∧ (Q ∧ (tr ≤ tr'))) ∨ (P ∧ (tr ≤ tr')) ∧ M+) ∧ (tr ≤ tr')
= (∃v0, v1 • (P ∧ (tr ≤ tr'))[v0/v', (Q ∧ (tr ≤ tr'))[v1/v'] ∧ M+[v0, v1/0.v, 1.v]] ∧ (tr ≤ tr')
= (∃v0, v1 • P[v0/v'] ∧ (tr ≤ tr0) ∧ Q[v1/v'] ∧ (tr ≤ tr1) ∧ M+[v0, v1/0.v, 1.v]) ∧ (tr ≤ tr')
= (∃v0, v1 • P[v0/v'] ∧ (tr ≤ tr0) ∧ Q[v1/v'] ∧ (tr ≤ tr1) ∧ M+[v0, v1/0.v, 1.v]) ∧ (tr ≤ tr')
= (∃v0, v1 • P[v0/v'] ∧ (tr ≤ tr0) ∧ Q[v1/v'] [v0/v0.1.v] ∧ M+[v1,v1/0.v,v1.v] ∧ (tr ≤ tr1) ∧
M+[v0,v1/0.v,1.v] ∧ (tr' = tr0 ∨ tr' = tr1) ∧ (tr ≤ tr')
= (∃v0, v1 • P[v0/v'] ∧ Q[v1/v'] ∧ M+[v0,v1/0.v,1.v] ∧ (tr ≤ tr0) ∧ (tr ≤ tr1) ∧ (tr' = tr0) ∧ (tr' ≤ tr1) ∧ (dr: tr' ≤ tr1) ∧ (tr ≤ tr')
= ∃v0, v1 • P[v0/v'] ∧ Q[v1/v'] ∧ M+[v0,v1/0.v,1.v] ∧ (tr ≤ tr0) ∧ (tr ≤ tr1) ∧ (tr = tr0) ∧ (dr: tr' ≤ tr1) ∧ (tr ≤ tr')
= ∃v0, v1 • P[v0/v'] ∧ Q[v1/v'] ∧ M+[v0,v1/0.v,1.v] ∧ (tr ≤ tr0) ∧ (tr ≤ tr1) ∧ (dr: tr' = tr0) ∧ (tr' ≤ tr1) ∧ (tr ≤ tr')
= ∃v0, v1 • (P ∧ (tr ≤ tr'))[v0/v'] ∧ (Q ∧ (tr ≤ tr'))[v1/v'] ∧ M+[v0,v1/0.v,1.v]
= ∃v0, v1 • (R1(P))[v0/v'] ∧ (R1(Q))[v1/v'] ∧ M+[v0,v1/0.v,1.v]
= ∃v0, v1 • P[v0/v'] ∧ Q[v1/v'] ∧ M+[v0,v1/0.v,1.v]
= (P +^θ Q); M+

□

Lemma 49 (closure-+^θ-R2)

R2(P +^θ Q) = P +^θ Q provided P and Q are R2 healthy
Proof.

\[ R2(P(tr, tr') +^0 Q(tr, tr')) = \]

(\text{def. of } R2)

\[ = (P(tr, tr') +^0 Q(tr, tr'))(\emptyset, tr' - tr) \]

(\text{def. of } +^0)

\[ = ((P(tr, tr') \land Q(tr, tr')); M_+(tr, tr'))(\emptyset, tr' - tr) \]

(Lemma 22)

\[ = (\exists v_1 \cdot P(tr, tr')[v_0/v] \land Q(tr, tr')[v_0/v'] \land M_+[v_0, v_1/0.v, 1.v](tr, tr'))(\emptyset, tr' - tr) \]

(assumption)

\[ = (\exists v_1 \cdot R2(P(tr, tr'))[v_0/v'] \land R2(Q(tr, tr'))[v_1/v'] \land M_+[v_0, v_1/0.v, 1.v](tr, tr'))(\emptyset, tr' - tr) \]

(def. of R2)

\[ = (\exists v_1 \cdot P(\emptyset, tr' - tr)[v_0/v'] \land Q(\emptyset, tr' - tr)[v_1/v'] \land M_+[v_0, v_1/0.v, 1.v](tr, tr'))(\emptyset, tr' - tr) \]

(substitution)

\[ = (\exists v_1 \cdot P(v_0/v')[\emptyset, tr_0 - tr] \land Q(v_0/v')[\emptyset, tr_1 - tr] + M_+[v_0, v_1/0.v, 1.v](tr, tr'))(\emptyset, tr' - tr) \]

{def. of } M_+ \text{ and substitution)

\[ = \exists v_1 \cdot P(v_0/v')[\emptyset, tr_0 - tr] \land Q(v_0/v')[\emptyset, tr_1 - tr] \land \]

\[ \left( \left( \begin{array}{c}
\left( \begin{array}{c}
\text{wait}_0 \leftrightarrow \text{wait}_1 \land \text{wait}' = \text{wait}_0 \land (ok' = (ok_0 \land ok_1)) \\
\neg \text{initQuiet}(tr_0 - tr) \vee \neg \text{initQuiet}(tr_1 - tr) \Rightarrow \neg \text{initQuiet}(tr' - tr)\end{array} \right)
\end{array} \right) < \delta^0 \right) \]

\[ \left( \begin{array}{c}
\left( \begin{array}{c}
\text{initQuiet}(tr_0 - tr) \lor \neg \text{initQuiet}(tr_1 - tr) \Rightarrow \neg \text{initQuiet}(tr_0 - tr)\end{array} \right) \\
\neg \text{initQuiet}(tr_1 - tr) \lor \neg \text{initQuiet}(tr_1 - tr) \Rightarrow \neg \text{initQuiet}(tr_1 - tr)\end{array} \right) \]

\[ \left( \begin{array}{c}
\left( \begin{array}{c}
\text{initQuiet}(tr_0 - tr) \lor \neg \text{initQuiet}(tr_1 - tr) \Rightarrow \neg \text{initQuiet}(tr_0 - tr)\end{array} \right) \\
\neg \text{initQuiet}(tr_1 - tr) \lor \neg \text{initQuiet}(tr_1 - tr) \Rightarrow \neg \text{initQuiet}(tr_1 - tr)\end{array} \right) \]

{substitution)

\[ = \exists v_1 \cdot P(v_0/v')[\emptyset, tr_0 - tr] \land Q(v_0/v')[\emptyset, tr_1 - tr] \land \]

\[ \left( \begin{array}{c}
\left( \begin{array}{c}
\text{wait}_0 \leftrightarrow \text{wait}_1 \land \text{wait}' = \text{wait}_0 \land (ok' = (ok_0 \land ok_1)) \\
\neg \text{initQuiet}(tr_0 - tr) \lor \neg \text{initQuiet}(tr_1 - tr) \Rightarrow \neg \text{initQuiet}(tr_0 - tr)\end{array} \right)
\end{array} \right) < \delta^0 \right) \]

\[ \left( \begin{array}{c}
\left( \begin{array}{c}
\text{initQuiet}(tr_0 - tr) \lor \neg \text{initQuiet}(tr_1 - tr) \Rightarrow \neg \text{initQuiet}(tr_0 - tr)\end{array} \right) \\
\neg \text{initQuiet}(tr_1 - tr) \lor \neg \text{initQuiet}(tr_1 - tr) \Rightarrow \neg \text{initQuiet}(tr_1 - tr)\end{array} \right) \]

{prop. calculus)

\[ = \exists v_1 \cdot P(v_0/v')[\emptyset, tr_0 - tr] \land Q(v_0/v')[\emptyset, tr_1 - tr] \land \]

\[ \left( \begin{array}{c}
\left( \begin{array}{c}
\text{wait}_0 \leftrightarrow \text{wait}_1 \land \text{wait}' = \text{wait}_0 \land (ok' = (ok_0 \land ok_1)) \\
\neg \text{initQuiet}(tr_0 - tr) \lor \neg \text{initQuiet}(tr_1 - tr) \Rightarrow \neg \text{initQuiet}(tr_0 - tr)\end{array} \right)
\end{array} \right) < \delta^0 \right) \]

\[ \left( \begin{array}{c}
\left( \begin{array}{c}
\text{initQuiet}(tr_0 - tr) \lor \neg \text{initQuiet}(tr_1 - tr) \Rightarrow \neg \text{initQuiet}(tr_0 - tr)\end{array} \right) \\
\neg \text{initQuiet}(tr_1 - tr) \lor \neg \text{initQuiet}(tr_1 - tr) \Rightarrow \neg \text{initQuiet}(tr_1 - tr)\end{array} \right) \]

{def. of } M_+ \text{ and substitution)

\[ = \exists v_1 \cdot P(v_0/v')[\emptyset, tr_0 - tr] \land Q(v_0/v')[\emptyset, tr_1 - tr] + M_+[v_0, v_1/0.v, 1.v] \]

(substitution)

\[ = \exists v_1 \cdot (P(\emptyset, tr' - tr)[v_0/v] \land Q(\emptyset, tr' - tr)[v_1/v] \land M_+[v_0, v_1/0.v, 1.v]) \]

(assumption)

\[ = \exists v_1 \cdot R2(P(tr, tr')[v_0/v'] \land R2(Q(tr, tr'))[v_1/v'] \land M_+[v_0, v_1/0.v, 1.v]) \]

(Lemma 22)

\[ = (P(tr, tr') \land Q(tr, tr')); M_+[v_0, v_1/0.v, 1.v] \]

(Lemma 22)

\[ = (P(tr, tr') \land Q(tr, tr')); M_+[v_0, v_1/0.v, 1.v] \]

{def. of } +^0

\[ = P(tr, tr') +^0 Q(tr, tr') \]

□

Lemma 50 (closure-+^0-R3^0)

\[ R3^0(P +^0 Q) = P +^0 Q \text{ provided } P \text{ and } Q \text{ are } R3^0 \text{ healthy} \]
Proof.

\[ R^\theta(P + ^\theta Q) = \]
\[ = \{ ^\theta \triangleleft \text{wait} \triangleright (P + ^\theta Q) \} \] (def. of \( R^\theta \))
\[ = (\{ ^\theta \triangleleft \text{wait} \triangleright (P + ^\theta Q) \} \cup M_+ \triangleleft \text{wait} \triangleright (P \triangleleft Q)) \] (def. of \(+^\theta\))
\[ = (\{ ^\theta \triangleleft \text{wait} \triangleright (P \triangleleft Q) \} \cup M_+ \triangleleft \text{wait} \triangleright (P + ^\theta Q)) \] (Lemma 22)
\[ = (\exists v_0, v_1 \bullet \{ ^\theta[v_0/v'] \wedge \{ ^\theta[v_1/v'] \wedge M_+[v_0, v_1/0.v, 1.v]\} \triangleleft \text{wait} \triangleright (P + ^\theta Q) \} \] (wait not quantified)
\[ = \exists v_0, v_1 \bullet ((\{ ^\theta[v_0/v'] \wedge \{ ^\theta[v_1/v'] \wedge M_+[v_0, v_1/0.v, 1.v]\} \triangleleft \text{wait} \triangleright (P + ^\theta Q)) \] (\&-if distr.)
\[ = \exists v_0, v_1 \bullet ((\{ ^\theta[v_0/v'] \wedge \{ ^\theta[v_1/v'] \wedge M_+[v_0, v_1/0.v, 1.v]\} \triangleleft \text{wait} \triangleright (P + ^\theta Q)) \] (2.1.2 of \([HH98]\))
\[ = \exists v_0, v_1 \bullet (\{ ^\theta \triangleleft \text{wait} \triangleright P[v_0/v'] \wedge (\{ ^\theta \triangleleft \text{wait} \triangleright Q[v_1/v'] \wedge M_+[v_0, v_1/0.v, 1.v]\) \] (distr. of [])
\[ = \exists v_0, v_1 \bullet (\{ ^\theta \triangleleft \text{wait} \triangleright P[v_0/v'] \wedge (\{ ^\theta \triangleleft \text{wait} \triangleright Q[v_1/v'] \wedge M_+[v_0, v_1/0.v, 1.v]\) \] (def. of \( R^\theta \))
\[ = \exists v_0, v_1 \bullet R^\theta P[v_0/v'] \wedge R^\theta(Q)[v_1/v'] M_+[v_0, v_1/0.v, 1.v] \] (assumption)
\[ = \exists v_0, v_1 \bullet P[v_0/v'] \wedge Q[v_1/v'] M_+[v_0, v_1/0.v, 1.v] \] (Lemma 22)
\[ = (P \triangleleft Q) ; M_+ \] (def. of \(+^\theta\))
\[ = P + ^\theta Q \]

\[ \square \]

Lemma 51 (closure-\(+^\theta\)-IOCO1)

\[ IOCO1(P + ^\theta Q) = P + ^\theta Q \] provided \( P \) and \( Q \) are \( IOCO1 \) healthy
Lemma 52 (closure-\(+^\theta\)-IOCO2)

\[ \text{IOCO2}(P +^\theta Q) = P +^\theta Q \] provided \(P\) and \(Q\) are \textbf{IOCO2} healthy
Proof.

\[ \text{ICO2}(P \, +^\theta \, Q) = \]

\[ = \text{ICO2}((P \land Q); M_+) \quad \text{(def. of } +^\theta \text{)} \]

\[ = \text{ICO2}(\exists v_0, v_1 \cdot P[v_0/v'] \land Q[v_1/v'] \land M_+[v_0, v_1/0.v, 1.v]) \quad \text{(Lemma 22)} \]

\[ = \exists v_0, v_1 \cdot \text{ICO2}(P)[v_0/v'] \land \text{ICO2}(Q)[v_1/v'] \land M_+[v_0, v_1/0.v, 1.v] \land (-\text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \text{quiescence}))) \]

\[ (\text{assumption}) \]

\[ = \exists v_0, v_1 \cdot \text{ICO2}(P)[v_0/v'] \land \text{ICO2}(Q)[v_1/v'] \land M_+[v_0, v_1/0.v, 1.v] \land \\
\quad (-\text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \text{quiescence}))) \quad \text{(def. ICO2 and substitution)} \]

\[ = \exists v_0, v_1 \cdot \text{ICO2}(P)[v_0/v'] \land \text{ICO2}(Q)[v_1/v'] \land M_+[v_0, v_1/0.v, 1.v] \land \\
\quad (-\text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \text{quiescence}))) \quad \text{(def. ICO2 and substitution)} \]

\[ = \exists v_0, v_1 \cdot \text{ICO2}(P)[v_0/v'] \land \text{ICO2}(Q)[v_1/v'] \land M_+[v_0, v_1/0.v, 1.v] \land \\
\quad (\text{wait}' = (\text{wait}_0 \land \text{wait}_1) \land \text{tr}' = \text{tr}_0 \land \cdots \land \text{tr}_n \land \text{ref}'_\text{in} = \text{ref}'_\text{in} \land \text{tr}' = \text{tr}_1 \land \cdots \land \text{tr}_n \land \text{ref}'_\text{in} = \text{ref}'_\text{in}) \]

\[ \quad \text{(prop. calculus)} \]

\[ = \exists v_0, v_1 \cdot \text{ICO2}(P)[v_0/v'] \land \text{ICO2}(Q)[v_1/v'] \land M_+[v_0, v_1/0.v, 1.v] \land \\
\quad (\text{wait}' = (\text{wait}_0 \land \text{wait}_1) \land \text{tr}' = \text{tr}_0 \land \cdots \land \text{tr}_n \land \text{ref}'_\text{in} = \text{ref}'_\text{in} \land \text{tr}' = \text{tr}_1 \land \cdots \land \text{tr}_n \land \text{ref}'_\text{in} = \text{ref}'_\text{in}) \]

\[ \quad \text{(substitution and Lemma 42)} \]

\[ = \exists v_0, v_1 \cdot \text{ICO2}(P)[v_0/v'] \land \text{ICO2}(Q)[v_1/v'] \land M_+[v_0, v_1/0.v, 1.v] \land \\
\quad (\text{wait}' = (\text{wait}_0 \land \text{wait}_1) \land \text{tr}' = \text{tr}_0 \land \cdots \land \text{tr}_n \land \text{ref}'_\text{in} = \text{ref}'_\text{in} \land \text{tr}' = \text{tr}_1 \land \cdots \land \text{tr}_n \land \text{ref}'_\text{in} = \text{ref}'_\text{in}) \]

\[ \quad \text{(substitution and def. of ICO2)} \]

\[ = \exists v_0, v_1 \cdot \text{ICO2}(P)[v_0/v'] \land \text{ICO2}(Q)[v_1/v'] \land M_+[v_0, v_1/0.v, 1.v] \land \\
\quad (\text{wait}' = (\text{wait}_0 \land \text{wait}_1) \land \text{tr}' = \text{tr}_0 \land \cdots \land \text{tr}_n \land \text{ref}'_\text{in} = \text{ref}'_\text{in} \land \text{tr}' = \text{tr}_1 \land \cdots \land \text{tr}_n \land \text{ref}'_\text{in} = \text{ref}'_\text{in}) \]

\[ \quad \text{(assumption)} \]

\[ = \exists v_0, v_1 \cdot \text{ICO2}(P)[v_0/v'] \land \text{ICO2}(Q)[v_1/v'] \land M_+[v_0, v_1/0.v, 1.v] \land \\
\quad (P \land Q); M_+ \quad \text{(Lemma 22)} \]

\[ = (P \land Q); M_+ \quad \text{(def. of } +^\theta \text{)} \]

\[ = P \, +^\theta \, Q \]

\[ \Box \]

Lemma 53 (closure-\( +^\theta \text{-ICO3})

\[ \text{ICO3}(P \, +^\theta \, Q) = P \, +^\theta \, Q \quad \text{provided P and Q are ICO3 healthy} \]

\[ ^5 \text{The validity of this step was automatically checked using the sat-modulo-theories-solver yices [Int08]. The proof-script can be found at http://www.ist.tugraz.at/staff/weighhofer} \]
Proof.

\( \text{IOCO3}(P +^\delta Q) = \) (def. of +^\delta)

\( = \text{IOCO3}((P \cup Q); M_+) \) (Lemma 22)

\( = \text{IOCO3}(\exists v_0, v_1 \cdot P[v_0/v'] \cup Q[v_1/v'] \cup M_+[v_0, v_1/0.v, 1.v]) \) (def. of IOCO3)

\( = \exists v_0, v_1 \cdot P[v_0/v'] \cup Q[v_1/v'] \cup M_+[v_0, v_1/0.v, 1.v] \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \exists s \cdot tr' - tr = s \theta^*)))) \) (assumption)

\( = \exists v_0, v_1 \cdot \text{IOCO3}(P)[v_0/v'] \cup \text{IOCO3}(Q)[v_1/v'] \cup M_+[v_0, v_1/0.v, 1.v] \land \)

\( (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \exists s \cdot tr' - tr = s \theta^*)))) \) (def. IOCO3 and substitution)

\( = \exists v_0, v_1 \cdot P[v_0/v'] \cup (\neg \text{wait} \Rightarrow (\text{wait}_0 \Rightarrow (\theta \not\in \text{ref}_0 \Rightarrow \exists s \cdot tr_0 - tr = s \theta^*)))) \cup Q[v_1/v'] \land \)

\( (\neg \text{wait} \Rightarrow (\text{wait}_1 \Rightarrow (\theta \not\in \text{ref}_1 \Rightarrow \exists s \cdot tr_1 - tr = s \theta^*)))) \cup M_+[v_0, v_1/0.v, 1.v] \land \)

\( (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \exists s \cdot tr' - tr = s \theta^*))) \) (Lemma 42 and substitution)

\( = \exists v_0, v_1 \cdot P[v_0/v'] \cup (\neg \text{wait} \Rightarrow (\text{wait}_0 \Rightarrow (\theta \not\in \text{ref}_0 \Rightarrow \exists s \cdot tr_0 - tr = s \theta^*)))) \cup Q[v_1/v'] \land \)

\( (\neg \text{wait} \Rightarrow (\text{wait}_1 \Rightarrow (\theta \not\in \text{ref}_1 \Rightarrow \exists s \cdot tr_1 - tr = s \theta^*)))) \cup M_+[v_0, v_1/0.v, 1.v] \land \)

\( (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \not\in \text{ref}' \Rightarrow \exists s \cdot tr' - tr = s \theta^*))) \)

\( = \exists v_0, v_1 \cdot \text{IOCO3}(P)[v_0/v'] \cup \text{IOCO3}(Q)[v_1/v'] \cup M_+[v_0, v_1/0.v, 1.v] \) (substitution and def. of IOCO3)

\( = \exists v_0, v_1 \cdot P[v_0/v'] \cup Q[v_1/v'] \cup M_+[v_0, v_1/0.v, 1.v] \) (assumption)

\( = (P \cup Q); M_+ \) (Lemma 22)

\( = P +^\delta Q \) (def. of +^\delta)

\( \square \)

Specification processes for the io\(co\) framework are defined as follows:

Definition 22 (io\(co\) specification) An io\(co\) specification is a reactive process satisfying the healthiness conditions IOCO1, IOCO2 and IOCO3. In addition its set of possible events is partitioned into the quiescent event, input events, and output events: \( A = A_{\text{out}} \cup A_{\text{in}} \cup \{ \theta \} \) where \( A_{\text{out}} \cap A_{\text{in}} = \emptyset \) and \( \theta \not\in A_{\text{out}} \cup A_{\text{in}} \)

Processes expressed in terms of \( d\theta_{\Delta A}; +^\delta \) and \( \cap^\theta \) are io\(co\) specifications.

Theorem 1 The set of io\(co\) specifications is a \(\{; +^\delta, \cap^\theta\}\)-closure.

---

6The validity of this step was automatically checked using the sat-modulo-theories-solver yices [Int08]. The proof-script can be found at http://www.ist.tugraz.at/staff/weiglhofer
Proof. Idempotence of healthiness conditions and healthiness conditions are preserved (see lemmas).

Remark 1 The class of labeled transition systems (LTS) used for the \textit{ioco} relation is restricted to image finite LTSs [Tre08]. Image finite LTSs are limited in their possible non-deterministic choices, i.e. image finite LTSs are bounded in terms of non-determinism. This requirement is only due to the properties of Tretmans’ test case generation algorithm. Since we are interested in a predicative semantics we do not face the problem of image-finiteness.

Remark 2 The \textit{tgv} tool [JJ05], which claims to generate test cases with respect to \textit{ioco}, uses a different notion of quiesence: \textit{quiesence}_{\textit{TGV}} = \textit{quiesence} \lor (\neg \textit{ok}' \land \neg \textit{wait}') . Note that we rely on \textit{quiesence} rather than \textit{quiescence}_{\textit{TGV}}.

3.3 IOCO Implementations

The input output conformance relation uses labeled transition systems to represent implementations. As mentioned in Section 2, it is not assumed that this LTS is known in advance, but only its existence is required. Our formalization requires something similar: implementations can be expressed as processes.

Processes for representing implementations in terms of the \textit{ioco} relation need to satisfy the properties of specifications plus three additional properties: some restrictions on allowed choices\(^7\), input-enabledness\(^8\), and fairness\(^9\).

Restrictions on choices. An implementation is not allowed to freely choose between the actions enabled in a particular state. The \textit{ioco} relation distinguishes between inputs and outputs not only by partitioning a process’ alphabet, but also by assigning responsibilities to these two alphabets (see Section 3.1.

In terms of choices this means that for implementations choices between outputs are internal choices. Internal choices are represented by a disjunctions over the refused actions, thus we restrict the choices between outputs:

\[
\text{IOCO4} \quad P = P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|A_{\text{out}}| - 1) \leq |\text{ref}_{\text{out}}'|))
\]

Example 6 Because of this healthiness condition the \(\tau\) transitions in state \(z_4\) of Figure 3 are required, i.e. the choice between \(!t\) and \(!c\) is an internal choice. Thus, as required by IOCO4, after the trace \((?1, \theta, ?1)\) \(z\) does not offer \(!t\) and \(!c\) for communication, i.e. \(!t \notin \text{ref'} \land \!c \notin \text{ref'}\). Instead, it non-deterministically offers only one of the two actions for communication, i.e. \(!t \notin \text{ref'} \lor \!c \notin \text{ref'}\).

Contrary, input actions are under control of the system’s environment. That is, choices between inputs are external choices. This restriction is enforced by requiring input-enabledness (see IOCO5).

In addition, if there are inputs and outputs enabled in a particular state of an implementation the choice between input and output is up to the environment. That is, choices between inputs and outputs are external choices. Again, this restriction is covered by having input-enabled implementations, i.e. \(\text{ref}'_{\text{in}} = \emptyset\) (see IOCO5). However, as identified in [PYH03], an external choice between input and output allows the environment to prevent the system from providing an output. That is, whenever the implementation can choose between providing an output or waiting for a stimulus, the choice is controlled by the environment.

Remark 3 Recently, the constraints on the semantics of choices within implementations have been relaxed [Tre08]. By changing the properties of test cases (see Remark 4 in Section 3.5), choices between inputs and outputs are now choices of the implementation. Our work focuses on the original definition of \textit{ioco}.

\(^7\)2nd column on page 104 of [Tre96]  \(^8\)1st column on page 107 of [Tre96]  \(^9\)2nd column on page 115 of [Tre96]
The healthiness condition restricting the allowed choices of implementations, i.e., IOCO4 is idempotent.

Lemma 54 (IOCO4-idempotent)

\[ \text{IOCO4} \circ \text{IOCO4} = \text{IOCO4} \]

Proof.

\[
\begin{align*}
\text{IOCO4}(\text{IOCO4}(P)) &= \text{IOCO4}(P) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|A_{out}| - 1) \leq |ref'_{out}|)) \\
&= P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|A_{out}| - 1) \leq |ref'_{out}|)) \land \\
& \quad (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|A_{out}| - 1) \leq |ref'_{out}|)) \\
&= P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|A_{out}| - 1) \leq |ref'_{out}|)) \\
&= \text{IOCO4}(P)
\end{align*}
\]

Furthermore, IOCO4 commutes with our other healthiness conditions.

Lemma 55 (commutativity-IOCO4-R1)

\[ \text{IOCO4} \circ R1 = R1 \circ \text{IOCO4} \]

Proof.

\[
\begin{align*}
\text{IOCO4}(\text{R1}(P)) &= \text{IOCO4}(P \land (tr \leq tr')) \\
&= P \land (tr \leq tr') \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|A_{out}| - 1) \leq |ref'_{out}|)) \\
&= P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|A_{out}| - 1) \leq |ref'_{out}|)) \land (tr \leq tr') \\
&= \text{IOCO4}(P) \land (tr \leq tr') \\
&= \text{R1}(\text{IOCO4}(P))
\end{align*}
\]

Lemma 56 (commutativity-IOCO4-R2)

\[ \text{IOCO4} \circ R2 = R2 \circ \text{IOCO4} \]

Proof.

\[
\begin{align*}
\text{IOCO4}(\text{R2}(P(tr,tr'))) &= \text{IOCO4}(P(\langle \rangle, tr' - tr)) \\
&= P(\langle \rangle, tr' - tr) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|A_{out}| - 1) \leq |ref'_{out}|)) \\
&= (P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|A_{out}| - 1) \leq |ref'_{out}|)))(\langle \rangle, tr' - tr) \\
&= \text{IOCO4}(P)(\langle \rangle, tr' - tr) \\
&= \text{R2}(\text{IOCO4}(P))
\end{align*}
\]

Lemma 57 (commutativity-IOCO4-R3)

\[ \text{IOCO4} \circ R3 = R3 \circ \text{IOCO4} \]
Proof.

\[
\text{IOCO4}(R_3^\theta(P)) =
\]

\[
= \text{IOCO4}(\Pi^\theta < \text{wait} \triangleright P)
\]

\[
= (\Pi^\theta < \text{wait} \triangleright P) \land (\lnot \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|A_{out}| - 1) \leq |ref_{out}'|))
\]

\[
= (\Pi^\theta < \text{wait} \triangleright (\text{wait}' \Rightarrow (|A_{out}| - 1) \leq |ref_{out}'|)) \land \lnot \text{wait}
\]

\[
= R_3^\theta(\text{IOCO4}(P))
\]

□

Lemma 58 (commutativity-IOCO4-IOCO1)

\[
\text{IOCO4} \circ \text{IOCO1} = \text{IOCO1} \circ \text{IOCO4}
\]

Proof.

\[
\text{IOCO4}(\text{IOCO1}(P)) =
\]

\[
= \text{IOCO4}(P \land (\text{ok} \Rightarrow (\text{wait}' \lor \text{ok}')))\land (\lnot \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|A_{out}| - 1) \leq |ref_{out}'|))
\]

\[
= \text{IOCO4}(P) \land (\text{ok} \Rightarrow (\text{wait}' \lor \text{ok}'))\land (\lnot \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|A_{out}| - 1) \leq |ref_{out}'|))
\]

\[
= \text{IOCO1}(\text{IOCO4}(P))
\]

□

Lemma 59 (commutativity-IOCO4-IOCO2)

\[
\text{IOCO4} \circ \text{IOCO2} = \text{IOCO2} \circ \text{IOCO4}
\]

Proof.

\[
\text{IOCO4}(\text{IOCO2}(P)) =
\]

\[
= \text{IOCO4}(P \land (\lnot \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{def}' \Rightarrow \text{quiescence}))))\land (\lnot \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|A_{out}| - 1) \leq |ref_{out}'|))
\]

\[
= \text{IOCO4}(P) \land (\lnot \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{def}' \Rightarrow \text{quiescence})))\land (\lnot \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|A_{out}| - 1) \leq |ref_{out}'|))
\]

\[
= \text{IOCO2}(\text{IOCO4}(P))
\]

□

Lemma 60 (commutativity-IOCO4-IOCO3)

\[
\text{IOCO4} \circ \text{IOCO3} = \text{IOCO3} \circ \text{IOCO4}
\]
Proof.

\[
\text{\textbf{IOCO4(IOCO3}(P)\text{)} = } \quad \text{(def. of IOCO3)} \\
= \text{\textbf{IOCO4}(P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref}' \Rightarrow \exists s \cdot \text{tr}' - \text{tr} = s \cdot \theta^*))))} \quad \text{(def. of IOCO4)} \\
= P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref}' \Rightarrow \exists s \cdot \text{tr}' - \text{tr} = s \cdot \theta^*))) \land \\
(\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|A_{\text{out}}| - 1) \leq |\text{ref}'_{\text{out}}|)) \land \\
(\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref}' \Rightarrow \exists s \cdot \text{tr}' - \text{tr} = s \cdot \theta^*))) \quad \text{(def. of IOCO4)} \\
= \text{\textbf{IOCO4}(P) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref}' \Rightarrow \exists s \cdot \text{tr}' - \text{tr} = s \cdot \theta^*)))} \quad \text{(def. of IOCO3)} \\
= \text{\textbf{IOCO3}(IOCO4}(P))
\]

\text{\textbf{IOCO4} is preserved by sequential composition (\text{;}), by quiescence preserving internal choice (\text{⊓\,θ}), and by quiescence preserving external choice (\text{+\,θ}}).

**Lemma 61 (closure-\text{;}-\text{IOCO4})**

\text{\textbf{IOCO4}(P;Q) = P;Q provided that P and Q are IOCO4 healthy and Q is R_{3,\theta} healthy}
Lemma 62 (closure-$\Gamma^P$-ICO4)

$$\text{ICO4}(P; Q) = P \Gamma^P_0 Q \text{ provided that } P \text{ and } Q \text{ are ICO4 healthy}$$
Lemma 63 (closure-+^θ-IOCO4)

\textit{IOCO4}(P \vdash^θ Q) = P \vdash^θ Q \text{ provided that } P \text{ and } Q \text{ are IOCO4 healthy}

\[\text{Lemma 63 (closure-+^θ-IOCO4)}\]

\[\text{IOCO4}(P \vdash^θ Q) = P \vdash^θ Q \text{ provided that } P \text{ and } Q \text{ are IOCO4 healthy}\]
Proof.

\[
\text{IOCO4}(P \ +^6 Q) = \text{IOCO4}(P \ \wedge Q; M_+) = \exists v_0, v_1 \cdot P[v_0/v'] \wedge Q[v_1/v'] \wedge M_+ [v_0, v_1/0.v, 1.v] \wedge (\text{wait} \lor \neg \text{wait}' \lor ((1.\text{A}_{\text{out}} - 1) \leq |\text{ref}_{\text{out}}'|))
\]

(assumption)

\[
= \exists v_0, v_1 \cdot \text{IOCO4}(P[v_0/v'] \wedge \text{IOCO4}(Q)[v_1/v'] \wedge M_+ [v_0, v_1/0.v, 1.v] \wedge (\text{wait} \lor \neg \text{wait}' \lor ((1.\text{A}_{\text{out}} - 1) \leq |\text{ref}_{\text{out}}'|))
\]

(lemma)

\[
= \exists v_0, v_1 \cdot P[v_0/v'] \wedge Q[v_1/v'] \wedge M_+ [v_0, v_1/0.v, 1.v] \wedge (\text{wait} \lor \neg \text{wait}' \lor ((1.\text{A}_{\text{out}} - 1) \leq |\text{ref}_{\text{out}}'|))
\]

(definition)

\[
= \exists v_0, v_1 \cdot P[v_0/v'] \wedge Q[v_1/v'] \wedge M_+ [v_0, v_1/0.v, 1.v] \wedge (\text{wait} \lor \neg \text{wait}' \lor ((1.\text{A}_{\text{out}} - 1) \leq |\text{ref}_{\text{out}}'|))
\]

(definition of \text{IOCO4})

\[
\]

\[
= \exists v_0, v_1 \cdot P[v_0/v'] \wedge Q[v_1/v'] \wedge M_+ [v_0, v_1/0.v, 1.v] \wedge (\text{wait} \lor \neg \text{wait}' \lor ((1.\text{A}_{\text{out}} - 1) \leq |\text{ref}_{\text{out}}'|))
\]

(substitution and lemma)

\[
= \exists v_0, v_1 \cdot P[v_0/v'] \wedge Q[v_1/v'] \wedge M_+ [v_0, v_1/0.v, 1.v] \wedge (\text{wait} \lor \neg \text{wait}' \lor ((1.\text{A}_{\text{out}} - 1) \leq |\text{ref}_{\text{out}}'|))
\]

(substitution and definition of \text{IOCO4})

\[
= \exists v_0, v_1 \cdot \text{IOCO4}(P)[v_0/v'] \wedge \text{IOCO4}(Q)[v_1/v'] \wedge M_+ [v_0, v_1/0.v, 1.v]
\]

(substitution and definition of \text{IOCO4})

\[
= \exists v_0, v_1 \cdot P[v_0/v'] \wedge Q[v_1/v'] \wedge M_+ [v_0, v_1/0.v, 1.v]
\]

(definition of \text{IOCO4})

\[
= P \ +^6 Q
\]

Input-enabledness. Input-enabledness requires that an implementation accepts every input in every (waiting) state. More precisely, an implementation cannot prevent the environment from providing an input, while running.

\[
\text{IOCO5} \quad P = P \wedge (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (\text{ref}_{\text{in}}' = \emptyset)))
\]

This healthiness condition is idempotent.

Lemma 64 (IOCO5-idempotent)

\[
\text{IOCO5} \circ \text{IOCO5} = \text{IOCO5}
\]

\[\text{This step was proven automatically using yices [Int08]. The proof-script can be found at} \quad \text{http://www.ist.tugraz.at/staff/weiglhofer}\]
Proof.

\[
\text{IOCO5}(\text{IOCO5}(P)) = \\
= \text{IOCO5}(P) \land (\neg \text{wait} \Rightarrow (\text{ref}'_n = \emptyset)) \\
= P \land (\neg \text{wait} \Rightarrow (\text{ref}'_n = \emptyset)) \land (\neg \text{wait} \Rightarrow (\text{ref}'_n = \emptyset)) \\
= P \land (\neg \text{wait} \Rightarrow (\text{ref}'_n = \emptyset)) \\
= \text{IOCO5}(P)
\]

Furthermore, \text{IOCO5} commutes with our other healthiness conditions.

Lemma 65 (commutativity-IOCO5-R1)

\[
\text{IOCO5} \circ R1 = R1 \circ \text{IOCO5}
\]

Proof.

\[
\text{IOCO5}(R1(P)) = \\
= \text{IOCO5}(P \land (tr \leq tr')) \\
= P \land (tr \leq tr') \land (\neg \text{wait} \Rightarrow (\text{ref}'_n = \emptyset)) \\
= P \land (\neg \text{wait} \Rightarrow (\text{ref}'_n = \emptyset)) \land (tr \leq tr') \\
= \text{IOCO5}(P) \land (tr \leq tr') \\
= R1(\text{IOCO5}(P))
\]

Lemma 66 (commutativity-IOCO5-R2)

\[
\text{IOCO5} \circ R2 = R2 \circ \text{IOCO5}
\]

Proof.

\[
\text{IOCO5}(R2(P(tr, tr'))) = \\
= \text{IOCO5}(P(\langle \rangle, tr' - tr)) \\
= P(\langle \rangle, tr' - tr) \land (\neg \text{wait} \Rightarrow (\text{ref}'_n = \emptyset)) \\
= P \land (\neg \text{wait} \Rightarrow (\text{ref}'_n = \emptyset)) \land (\langle \rangle, tr' - tr) \\
= \text{IOCO5}(P)(\langle \rangle, tr' - tr) \\
= R2(\text{IOCO5}(P))
\]

Lemma 67 (commutativity-IOCO5-R3)

\[
\text{IOCO5} \circ R3 = R3 \circ \text{IOCO5}
\]

Proof.

\[
\text{IOCO5}(R3^\theta(\langle \rangle)) = \\
= \text{IOCO5}(\text{IOCO5}(\langle \rangle) \land (\neg \text{wait} \Rightarrow (\text{ref}'_n = \emptyset))) \\
= \text{IOCO5}(\langle \rangle) \land (\neg \text{wait} \Rightarrow (\text{ref}'_n = \emptyset)) \land (\langle \rangle) < \text{wait} \\
= \langle \rangle < \text{wait} \lor (P \land (\neg \text{wait} \Rightarrow (\text{ref}'_n = \emptyset))) \\
= \langle \rangle < \text{wait} \lor (\text{IOCO5}(P)) \\
= R3^\theta(\text{IOCO5}(P))
\]
Lemma 68 (commutativity-IOCO5-IOCO1)

\[ \text{IOCO5} \circ \text{IOCO1} = \text{IOCO1} \circ \text{IOCO5} \]

Proof.

\[
\begin{align*}
\text{IOCO5}(\text{IOCO1}(P)) &= \text{IOCO5}(P \land (ok \Rightarrow (\text{wait}' \lor \text{ok}'))) \\
&= P \land (\text{wait'} \Rightarrow (\theta \notin \text{ref}' \Rightarrow \text{quiescence}')) \\
&= P \land (\text{wait'} \Rightarrow (\theta \notin \text{ref}' \Rightarrow \text{quiescence}')) \land (\text{wait'} \Rightarrow (\text{ref}'_\text{in} = \emptyset)) \\
&= P \land (\text{wait'} \Rightarrow (\theta \notin \text{ref}' \Rightarrow \text{quiescence}')) \land (\text{wait'} \Rightarrow (\text{ref}'_\text{in} = \emptyset)) \\
&= \text{IOCO5}(P) \land (ok \Rightarrow (\text{wait}' \lor \text{ok}')) \\
&= \text{IOCO1}(\text{IOCO5}(P))
\end{align*}
\]

Lemma 69 (commutativity-IOCO5-IOCO2)

\[ \text{IOCO5} \circ \text{IOCO2} = \text{IOCO2} \circ \text{IOCO5} \]

Proof.

\[
\begin{align*}
\text{IOCO5}(\text{IOCO2}(P)) &= \text{IOCO5}(P \land (\text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref}' \Rightarrow \text{quiescence}')))) \\
&= P \land (\text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref}' \Rightarrow \text{quiescence}')) \land (\text{wait}' \Rightarrow (\text{ref}'_\text{in} = \emptyset)) \\
&= P \land (\text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref}' \Rightarrow \text{quiescence}')) \land (\text{wait}' \Rightarrow (\text{ref}'_\text{in} = \emptyset)) \\
&= \text{IOCO5}(P) \land (\text{wait} \Rightarrow (\text{wait}' \Rightarrow (\theta \notin \text{ref}' \Rightarrow \text{quiescence}'))) \\
&= \text{IOCO2}(\text{IOCO5}(P))
\end{align*}
\]

Lemma 70 (commutativity-IOCO5-IOCO3)

\[ \text{IOCO5} \circ \text{IOCO3} = \text{IOCO3} \circ \text{IOCO5} \]

Proof.

\[
\begin{align*}
\text{IOCO5}(\text{IOCO3}(P)) &= \text{IOCO5}(P \land (\text{wait} \Rightarrow (\theta \notin \text{ref}' \Rightarrow \exists s \bullet tr' - tr = s \cdot \theta'^*))))) \\
&= P \land (\text{wait} \Rightarrow (\theta \notin \text{ref}' \Rightarrow \exists s \bullet tr' - tr = s \cdot \theta'^*)) \land (\text{wait} \Rightarrow (\text{ref}'_\text{in} = \emptyset)) \\
&= P \land (\text{wait} \Rightarrow (\theta \notin \text{ref}' \Rightarrow \exists s \bullet tr' - tr = s \cdot \theta'^*)) \land (\text{wait} \Rightarrow (\text{ref}'_\text{in} = \emptyset)) \\
&= \text{IOCO5}(P) \land (\text{wait} \Rightarrow (\theta \notin \text{ref}' \Rightarrow \exists s \bullet tr' - tr = s \cdot \theta'^*)) \\
&= \text{IOCO3}(\text{IOCO5}(P))
\end{align*}
\]

\(\text{IOCO5}\) is preserved by sequential composition (\(\cdot\)), by quiescence preserving internal choice (\(\text{ref}\)), and by quiescence preserving external choice (\(\text{ref}'\)):

Lemma 71 (closure-\(\cdot\)-\(\text{IOCO5}\))

\[ \text{IOCO5}(P; Q) = P; Q \text{ provided that } P \text{ and } Q \text{ are } \text{IOCO5} \text{ healthy and } Q \text{ is } R3^\theta \text{ healthy} \]
Proof.

\[ \text{IOCO5}(P;Q) = \]

(assumption and def. of R3'')

\[ = \text{IOCO5}(P; (I_0^\emptyset < wait > Q)) \]

(def. of :)

\[ = \text{IOCO5} \exists v_0 \cdot P[v_0/v'] \land (I_0^\emptyset < wait > Q)[v_0/v] \]

(def. IOCO5, def. if. and substitution)

\[ = \exists v_0 \cdot \left( P[v_0/v'] \land \neg wait_0 \land Q[v_0/v] \land (wait \lor \neg wait' \lor (ref'_in = \emptyset)) \right) \]

(def. of \(I_0^\emptyset\) and prop. calculus)

\[ = \exists v_0 \cdot \left( \left( P[v_0/v'] \land \neg wait_0 \land \neg ok_0 \land \cdots \land (wait' \lor (ref'_in = \emptyset)) \land (wait \lor \neg wait' \lor (ref'_in = \emptyset)) \land \right) \]

(substitution and prop. calculus)

\[ = \exists v_0 \cdot \left( P[v_0/v'] \land \neg wait_0 \land \neg ok_0 \land \cdots \land (wait' = wait_0) \land (ref'_in = ref_{in_0}) \land (wait \lor \neg wait' \lor (ref'_in = \emptyset)) \land \right) \]

(prop. calculus)

\[ = \exists v_0 \cdot \left( P[v_0/v'] \land \neg wait_0 \land \neg ok_0 \land \cdots \land (wait' = wait_0) \land (ref'_in = ref_{in_0}) \land (wait \lor \neg wait' \lor (ref'_in = \emptyset)) \land \right) \]

(substitution)

\[ = \exists v_0 \cdot \left( P[v_0/v'] \land \neg wait_0 \land \neg ok_0 \land \cdots \land (wait' = wait_0) \land (ref'_in = ref_{in_0}) \land \right) \]

(def. of IOCO5 and assumption)

\[ = \exists v_0 \cdot \left( P[v_0/v'] \land \neg wait_0 \land \neg ok_0 \land \cdots \land (wait' = wait_0) \land (ref'_in = ref_{in_0}) \land \right) \]

(prop. calculus and def. of \(I_0^\emptyset\) and assumption)

\[ = \exists v_0 \cdot \left( P[v_0/v'] \land \neg wait_0 \land \neg I_0^\emptyset[v_0/v] \lor \right) \]

(assumption and def. of R3'')

\[ = \exists v_0 \cdot \left( P[v_0/v'] \land \neg wait_0 \land IOCO5[v_0/v] \land (wait \lor \neg wait' \lor (ref'_in = \emptyset)) \right) \]

(def. IOCO5 and renaming)

\[ = \exists v_0 \cdot \left( P[v_0/v'] \land \neg wait_0 \land \neg I_0^\emptyset[v_0/v] \lor \right) \]

(prop. calculus)

\[ = \exists v_0 \cdot \left( P[v_0/v'] \land \neg wait_0 \land \neg I_0^\emptyset[v_0/v] \lor \right) \]

(substitution and def. of IOCO5)

\[ = \exists v_0 \cdot \left( P[v_0/v'] \land \neg wait_0 \land \neg I_0^\emptyset[v_0/v] \lor \right) \]

(assumption and def. of R3'')

\[ = \exists v_0 \cdot \left( P[v_0/v'] \land \neg wait_0 \land IOCO5(Q)[v_0/v] \right) \]

(assumption and def. of :)

\[ = P; Q \]

\[ \square \]

Lemma 72 (closure-\(\neg^\emptyset\)-IOCO5)

\[ \text{IOCO5}(P \lnot^\emptyset Q) = P \lnot^\emptyset Q \text{ provided that } P \text{ and } Q \text{ are IOCO5 healthy} \]
Proof.

\[ \text{IOCO5}(P \uparrow^\theta Q) = \] (def. of \( \uparrow^\theta \))

\[ = \text{IOCO5}(P \land Q); M^n) \] (Lemma 22)

\[ = \text{IOCO5}(\exists v_0, v_1 \bullet P[v_0/v'] \land Q[v_1/v'] \land M^n[v_0, v_1/0.v, 1.v]) \] (def. of IOCO5)

\[ = \exists v_0, v_1 \bullet P[v_0/v'] \land Q[v_1/v'] \land M^n[v_0, v_1/0.v, 1.v] \land (wait \lor \neg \text{wait}' \lor (\text{ref}_{in} = \emptyset)) \] (assumption)

\[ = \exists v_0, v_1 \bullet \text{IOCO5}(P)[v_0/v'] \land \text{IOCO5}(Q)[v_1/v'] \land M^n[v_0, v_1/0.v, 1.v] \land (wait \lor \neg \text{wait}' \lor (\text{ref}_{in} = \emptyset)) \] (def. of IOCO5 and substitution)

\[ = \exists v_0, v_1 \bullet P[v_0/v'] \land (\text{wait} \lor \neg \text{wait}_0 \lor (\text{ref}_{in_0} = \emptyset)) \land Q[v_1/v'] \land (\text{wait} \lor \neg \text{wait}_1 \lor (\text{ref}_{in_1} = \emptyset)) \land M^n[v_0, v_1/0.v, 1.v] \land \] (Lemma 29 and substitution)

\[ (\text{wait}' = (\text{wait}_0 \lor \text{wait}_1)) \land \] (prop. calculus\footnote{This step was proven automatically using yices \cite{Int08}. The proof-script can be found at http://www.ist.tugraz.at/staff/weiglhofer})

\[ \left( \begin{array}{c}
\text{tr}' = t_0 \land \cdots \land \text{ref}_{in_0}' = \text{ref}_{in_0} \forall \\
\text{tr}' = t_1 \land \cdots \land \text{ref}_{in_1}' = \text{ref}_{in_1}
\end{array} \right) \] (Lemma 22)

\[ \left( \begin{array}{c}
\text{tr}' = t_0 \land \cdots \land \text{ref}_{in_0}' = \text{ref}_{in_0} \forall \\
\text{tr}' = t_1 \land \cdots \land \text{ref}_{in_1}' = \text{ref}_{in_1}
\end{array} \right) \] (substitution and Lemma 29)

\[ = \exists v_0, v_1 \bullet P[v_0/v'] \land Q[v_1/v'] \land M^n[v_0, v_1/0.v, 1.v] \land \] (substitution and def. of IOCO5)

\[ Q[v_1/v'] \land (\text{wait} \lor \neg \text{wait}_1 \lor (\text{ref}_{in_1} = \emptyset)) \land M^n[v_0, v_1/0.v, 1.v] \] (assumption)

\[ = \exists v_0, v_1 \bullet \text{IOCO5}(P)[v_0/v'] \land \text{IOCO5}(Q)[v_1/v'] \land M^n[v_0, v_1/0.v, 1.v] \] (Lemma 22)

\[ = (P \land Q); M^n \] (def. of \( \uparrow^\theta \))

\[ = P \uparrow^\theta Q \]

\[ \square \]

Lemma 73 (closure-\( +^\theta \)-IOCO5)

\[ \text{IOCO5}(P \ +^\theta Q) = P \uparrow^\theta Q \] provided that P and Q are IOCO5 healthy

\[ \]
Proof.

\[ \text{IOCO5} \left( P \ +^p Q \right) = \]

\[ = \text{IOCO5} \left( P \perp Q; \ M_+ \right) \quad \text{(def. of } +^p \text{)} \]

\[ = \text{IOCO5} \left( \exists v_0, v_1 \cdot P[v_0/v'] \land Q[v_1/v'] \land M_+ [v_0, v_1/0,v,1.v] \right) \quad \text{(def. of } \text{IOCO5} \text{)} \]

\[ = \exists v_0, v_1 \cdot P[v_0/v'] \land Q[v_1/v'] \land M_+ [v_0, v_1/0,v,1.v] \land (\text{wait } \lor \text{ref}_{in} = \emptyset) \quad \text{(assumption)} \]

\[ = \exists v_0, v_1 \cdot \text{IOCO5} \left( P[v_0/v'] \land \text{IOCO5} \left( Q[v_1/v'] \land M_+ [v_0, v_1/0,v,1.v] \land (\text{wait } \lor \text{ref}_{in} = \emptyset) \right) \right) \quad \text{(def. } \text{IOCO5} \text{ and substitution)} \]

\[ = \exists v_0, v_1 \cdot P[v_0/v'] \land (\text{wait } \lor \text{ref}_{in} = \emptyset) \land Q[v_1/v'] \land (\text{wait } \lor \text{ref}_{in} = \emptyset) \land M_+ [v_0, v_1/0,v,1.v] \land (\text{wait } \lor \text{ref}_{in} = \emptyset) \]

\[ = \exists v_0, v_1 \cdot P[v_0/v'] \land Q[v_1/v'] \land M_+ [v_0, v_1/0,v,1.v] \land (\text{wait } \lor \text{ref}_{in} = \emptyset) \]

\[ = \exists v_0, v_1 \cdot P[v_0/v'] \land Q[v_1/v'] \land M_+ [v_0, v_1/0,v,1.v] \land \text{IOCO5} \left( P \perp Q; \ M_+ \right) \quad \text{(Lemma 42 and substitution)} \]

\[ = \exists v_0, v_1 \cdot P[v_0/v'] \land Q[v_1/v'] \land M_+ [v_0, v_1/0,v,1.v] \land \text{IOCO5} \left( P \perp Q; \ M_+ \right) \quad \text{(substitution and def. of } \text{IOCO5} \text{)} \]

\[ = P \ +^p Q \]

As for specifications we need to redefine \( \sqsubseteq \) such that even in the case of divergence the additional properties of implementations are satisfied.

\[ \sqsubseteq_\theta^\phi_t = \left( \text{ok } \land (\text{tr } \preceq \text{tr'}) \land (\text{wait'} \Rightarrow (\theta \notin \text{ref}' \Rightarrow \text{quiescence}) ) \land (\text{wait'} \Rightarrow (\theta \notin \text{ref}' \Rightarrow \exists s \cdot \text{tr'} = s \theta^* ) ) \land (\text{wait'} \Rightarrow (|A_{out}| - 1) \leq |\text{ref}'_{out}| ) \land (\text{wait'} \Rightarrow (\text{ref}'_{in} = \emptyset ) ) \right) \lor \left( \text{ok}^\phi \land (v' = v) \right) \]

Using this new version of the \( \sqsubseteq \) relation within the healthiness condition \( \text{R3} \) leads to \( \text{R3}^\phi_t \) which is used for implementations

\[ \text{R3}^\phi_t (P) = \sqsubseteq_\theta^\phi_t \prec \text{wait } \triangleright P \]

While specifications are expressed in terms of \( d_{\theta}^\phi \), implementations use \( \iota_{\lambda} \). More precisely, we express implementations by the use of \( \theta^\phi_t \). Let us start with \( \iota_{\lambda} \) first. \( \iota_{\lambda} \) takes care of the input-enabledness of processes.

For the sake of simplicity we use the following abbreviation to denote a sequence of inputs without a particular action: \( A_{in}^\phi = \sqsubseteq \left( A_{in} \setminus \{ a \} \right)^* \).

---

13This step was proven automatically using yices [Int08]. The proof-script can be found at
http://www.ist.tugraz.at/staff/weighner
Definition 23 (Input-enabled communication) Let \( a \in A \) be an action of a process’ alphabet, then
\[
\iota^\theta_A(a) = a \Phi^i(ref )'_{in} = \emptyset \land a \notin ref \land tr' - tr \in A_{in}^* \, \prec \, wait' \triangleright \ tr' - tr \in A_{in\setminus a}^* \, \prec \, (a)
\]
\( \iota^\theta_A(a) \) is similar to \( do_A(a) \). It denotes that the process \( \iota^\theta_A(a) \) cannot refuse to perform an \( a \)-action. Furthermore, \( \iota^\theta_A(a) \) cannot refuse to perform any input action. After executing any input action sequence ended by an \( a \) action the process \( \iota^\theta_A(a) \) terminates successfully.

Input enabledness also affects the representation of a deadlock. An input-enabled process needs to accept an input action at any time. That is an input-enabled process can only deadlock on outputs. Therefore, the deadlock process \( \delta_i \), which substitutes \( \delta \) in the case of input-enabled processes, is given by:

Definition 24 (Output deadlock)
\[
\delta_i = a \Theta^i_A(tr' - tr \in A_{in}^* \land \text{wait'})
\]

The output deadlock is a left-zero for sequential composition

Lemma 74 (\( \delta_i \)-left-zero)
\[
\delta_i; P = \delta_i
\]

Proof.
\[
\begin{align*}
\delta_i; P &= \\
&= \delta_i \prec \text{wait} \triangleright (\delta_i; P) \\
&= \delta_i \prec \text{wait} \triangleright ((\delta_i \prec \text{wait} \triangleright tr' - tr \in A_{in}^* \land \text{wait'}); P) \quad \text{(def. of } \delta_i) \\
&= \delta_i \prec \text{wait} \triangleright ((tr' - tr \in A_{in}^* \land \text{wait'}); \delta_i \prec \text{wait} \triangleright P) \quad \text{(P meets R3)} \\
&= \delta_i \prec \text{wait} \triangleright ((tr' - tr \in A_{in}^* \land \text{wait'}); \delta_i \prec \text{wait} \triangleright P) \quad \text{(def. of ;)} \\
&= \delta_i \prec \text{wait} \triangleright ((tr' - tr \in A_{in}^* \land \text{wait'})) \quad \text{(; unit)} \\
&= \delta_i. \quad \Box
\end{align*}
\]

Again, as for the non-input-enabled case, we need a quiescent version of \( \iota_A \). \( \iota^\theta_A(a) \) has \( \theta \) events within its traces if \( a \) is an input event.

Definition 25 (Input-enabled quiescent communication) Let \( a \in A \) be an action of a process’ alphabet, then
\[
\iota^\theta_A(a) =
\begin{cases}
\iota_A(a) & \text{if } a \in A_{out} \\
\Phi^i(ref )'_{in} = \emptyset \land \theta \notin ref \land tr' - tr \in (A_{in\setminus a} \cup \theta)^* \, \prec \, \text{wait'} \triangleright \ tr' - tr \in (A_{in\setminus a} \cup \theta)^* \, \prec \, (a) & \text{if } a \in A_{in}
\end{cases}
\]

Combining input-enabledness with quiescence again requires a slight modification of the deadlock process. This leads to the output quiescent deadlock:

Definition 26 (Quiescent output deadlock)
\[
\delta^\theta_i = a \Theta^i_A(tr' - tr \in (A_{in} \cup \theta)^* \land \text{wait'})
\]

The quiescent output deadlock is a left-zero for sequential composition

Lemma 75 (\( \delta^\theta_i \)-left-zero)
\[
\delta^\theta_i; P = \delta^\theta_i
\]
Proof. 
\[ \delta_0^\theta; P = \] (closure of \( R_3^\theta \)) 
\[ = \overline{1}^\theta \triangleleft \text{wait} \triangleright (\delta_0^\theta; P) \] (def. of \( \delta_0^\theta \))
\[ = \overline{1}^\theta \triangleleft \text{wait} \triangleright ((\overline{1}^\theta \triangleleft \text{wait} \triangleright tr' - tr \in (A_m \cup \theta)^* \land \text{wait}' \land \text{wait}'); P) \] (def. of \( \triangleleft_p \))
\[ = \overline{1}^\theta \triangleleft \text{wait} \triangleright ((tr' - tr \in (A_m \cup \theta)^* \land \text{wait}' \land \text{wait}'); P) \] (P meets \( R_3 \))
\[ = \overline{1}^\theta \triangleleft \text{wait} \triangleright ((tr' - tr \in (A_m \cup \theta)^* \land \text{wait}' \land \text{wait}'); \overline{1}^\theta \triangleleft \text{wait} \triangleright P) \] (P meets \( R_3 \))
\[ = \overline{1}^\theta \triangleleft \text{wait} \triangleright ((tr' - tr \in (A_m \cup \theta)^* \land \text{wait}' \land \text{wait}'); \overline{1}^\theta \triangleleft \text{wait} \triangleright P) \] (def. of \( \cdot \))
\[ = \overline{1}^\theta \triangleleft \text{wait} \triangleright ((tr' - tr \in (A_m \cup \theta)^* \land \text{wait}' \land \text{wait}')); \overline{1}^\theta \triangleleft \text{wait} \triangleright P \] (P meets \( R_3 \))
\[ = \overline{1}^\theta \triangleleft \text{wait} \triangleright ((tr' - tr \in (A_m \cup \theta)^* \land \text{wait}' \land \text{wait}')); (\text{unit}) \) (def. of \( \delta_0^\theta \))
\[ = \delta_0^\theta \] \( \square \)

Fairness. Fairness is especially important for allowing theoretical exhaustive test case generation algorithms. The fairness assumption for the \( \text{icoco} \) relation requires that an implementation eventually shows all its possible non-deterministic behaviors when it is re-executed with a particular set of inputs. Without assuming fairness of implementations there is not even a theoretical possibility of generating a failing test case for any non-conforming implementation. An unfair implementation may always lead a test case away from its errors. To express this fairness on an implementation we use a probabilistic choice operator similar to He and Sanders [HS06]. According to [HS06] a probabilistic choice between two processes A and B is expressed with \( A_p \oplus B \) where \( 0 \leq p \leq 1 \). This expression equals A with probability \( p \) and B with probability \( 1 - p \). For example, \( A_{0.9} \oplus B \) denotes that during execution A is chosen in 90\% of the cases.

The probabilistic version of our quiescence preserving internal choice, i.e. \( \triangleright^\theta \), is given by \( p^\triangleright^\theta \). The laws for \( p^\triangleright^\theta \) are similar to the laws for \( p^\oplus \), i.e.,
\[ P_{1^\theta} Q = P \]
\[ P_{p^\theta} Q = Q_{1-p^\theta} P \]
\[ P_{p^\theta} P = P \]
\[ (P_{p^\theta} Q); q^\theta R = P_{pq^\theta} (Q_{1-q^\theta} R), \quad r = ((1-p)q)/(1-(pq)) \]
\[ (P_{p^\theta} Q); R = (P; R)_{p^\theta} (Q; R) \]

A quiescence preserving internal choice is given by the non-deterministic choice of all possible probabilistic choices, i.e.
\[ P \triangleright^\theta Q = \sqcap \{P_{p^\theta} Q\}_{0 \leq p \leq 1} \subseteq P_{p^\theta} Q \]

Relying on probabilistic choices means that when one implements a choice the specification is refined by choosing a particular probability for this choice. Fairness is expressed by restricting the probabilities \( p \) to \( 0 < p < 1 \):

Definition 27 (Internal fair (quiescence preserving) choice)
\[ P \triangleright_f^\theta Q = \sqcap \{P_{p^\theta} Q\}_{0 < p < 1} \]

Thus, when executing a test case on the implementation the implementation will eventually exhibit all its possible behavior. As stated by the following lemma fair internal quiescence preserving choices are valid implementations of internal quiescence preserving choices. This guarantees that our internal quiescence preserving choice can be safely implemented by its fair version.

Lemma 76 \( P \triangleright^\theta Q \subseteq P \triangleright_f^\theta Q \)
Proof.

\[ P \sqcap^\theta Q = \{ \text{definition of } \sqcap^\theta \} \]
\[ = \sqcap \{ P \sigma^\theta Q | 0 < p < 1 \} \{ \text{definition of } \sqcap \} \]
\[ \sqsubseteq \sqcap \{ P \sigma^\theta Q | 0 < p < 1 \} \sqcap P \sqcap Q \{ \text{laws for } \sigma^\theta \} \]
\[ = \sqcap \{ P \sigma^\theta Q | 0 < p < 1 \} \sqcap P \sqcap Q \{ \text{definition of } \sqcap \} \]
\[ = \sqcap \{ P \sigma^\theta Q | 0 \leq p \leq 1 \} \{ \text{laws for } \sigma^\theta \} \]
\[ = P \sqcap^\theta Q \]

Given the notion of input-enabledness and fairness we can now define which processes serve to represent ioco testable implementations:

**Definition 28 (ioco testable implementation)** An ioco testable implementation is a reactive process satisfying the healthiness conditions IOCO1-IOCO5. In addition, an ioco testable implementation must be fair.

Processes expressed in terms of \( \iota^\theta_A, \cdot^\theta \), and \( \sqcap^\theta \) are ioco testable implementations if their choices obey to the following rules: (1) Choices between outputs are fair internal choices (\( \sqcap^\theta \)); (2) Choices between inputs and choices between inputs and outputs are external choices (+\( \theta \)).

**Theorem 2** Implementation processes are closed under \( \iota^\theta_A, \cdot^\theta \), and \( \sqcap^\theta_f \).

**Proof.** Idempotence of healthiness conditions and healthiness conditions are preserved (see lemmas). □

### 3.4 Predicative Input Output Conformance Relation

Recall that informally an IUT conforms to a specification S, iff the outputs of the IUT are outputs of S after an arbitrary suspension trace of S.

Thus, we need the (suspension) traces of a process, which are obtained by hiding all observations except the traces.

**Definition 29 (Traces of a process)**

\[ \text{Trace}(P) =_{df} \exists ref, ref', wait, wait', ok, ok' \cdot P \]

In addition to all traces of a particular process we need the traces after which a process is quiescent. Due to the chosen representation of quiescence (see Section 3.2) we use the following predicate in order to obtain the traces after which a process is quiescent.

**Definition 30 (Quiet traces of a process)**

\[ \text{Quiet}(P) =_{df} \exists ref'_{\text{in}} \cdot (P[false/wait'] \lor P[A_{\text{out}}/ref'_{\text{out}}]) \]

Using these two predicates the input output conformance relation between implementation processes (see Definition 28) and specification processes (see Definition 22) can be defined as follows:

**Definition 31 (\( \sqsubseteq_{\text{ioco}} \))** Given an implementation process \( I \) and a specification process \( S \), then

\[ S \sqsubseteq_{\text{ioco}} I =_{df} \forall t \in \mathcal{A}_{\text{in}}, \forall o \in \mathcal{A}_{\text{out}} \bullet \]
\[ ((\text{Trace}(S)[t/tr'] \land \text{Trace}(I)[t \diamond o/tr']) \Rightarrow \text{Trace}(S)[t \diamond o/tr']) \land \]
\[ ((\text{Trace}(S)[t/tr'] \land \text{Quiet}(I)[t/tr']) \Rightarrow \text{Quiet}(S)[t/tr']) \]
In order to distinguish the input output conformance given in denotational semantics from its operational semantics version we use different symbols. Note that because \( \subseteq_{\text{ioco}} \) is related to refinement \( I \text{ ioco} S \) is given by \( S \subseteq_{\text{ioco}} I \).

\textit{ioco} relates the outputs (including quiescence) of \( I \) and \( S \) for all suspension traces of \( S \). Contrary, our \( \subseteq_{\text{ioco}} \) definition comprises two different parts. The first part considers only outputs while the second part deals with quiescence.

Using the predicative definition \( \subseteq_{\text{ioco}} \) we can now show the relation between the input output conformance relation and refinement.

**Theorem 3** \( \subseteq \subseteq \subseteq_{\text{ioco}} \)

**Proof.** In order to prove \( \subseteq \subseteq \subseteq_{\text{ioco}} \), we have to show that \( S \subseteq I \Rightarrow S \subseteq_{\text{ioco}} I \)

\[
S \subseteq I \quad \{\text{definition of } \subseteq \text{ and propositional calculus}\}
\]

\[
= [ I \Rightarrow S ] \land ([ I \Rightarrow S ] \lor [ I \Rightarrow S ])
\]

\[
= [ I \Rightarrow S ] \land ([ I[\text{false}/\text{wait}] \Rightarrow S[\text{false}/\text{wait}']]) \lor
\]

\[
[ I[A_{\text{out}}/\text{ref}'_{\text{out}}] \Rightarrow S[A_{\text{out}}/\text{ref}'_{\text{out}}])
\]

\[
= [ I \Rightarrow S ] \land ([ I[\text{false}/\text{wait}] \land I[A_{\text{out}}/\text{ref}'_{\text{out}}]) \Rightarrow
\]

\[
(S[\text{false}/\text{wait}] \lor S[A_{\text{out}}/\text{ref}'_{\text{out}}])
\]

\[
\Rightarrow [(I \Rightarrow S) \land ((\exists \text{ref}'_{\text{in}} \bullet (I[\text{false}/\text{wait}] \lor I[A_{\text{out}}/\text{ref}'_{\text{out}}]))
\]

\[
\Rightarrow (\exists \text{ref}'_{\text{in}} \bullet (S[\text{false}/\text{wait}] \lor S[A_{\text{out}}/\text{ref}'_{\text{out}}]))]
\]

\[
\Rightarrow [(\exists \text{ref}, \text{ref}', \text{wait}, \text{wait}', \text{ok}, \text{ok}' \bullet I) \Rightarrow (\exists \text{ref}, \text{ref}', \text{wait}, \text{wait}', \text{ok}, \text{ok}' \bullet S)]
\]

\[
\Rightarrow (\text{prop. calculus and def. of } \text{Tr} \text{ac})
\]

\[
\Rightarrow [(\forall t \in A_{\text{g}} \bullet (\text{Tr} \text{ac}(I)[t/t'] \Rightarrow \text{Tr} \text{ac}(S)[t/t']]) \land
\]

\[
(\forall t \in A_{\text{g}} \bullet (\text{Qu} \text{iet}(I)[t/t'] \Rightarrow \text{Qu} \text{iet}(S)[t/t']])
\]

\[
\Rightarrow [(\forall t \in A_{\text{g}} \bullet (\text{Tr} \text{ac}(I)[\check{t} o/tr'] \Rightarrow \text{Tr} \text{ac}(S)[\check{t} o/tr']) \land
\]

\[
(\forall t \in A_{\text{g}} \bullet (\text{Qu} \text{iet}(I)[t/t'] \Rightarrow \text{Qu} \text{iet}(S)[t/t']])]
\]

\[
\Rightarrow [(\forall t \in A_{\text{g}} \bullet (T_{\text{r}}(S)[t/t'] \land T_{\text{r}}(I)[\check{t} o/tr']) \Rightarrow T_{\text{r}}(S)[\check{t} o/tr']) \land
\]

\[
(\forall t \in A_{\text{g}} \bullet (T_{\text{r}}(S)[t/t'] \land Q_{\text{ui}}(I)[t/t']) \Rightarrow Q_{\text{ui}}(S)[t/t'])]
\]

\[
\Rightarrow [(\forall t \in A_{\text{g}} \bullet (\text{Tr} \text{ac}(S)[t/t'] \land \text{Tr} \text{ac}(I)[\check{t} o/tr']) \Rightarrow \text{Tr} \text{ac}(S)[\check{t} o/tr']) \land
\]

\[
(\forall t \in A_{\text{g}} \bullet (\text{Q} \text{ui}(S)[t/t'] \land Q_{\text{ui}}(I)[t/t']) \Rightarrow Q_{\text{ui}}(S)[t/t'])]
\]

\[
\Rightarrow S \subseteq_{\text{ioco}} I
\]

\[
\square
\]

Although, \( \subseteq_{\text{ioco}} \) and \( \text{ioco} \) are not transitive in general, an interesting property is that refining a conforming implementation does not break conformance.

**Theorem 4** \((S \subseteq_{\text{ioco}} I_{2}) \land (I_{2} \subseteq I_{1}) \Rightarrow S \subseteq_{\text{ioco}} I_{1}\)

**Proof.**

\[
S \subseteq_{\text{ioco}} I_{2} \land I_{2} \subseteq I_{1} \quad \{\text{definition of } \subseteq \text{ and propositional calculus}\}
\]

\[
= (S \subseteq_{\text{ioco}} I_{2}) \land (I_{1} \Rightarrow I_{2}) \land (I_{1} \Rightarrow I_{2}) \quad \{\text{def. of } \exists, \text{Tr} \text{ac}, \text{and Q} \text{ui}\}
\]

\[
\Rightarrow (S \subseteq_{\text{ioco}} I_{2}) \land (\text{Tr} \text{ac}(I_{1}) \Rightarrow \text{Tr} \text{ac}(I_{2})) \land (\text{Q} \text{ui}(I_{1}) \Rightarrow \text{Q} \text{ui}(I_{2})) \quad \{\text{propositional calculus}\}
\]

\[
\Rightarrow [(\forall t \in A_{\text{g}} \bullet (\forall o \in A_{\text{out}} \bullet
\]

\[
(\text{Tr} \text{ac}(S)[t/t'] \land \text{Tr} \text{ac}(I)[\check{t} o/tr']) \Rightarrow \text{Tr} \text{ac}(S)[\check{t} o/tr']) \land
\]

\[
(\forall t \in A_{\text{g}} \bullet (\forall o \in A_{\text{out}} \bullet (\text{Qu} \text{iet}(S)[t/t'] \land \text{Qu} \text{iet}(I)[t/t']) \Rightarrow \text{Qu} \text{iet}(S)[t/t'])]
\]

\[
\Rightarrow [\forall t \in A_{\text{g}} \bullet (\forall o \in A_{\text{out}} \bullet
\]

\[
((\text{Tr} \text{ac}(S)[t/t'] \land \text{Tr} \text{ac}(I)[\check{t} o/tr']) \Rightarrow \text{Tr} \text{ac}(S)[\check{t} o/tr']) \land
\]

\[
(\forall t \in A_{\text{g}} \bullet (\forall o \in A_{\text{out}} \bullet (\text{Q} \text{ui}(S)[t/t'] \land \text{Q} \text{ui}(I)[t/t']) \Rightarrow \text{Q} \text{ui}(S)[t/t'])]
\]

\[
\Rightarrow S \subseteq_{\text{ioco}} I_{1}
\]
\[ \forall t \in A^*_o \bullet ( ((\text{Trace}(S)[t/tr'] \land \text{Quiet}(I_2)[t/tr']) \Rightarrow \text{Quiet}(S)[t/tr']) \land \\
(Q\text{uiet}(I_1)[t/tr'] \Rightarrow \text{Quiet}(I_2)[t/tr'])) \]  \{ \text{prop. calculus} \}

\Rightarrow \forall t \in A^*_o, \forall o \in A_{\text{out}} \bullet 

\((((\text{Trace}(S)[t/tr'] \land \text{Trace}(I_1)[t\overset{o}{/tr'}]) \Rightarrow \text{Trace}(S)[t\overset{o}{/tr'}]) \land \\
((\text{Trace}(S)[t/tr'] \land \text{Quiet}(I_1)[t/tr']) \Rightarrow \text{Quiet}(S)[t/tr'])) \} \{ \text{def. of } \sqsubseteq_{\text{ioco}} \}

= S \sqsubseteq_{\text{ioco}} I_1

Although, the definition of \( \sqsubseteq_{\text{ioco}} \) corresponds to the definition of \( \text{io} \text{co} \) we can reformulate our definition to a more generic version \( \sqsubseteq_{\text{ioco}} \) which corresponds to \( \text{io} \text{co}_F \). Like \( F \) in \( \text{io} \text{co}_F \), \( P \) is used to select the proper set of traces.

**Definition 32** (\( \sqsubseteq_{\text{io}co}^P \)) Given an implementation process \( I \) and a specification process \( S \), then

\[ S \sqsubseteq_{\text{io}co}^P I =_{def} [ \forall t \in A^*_o, \forall o \in A_{\text{out}} \bullet 

((P(I,S,t) \land \text{Trace}(I)[t\overset{o}{/tr'}]) \Rightarrow \text{Trace}(S)[t\overset{o}{/tr'}]) \land \\
((P(I,S,t) \land \text{Quiet}(I)[t/tr']) \Rightarrow \text{Quiet}(S)[t/tr']) ] \]

The conformance relations listed in Section 2 can now be defined as follows:

**Definition 33** (Conformance relations)

\[ S \sqsubseteq_{\text{int}} I \ =_{def} S \sqsubseteq_{\text{io}co}^{P_{\text{int}}} I, \text{ where } P_{\text{int}}(S,I,t) = t \in A^*_o \]

\[ S \sqsubseteq_{\text{ior}} I \ =_{def} S \sqsubseteq_{\text{io}co}^{P_{\text{ior}}} I, \text{ where } P_{\text{ior}}(S,I,t) = t \in A^* \]

\[ S \sqsubseteq_{\text{io}co_f} I \ =_{def} S \sqsubseteq_{\text{io}co}^{P_{\text{io}co_f}} I, \text{ where } P_{\text{io}co_f}(S,I,t) = \text{Trace}(S)[t/tr'] \land t \in A^* \]

\[ S \sqsubseteq_{\text{io}co} I \ =_{def} S \sqsubseteq_{\text{io}co}^{P_{\text{io}co}} I, \text{ where } P_{\text{io}co}(S,I,t) = \text{Trace}(S)[t/tr'] \land t \in A^*_o \]

### 3.5 Test Cases, Test Processes, and Test Suites

Testing for conformance is done by applying a set of test cases to an implementation. Test cases are processes satisfying additional properties.

A test process has finite behavior such that testing can be eventually stopped. In the case of divergence one needs to interrupt testing externally.

\[ \text{TC1} \quad P(tr, tr') = P \land (\exists n \in \mathbb{N} \bullet \text{length}(tr' - tr) \leq n) \]

This healthiness condition is idempotent.

**Lemma 77** (TC1-idempotent)

\[ \text{TC1} \circ \text{TC1} = \text{TC1} \]

**Proof.**

\[ \text{TC1}(\text{TC1}(P)) = \]

\[ = \text{TC1}(P) \land (\exists n \in \mathbb{N} \bullet \text{length}(tr' - tr) \leq n) \]

\[ = P \land (\exists n \in \mathbb{N} \bullet \text{length}(tr' - tr) \leq n) \land (\exists n \in \mathbb{N} \bullet \text{length}(tr' - tr) \leq n) \]

\[ = P \land (\exists n \in \mathbb{N} \bullet \text{length}(tr' - tr) \leq n) \]

\[ = \text{TC1}(P) \]

Furthermore, \( \text{TC1} \) commutes with \( R1, R2, R3, \) and \( \text{IOCO1} \).

**Lemma 78** (commutativity-TC1-R1)

\[ \text{TC1} \circ R1 = R1 \circ \text{TC1} \]
Proof.

\[ TC_1(R_1(P)) = \]
\[ = TC_1(P \land (tr \leq tr')) \quad \text{(def. of } \text{R1}) \]
\[ = TC_1(P) \land (tr \leq tr') \quad \text{(def. of } \text{TC1}) \]
\[ = P \land (tr \leq tr') \land (\exists n \in \mathbb{N} \cdot \text{length}(tr' - tr) \leq n) \quad \text{(prop. calculus)} \]
\[ = TC_1(P) \land (tr \leq tr') \quad \text{(def. of } \text{TC1}) \]
\[ = R_1(TC_1(P)) \]

\[ \text{Lemma 79 (commutativity-TC1-R2)} \]
\[ TC_1 \circ R_2 = R_2 \circ TC_1 \]

Proof.

\[ TC_1(R_2(P(tr,tr'))) = \]
\[ = TC_1(P(\emptyset, tr' - tr)) \quad \text{(def. of } \text{R2}) \]
\[ = TC_1(P(\emptyset, tr' - tr)) \land (\exists n \in \mathbb{N} \cdot \text{length}(tr' - tr) \leq n) \quad \text{(def. of } \text{TC1}) \]
\[ = (P \land (\exists n \in \mathbb{N} \cdot \text{length}(tr' - tr) \leq n))(\emptyset, tr' - tr) \quad \text{(def. of } \text{TC1}) \]
\[ = TC_1(P)((\emptyset, tr' - tr)) \quad \text{(def. of } \text{R2}) \]
\[ = R_2(TC_1(P)) \]

\[ \text{Lemma 80 (commutativity-TC1-IOCO1)} \]
\[ TC_1 \circ IOCO_1 = IOCO_1 \circ TC_1 \]

Proof.

\[ TC_1(IOCO_1(P)) = \]
\[ = TC_1(P \land (ok' \rightarrow (\text{wait}' \lor \text{ok}')))) \quad \text{(def. of } \text{IOCO1}) \]
\[ = TC_1(P(\emptyset, tr' - tr)) \quad \text{(def. of } \text{TC1}) \]
\[ = TC_1(P(\emptyset, tr' - tr)) \land (\exists n \in \mathbb{N} \cdot \text{length}(tr' - tr) \leq n) \quad \text{(prop. calculus)} \]
\[ = (P \land (\exists n \in \mathbb{N} \cdot \text{length}(tr' - tr) \leq n))(\emptyset, tr' - tr) \quad \text{(def. of } \text{TC1}) \]
\[ = TC_1(P) \land (ok' \rightarrow (\text{wait}' \lor \text{ok}')))) \quad \text{(def. of } \text{IOCO1}) \]
\[ = IOCO_1(TC_1(P)) \]

Furthermore, a test case either accepts all responses from an implementation, i.e., inputs from the view of the test case, or it accepts no inputs at all\footnote{13}:

\[ \text{TC2} \]
\[ P = P \land (\neg \text{wait} \rightarrow (\text{wait}' \Rightarrow (ref_{in}' = A_{in} \lor ref_{in}' = \emptyset))) \]

This healthiness condition is idempotent.

\[ \text{Lemma 81 (TC2-idempotent)} \]
\[ TC_2 \circ TC_2 = TC_2 \]

Proof.

\[ TC_2(TC_2(P)) = \]
\[ = TC_2(P) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (ref_{in}' = A_{in} \lor ref_{in}' = \emptyset)))) \quad \text{(def. of } \text{TC2}) \]
\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (ref_{in}' = A_{in} \lor ref_{in}' = \emptyset)))) \land 
\[ (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (ref_{in}' = A_{in} \lor ref_{in}' = \emptyset)))) \quad \text{(prop. calculus)} \]
\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (ref_{in}' = A_{in} \lor ref_{in}' = \emptyset)))) \quad \text{(def. of } \text{TC2}) \]
\[ = TC_2(P) \]

Furthermore, \text{TC2} commutes with \text{R1}, \text{R2}, \text{R3}, \text{IOCO1} and \text{TC1}.
Lemma 82 (commutativity-TC2-R1)

\[ TC_2 \circ R_1 = R_1 \circ TC_2 \]

Proof.

\[
TC_2(R_1(P)) = \quad \text{(def. of } R_1) \\
= TC_2(P \land (tr \leq tr')) \quad \text{(def. of } TC_2) \\
= P \land (tr \leq tr') \land (\neg wait \Rightarrow (wait' \Rightarrow (ref'_in = A_{in} \lor ref'_{in} = \emptyset))) \quad \text{(prop. calculus)} \\
= P \land (\neg wait \Rightarrow (wait' \Rightarrow (ref'_in = A_{in} \lor ref'_{in} = \emptyset))) \land (tr \leq tr') \quad \text{(def. of } TC_2) \\
= TC_2(P) \land (tr \leq tr') \quad \text{(def. of } R_1) \\
= R_1(TC_2(P)) \\
\]

Lemma 83 (commutativity-TC2-R2)

\[ TC_2 \circ R_2 = R_2 \circ TC_2 \]

Proof.

\[
TC_2(R_2(P(tr, tr'))) = \quad \text{(def. of } R_2) \\
= TC_2(P(\langle \rangle, tr' - tr)) \quad \text{(def. of } TC_2) \\
= P(\langle \rangle, tr' - tr) \land (\neg wait \Rightarrow (wait' \Rightarrow (ref'_in = A_{in} \lor ref'_{in} = \emptyset))) \quad \text{(tr', tr are not used in } TC_2) \\
= (P \land (\neg wait \Rightarrow (wait' \Rightarrow (ref'_in = A_{in} \lor ref'_{in} = \emptyset))))(\langle \rangle, tr' - tr) \quad \text{(def. of } TC_2) \\
= TC_2(P)(\langle \rangle, tr' - tr) \quad \text{(def. of } R_2) \\
= R_2(TC_2(P)) \\
\]

Lemma 84 (commutativity-TC2-IOCO1)

\[ TC_2 \circ IOCO_1 = IOCO_1 \circ TC_2 \]

Proof.

\[
TC_2(IOCO_1(P)) = \quad \text{(def. of } IOCO_1) \\
= TC_2(P \land (ok \Rightarrow (wait' \lor ok'))) \quad \text{(def. of } TC_2) \\
= P \land (ok \Rightarrow (wait' \lor ok')) \land (\neg wait \Rightarrow (wait' \Rightarrow (ref'_in = A_{in} \lor ref'_{in} = \emptyset))) \quad \text{(prop. calculus)} \\
= P \land (\neg wait \Rightarrow (wait' \Rightarrow (ref'_in = A_{in} \lor ref'_{in} = \emptyset))) \land (ok \Rightarrow (wait' \lor ok')) \quad \text{(def. of } TC_2) \\
= TC_2(P) \land (ok \Rightarrow (wait' \lor ok')) \quad \text{(def. of } IOCO_1) \\
= IOCO_1(TC_2(P)) \\
\]

Lemma 85 (commutativity-TC2-TC1)

\[ TC_2 \circ TC_1 = TC_1 \circ TC_2 \]

Proof.

\[
TC_2(TC_1(P)) = \quad \text{(def. of } TC_1) \\
= TC_2(P \land (\exists n \in N \bullet length(tr' - tr) \leq n)) \quad \text{(def. of } TC_2) \\
= P \land (\exists n \in N \bullet length(tr' - tr) \leq n) \land (\exists n \in N \bullet length(tr' - tr) \leq n) \quad \text{(prop. calculus)} \\
= P \land (\exists n \in N \bullet length(tr' - tr) \leq n) \land (\exists n \in N \bullet length(tr' - tr) \leq n) \quad \text{(def. of } TC_2) \\
= TC_2(P) \land (\exists n \in N \bullet length(tr' - tr) \leq n) \quad \text{(def. of } TC_1) \\
= TC_1(TC_2(P)) \\
\]
If the test case has to provide a particular stimuli to the IUT it is always clear which output (from the view of the test case) should be send\textsuperscript{13}:

\[
\text{TC3} \quad P = P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|r_{\text{ref}_{\text{out}}}'| \geq |A_{\text{out}}| - 1)))
\]

This healthiness condition is idempotent.

**Lemma 86 (TC3-idempotent)**

\[
\text{TC3} \circ \text{TC3} = \text{TC3}
\]

**Proof.**

\[
\begin{align*}
\text{TC3}(\text{TC3}(P)) &= \quad \text{(def. of TC3)} \\
= \text{TC3}(P) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|r_{\text{ref}_{\text{out}}}'| \geq |A_{\text{out}}| - 1))) & \quad \text{(def. of TC3)} \\
= P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|r_{\text{ref}_{\text{out}}}'| \geq |A_{\text{out}}| - 1))) \land \\
(\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|r_{\text{ref}_{\text{out}}}'| \geq |A_{\text{out}}| - 1))) & \quad \text{(prop. calculus)} \\
= P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|r_{\text{ref}_{\text{out}}}'| \geq |A_{\text{out}}| - 1))) & \quad \text{(def. of TC3)} \\
= \text{TC3}(P) & \quad \text{(def. of R1)}
\end{align*}
\]

Furthermore, TC3 commutes with R1, R2, R3, IOCO1, TC1 and TC2.

**Lemma 87 (commutativity-TC3-R1)**

\[
\text{TC3} \circ \text{R1} = \text{R1} \circ \text{TC3}
\]

**Proof.**

\[
\begin{align*}
\text{TC3}(\text{R1}(P)) &= \quad \text{(def. of R1)} \\
= \text{TC3}(P \land (tr \leq tr')) & \quad \text{(def. of TC3)} \\
= P \land (tr \leq tr') \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|r_{\text{ref}_{\text{out}}}'| \geq |A_{\text{out}}| - 1))) & \quad \text{(prop. calculus)} \\
= P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|r_{\text{ref}_{\text{out}}}'| \geq |A_{\text{out}}| - 1))) \land (tr \leq tr') & \quad \text{(def. of TC3)} \\
= \text{TC3}(P) \land (tr \leq tr') & \quad \text{(def. of R1)} \\
= \text{R1}(\text{TC3}(P)) & \quad \text{\Box}
\end{align*}
\]

**Lemma 88 (commutativity-TC3-R2)**

\[
\text{TC3} \circ \text{R2} = \text{R2} \circ \text{TC3}
\]

**Proof.**

\[
\begin{align*}
\text{TC3}(\text{R2}(P(tr, tr'))) &= \quad \text{(def. of R2)} \\
= \text{TC3}(P(\langle \rangle, tr' - tr)) & \quad \text{(def. of TC3)} \\
= P(\langle \rangle, tr' - tr) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|r_{\text{ref}_{\text{out}}}'| \geq |A_{\text{out}}| - 1))) & \quad \text{(tr',tr are not used in TC3)} \\
= (P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|r_{\text{ref}_{\text{out}}}'| \geq |A_{\text{out}}| - 1))))(\langle \rangle, tr' - tr) & \quad \text{(def. of TC3)} \\
= \text{TC3}(P)(\langle \rangle, tr' - tr) & \quad \text{(def. of R2)} \\
= \text{R2}(\text{TC3}(P)) & \quad \text{\Box}
\end{align*}
\]

**Lemma 89 (commutativity-TC3-IOCO1)**

\[
\text{TC3} \circ \text{IOCO1} = \text{IOCO1} \circ \text{TC3}
\]
Proof.

\[ \text{TC3}(\text{IOCO1}(P)) = \]
\[ = \text{TC3}(P \land (\text{ok} \Rightarrow (\text{wait'} \lor \text{ok'}))) \quad \text{(def. of IOCO1)} \]
\[ = P \land (\text{ok} \Rightarrow (\text{wait'} \lor \text{ok'})) \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (|ref_{out}'| \geq |A_{out}| - 1))) \quad \text{(def. of TC3)} \]
\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (|ref_{out}'| \geq |A_{out}| - 1))) \land (\text{ok} \Rightarrow (\text{wait'} \lor \text{ok'})) \quad \text{(def. of IOCO1)} \]
\[ = \text{TC3}(P) \land (\text{ok} \Rightarrow (\text{wait'} \lor \text{ok'})) \quad \text{(def. of IOCO1)} \]
\[ = \text{IOCO1}(\text{TC3}(P)) \]

Lemma 90 (commutativity-TC3-TC1)

\[ \text{TC3} \circ \text{TC1} = \text{TC1} \circ \text{TC3} \]

Proof.

\[ \text{TC3}(\text{TC1}(P)) = \]
\[ = \text{TC3}(P \land (\exists n \in \mathbb{N} \cdot \text{length}(tr' - tr) \leq n)) \quad \text{(def. of TC1)} \]
\[ = P \land (\exists n \in \mathbb{N} \cdot \text{length}(tr' - tr) \leq n) \land \]
\[ (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (|ref_{out}'| \geq |A_{out}| - 1))) \quad \text{(prop. calculus)} \]
\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (|ref_{out}'| \geq |A_{out}| - 1))) \land \]
\[ (\exists n \in \mathbb{N} \cdot \text{length}(tr' - tr) \leq n) \quad \text{(def. of TC3)} \]
\[ = \text{TC3}(P) \land (\exists n \in \mathbb{N} \cdot \text{length}(tr' - tr) \leq n) \quad \text{(def. of TC1)} \]
\[ = \text{TC1}(\text{TC3}(P)) \]

Lemma 91 (commutativity-TC3-TC2)

\[ \text{TC3} \circ \text{TC2} = \text{TC2} \circ \text{TC3} \]

Proof.

\[ \text{TC3}(\text{TC2}(P)) = \]
\[ = \text{TC3}(P \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (ref_{in}' = A_{in} \lor ref_{in}' = \emptyset)))) \quad \text{(def. of TC2)} \]
\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (ref_{in}' = A_{in} \lor ref_{in}' = \emptyset))) \land \]
\[ (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (|ref_{out}'| \geq |A_{out}| - 1))) \quad \text{(prop. calculus)} \]
\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (|ref_{out}'| \geq |A_{out}| - 1))) \land \]
\[ (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (ref_{in}' = A_{in} \lor ref_{in}' = \emptyset))) \quad \text{(def. of TC3)} \]
\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (ref_{in}' = A_{in} \lor ref_{in}' = \emptyset))) \land \]
\[ (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (ref_{in}' = A_{in} \lor ref_{in}' = \emptyset))) \quad \text{(def. of TC2)} \]
\[ = \text{TC2}(\text{TC3}(P)) \]

Furthermore, testing should be a deterministic activity, i.e. test cases should be deterministic\textsuperscript{13}. Determinism includes that a tester can always deterministically decide what to do: send a particular stimuli to the IUT or wait for a possible response\textsuperscript{14}. This is ensured by the following two healthiness conditions.

\[ \text{TC4} \quad P = P \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow ((|ref_{out}'| = |A_{out}| - 1) \iff ref_{in}' = A_{in}))) \]
\[ \text{TC5} \quad P = P \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow ((|ref_{in}' = \emptyset) \iff ref_{out}' = A_{out}))) \]

These two healthiness conditions are idempotent.

Lemma 92 (TC4-idempotent)

\[ \text{TC4} \circ \text{TC4} = \text{TC4} \]

\textsuperscript{14}1st column on page 115 of \cite{Tre96}
Proof.

\[ \text{TC4}(\text{TC4}(P)) = \]
\[ = \text{TC4}(P) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|\text{ref}_{\text{out}}'| = |\text{A}_{\text{out}}| - 1) \Leftrightarrow \text{ref}_{\text{in}}' = \text{A}_{\text{in}}))) \quad \text{(def. of TC4)} \]
\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|\text{ref}_{\text{out}}'| = |\text{A}_{\text{out}}| - 1) \Leftrightarrow \text{ref}_{\text{in}}' = \text{A}_{\text{in}}))) \land \]
\[ (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|\text{ref}_{\text{out}}'| = |\text{A}_{\text{out}}| - 1) \Leftrightarrow \text{ref}_{\text{in}}' = \text{A}_{\text{in}}))) \quad \text{(prop. calculus)} \]
\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|\text{ref}_{\text{out}}'| = |\text{A}_{\text{out}}| - 1) \Leftrightarrow \text{ref}_{\text{in}}' = \text{A}_{\text{in}}))) \quad \text{(def. of TC4)} \]
\[ = \text{TC4}(P) \]

Lemma 93 (TC5-idempotent)
\[ \text{TC5} \circ \text{TC5} = \text{TC5} \]

Proof.

\[ \text{TC5}(\text{TC5}(P)) = \]
\[ = \text{TC5}(P) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|\text{ref}_{\text{out}}'| = |\text{A}_{\text{out}}| - 1) \Leftrightarrow \text{ref}_{\text{in}}' = \text{A}_{\text{in}}))) \quad \text{(def. of TC5)} \]
\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|\text{ref}_{\text{out}}'| = |\text{A}_{\text{out}}| - 1) \Leftrightarrow \text{ref}_{\text{in}}' = \text{A}_{\text{in}}))) \land \]
\[ (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|\text{ref}_{\text{out}}'| = |\text{A}_{\text{out}}| - 1) \Leftrightarrow \text{ref}_{\text{in}}' = \text{A}_{\text{in}}))) \quad \text{(prop. calculus)} \]
\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|\text{ref}_{\text{out}}'| = |\text{A}_{\text{out}}| - 1) \Leftrightarrow \text{ref}_{\text{in}}' = \text{A}_{\text{in}}))) \quad \text{(def. of TC5)} \]
\[ = \text{TC5}(P) \]

Furthermore, both commute with \textbf{R1}, \textbf{R2}, \textbf{R3}, \textbf{IOCO1}, \textbf{TC1}, \textbf{TC2} and with each other.

Lemma 94 (commutativity-TC4-R1)
\[ \text{TC4} \circ \text{R1} = \text{R1} \circ \text{TC4} \]

Proof.

\[ \text{TC4}(\text{R1}(P)) = \]
\[ = \text{TC4}(P \land (\text{tr} \leq \text{tr}')) \quad \text{(def. of R1)} \]
\[ = P \land (\text{tr} \leq \text{tr}') \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|\text{ref}_{\text{out}}'| = |\text{A}_{\text{out}}| - 1) \Leftrightarrow \text{ref}_{\text{in}}' = \text{A}_{\text{in}}) \land (\text{tr} \leq \text{tr}')) \quad \text{(def. of TC4)} \]
\[ = \text{R1}(\text{TC4}(P)) \]

Lemma 95 (commutativity-TC4-R2)
\[ \text{TC4} \circ \text{R2} = \text{R2} \circ \text{TC4} \]

Proof.

\[ \text{TC4}(\text{R2}(P(\text{tr}, \text{tr}'))) = \]
\[ = \text{TC4}(P(\text{tr}, \text{tr}')) \quad \text{(def. of R2)} \]
\[ = P(\text{tr}, \text{tr}' - \text{tr}) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|\text{ref}_{\text{out}}'| = |\text{A}_{\text{out}}| - 1) \Leftrightarrow \text{ref}_{\text{in}}' = \text{A}_{\text{in}}))) \quad \text{(tr',tr are not used in TC4)} \]
\[ = (P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow (|\text{ref}_{\text{out}}'| = |\text{A}_{\text{out}}| - 1) \Leftrightarrow \text{ref}_{\text{in}}' = \text{A}_{\text{in}})))(\text{tr}', \text{tr} - \text{tr}) \quad \text{(def. of TC4)} \]
\[ = \text{R2}(\text{TC4}(P)) \]

Lemma 96 (commutativity-TC4-IOCO1)
\[ \text{TC4} \circ \text{IOCO1} = \text{IOCO1} \circ \text{TC4} \]
Lemma 99 (commutativity-TC4-TC3)

Proof.

\[ \text{TC4(IOC01(P))} = \]
\[ = \text{TC4}(P \land (ok \Rightarrow (wait' \lor ok'))) \]
\[ = P \land (ok \Rightarrow (wait' \lor ok')) \land \]
\[ (\neg wait \Rightarrow (wait' \Rightarrow (|ref'_{out}| = |A_{out}| - 1) \Leftrightarrow ref'_{in} = A_{in}))) \]
\[ = P \land (\neg wait \Rightarrow (wait' \Rightarrow (|ref'_{out}| = |A_{out}| - 1) \Leftrightarrow ref'_{in} = A_{in}))) \land \]
\[ (ok \Rightarrow (wait' \lor ok')) \]
\[ = \text{TC4}(P) \land (ok \Rightarrow (wait' \lor ok')) \]
\[ = \text{IOC01}(\text{TC4}(P)) \]

Lemma 97 (commutativity-TC4-TC1)

\[ \text{TC4} \circ \text{TC1} = \text{TC1} \circ \text{TC4} \]

Proof.

\[ \text{TC4(TC1(P))} = \]
\[ = \text{TC4}(P \land (\exists n \in \mathbb{N} \cdot length(tr' - tr) \leq n)) \]
\[ = P \land (\exists n \in \mathbb{N} \cdot length(tr' - tr) \leq n) \land \]
\[ (\neg wait \Rightarrow (wait' \Rightarrow (|ref'_{out}| = |A_{out}| - 1) \Leftrightarrow ref'_{in} = A_{in}))) \]
\[ = P \land (\neg wait \Rightarrow (wait' \Rightarrow (|ref'_{out}| = |A_{out}| - 1) \Leftrightarrow ref'_{in} = A_{in}))) \land \]
\[ (\exists n \in \mathbb{N} \cdot length(tr' - tr) \leq n) \]
\[ = \text{TC1}(\text{TC4}(P)) \]

Lemma 98 (commutativity-TC4-TC2)

\[ \text{TC4} \circ \text{TC2} = \text{TC2} \circ \text{TC4} \]

Proof.

\[ \text{TC4(TC2(P))} = \]
\[ = \text{TC4}(P \land (\neg wait \Rightarrow (wait' \Rightarrow (ref'_{in} = A_{in} \lor ref'_{in} = \emptyset)))) \]
\[ = P \land (\neg wait \Rightarrow (wait' \Rightarrow (ref'_{in} = A_{in} \lor ref'_{in} = \emptyset))) \land \]
\[ (\neg wait \Rightarrow (wait' \Rightarrow (|ref'_{out}| = |A_{out}| - 1) \Leftrightarrow ref'_{in} = A_{in}))) \]
\[ = P \land (\neg wait \Rightarrow (wait' \Rightarrow (|ref'_{out}| = |A_{out}| - 1) \Leftrightarrow ref'_{in} = A_{in}))) \land \]
\[ (\neg wait \Rightarrow (wait' \Rightarrow (ref'_{in} = A_{in} \lor ref'_{in} = \emptyset))) \]
\[ = \text{TC2}(\text{TC4}(P)) \]

Lemma 99 (commutativity-TC4-TC3)

\[ \text{TC4} \circ \text{TC3} = \text{TC3} \circ \text{TC4} \]
Proof.

\[ TC4(TC3(P)) = \]
\[ = TC4(P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow ((\lceil ref'_\text{out} \rceil \geq |A_{\text{out}}| - 1)))))) \]  
\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow ((\lceil ref'_\text{out} \rceil \geq |A_{\text{out}}| - 1)))) \land \]
\[ (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow ((|ref'_\text{out}| = |A_{\text{out}}| - 1) \iff ref'_\text{in} = A_{\text{in}})))) \]  
\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow ((|ref'_\text{out}| = |A_{\text{out}}| - 1) \iff ref'_\text{in} = A_{\text{in}})))) \land \]
\[ (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow ((\lceil ref'_\text{out} \rceil \geq |A_{\text{out}}| - 1)))) \]  
\[ = TC4(P) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow ((\lceil ref'_\text{out} \rceil \geq |A_{\text{out}}| - 1)))) \]  
\[ = TC3(TC4(P)) \]

Lemma 100 (commutativity-TC5-R1)

\[ TC5 \circ R1 = R1 \circ TC5 \]

Proof.

\[ TC5(R1(P)) = \]
\[ = TC5(P \land (tr \leq tr')) \]  
\[ = P \land (tr \leq tr') \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow ((\lceil ref'_\text{in} \rceil = |A_{\text{in}}| - 1) \iff ref'_\text{out} = A_{\text{out}})))) \]  
\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow ((\lceil ref'_\text{in} \rceil = |A_{\text{in}}| - 1) \iff ref'_\text{out} = A_{\text{out}})))) \land (tr \leq tr') \]  
\[ = TC5(P) \land (tr \leq tr') \]  
\[ = R1(TC5(P)) \]

Lemma 101 (commutativity-TC5-R2)

\[ TC5 \circ R2 = R2 \circ TC5 \]

Proof.

\[ TC5(R2(P(tr,tr'))) = \]
\[ = TC5(P(\langle \rangle, tr' - tr)) \]  
\[ = P(\langle \rangle, tr' - tr) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow ((\lceil ref'_\text{in} \rceil = |A_{\text{in}}| - 1) \iff ref'_\text{out} = A_{\text{out}})))) \]  
\[ = P(\langle \rangle, tr' - tr) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow ((\lceil ref'_\text{in} \rceil = |A_{\text{in}}| - 1) \iff ref'_\text{out} = A_{\text{out}})))) \land (tr, tr' \text{ are not used in } TC5) \]  
\[ = (P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow ((\lceil ref'_\text{in} \rceil = |A_{\text{in}}| - 1) \iff ref'_\text{out} = A_{\text{out}}))))(\langle \rangle, tr' - tr) \]  
\[ = TC5(P)(\langle \rangle, tr' - tr) \]  
\[ = R2(TC5(P)) \]

Lemma 102 (commutativity-TC5-IOCO1)

\[ TC5 \circ IOCO1 = IOCO1 \circ TC5 \]

Proof.

\[ TC5(IOCO1(P)) = \]
\[ = TC5(P \land (ok \Rightarrow (\text{wait}' \lor ok')))) \]  
\[ = P \land (ok \Rightarrow (\text{wait}' \lor ok')) \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow ((\lceil ref'_\text{in} \rceil = |A_{\text{in}}| - 1) \iff ref'_\text{out} = A_{\text{out}})))) \]  
\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait}' \Rightarrow ((\lceil ref'_\text{in} \rceil = |A_{\text{in}}| - 1) \iff ref'_\text{out} = A_{\text{out}})))) \land (ok \Rightarrow (\text{wait}' \lor ok')) \]  
\[ = TC5(P) \land (ok \Rightarrow (\text{wait}' \lor ok')) \]  
\[ = IOCO1(TC5(P)) \]
Lemma 103 (commutativity-TC5-TC1)

\[ TC_5 \circ TC_1 = TC_1 \circ TC_5 \]

Proof.

\[
TC_5(TC_1(P)) =
\]

\[ = TC_5(P \land (\exists n \in \mathbb{N} \cdot \text{length}(tr' - tr) \leq n)) \]  

\[ = P \land (\exists n \in \mathbb{N} \cdot \text{length}(tr' - tr) \leq n) \wedge \]

\[ (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow ((ref'_in = \emptyset) \Leftrightarrow ref'_out = A_{out}))) \]  

\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow ((ref'_in = \emptyset) \Leftrightarrow ref'_out = A_{out}))) \wedge \]

\[ (\exists n \in \mathbb{N} \cdot \text{length}(tr' - tr) \leq n) \]  

\[ = TC_5(P) \land (\exists n \in \mathbb{N} \cdot \text{length}(tr' - tr) \leq n) \]  

\[ = TC_1(TC_5(P)) \]  

\[
\square
\]

Lemma 104 (commutativity-TC5-TC2)

\[ TC_5 \circ TC_2 = TC_2 \circ TC_5 \]

Proof.

\[
TC_5(TC_2(P)) =
\]

\[ = TC_5(P \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (ref'_in = A_{in} \lor ref'_in = \emptyset)))) \]  

\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (ref'_in = A_{in} \lor ref'_in = \emptyset))) \wedge \]

\[ (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow ((ref'_in = \emptyset) \Leftrightarrow ref'_out = A_{out}))) \]  

\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow ((ref'_in = \emptyset) \Leftrightarrow ref'_out = A_{out}))) \wedge \]

\[ (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (ref'_in = A_{in} \lor ref'_in = \emptyset))) \]  

\[ = TC_5(P) \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (ref'_in = A_{in} \lor ref'_in = \emptyset))) \]  

\[ = TC_2(TC_5(P)) \]  

\[
\square
\]

Lemma 105 (commutativity-TC5-TC3)

\[ TC_5 \circ TC_3 = TC_3 \circ TC_5 \]

Proof.

\[
TC_5(TC_3(P)) =
\]

\[ = TC_5(P \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (|ref'_out| \geq |A_{out}| - 1)))) \]  

\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (|ref'_out| \geq |A_{out}| - 1))) \wedge \]

\[ (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow ((ref'_in = \emptyset) \Leftrightarrow ref'_out = A_{out}))) \]  

\[ = P \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow ((ref'_in = \emptyset) \Leftrightarrow ref'_out = A_{out}))) \wedge \]

\[ (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (|ref'_out| \geq |A_{out}| - 1))) \]  

\[ = TC_5(P) \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (|ref'_out| \geq |A_{out}| - 1))) \]  

\[ = TC_3(TC_5(P)) \]  

\[
\square
\]

Lemma 106 (commutativity-TC5-TC4)

\[ TC_5 \circ TC_4 = TC_4 \circ TC_5 \]
Proof.

\[ TC5(\text{TC4}(P)) = \]
\[ = TC5(P \land (\neg\text{wait} \Rightarrow (\text{pass'} \Rightarrow \neg\text{fail'}))) \]
\[ = P \land (\neg\text{wait} \Rightarrow (\text{pass'} \Rightarrow (|(\text{ref}'_{\text{out}} | = |A_{\text{out}}| - 1) \Leftrightarrow \text{ref}'_{\text{in}} = A_{\text{in}})))) \land \]
\[ (\neg\text{wait} \Rightarrow (\text{wait'} \Rightarrow (|(\text{ref}'_{\text{out}} | = |A_{\text{out}}| - 1) \Leftrightarrow \text{ref}'_{\text{in}} = A_{\text{in}})))) \]
\[ = TC5(P) \land (\neg\text{wait} \Rightarrow (\text{pass'} \Rightarrow \neg\text{fail'})) \]
\[ = TC4(TC5(P)) \] (def. of TC5

After termination a test case should give a verdict about the test execution

\[ TC6 \]
\[ P = P \land (\neg\text{wait'} \Rightarrow (\text{pass'} \Rightarrow \neg\text{fail'})) \]
This healthiness condition is idempotent.

Lemma 107 (TC6-idempotent)

\[ TC6 \circ TC6 = TC6 \]

Proof.

\[ TC6(TC6(P)) = \]
\[ = TC6(P) \land (\neg\text{wait} \Rightarrow (\text{pass'} \Rightarrow \neg\text{fail'})) \]
\[ = P \land (\neg\text{wait} \Rightarrow (\text{pass'} \Rightarrow \neg\text{fail'})) \land (\neg\text{wait} \Rightarrow (\text{pass'} \Rightarrow \neg\text{fail'})) \]
\[ = P \land (\neg\text{wait} \Rightarrow (\text{pass'} \Rightarrow \neg\text{fail'})) \]
\[ = TC6(P) \] (def. of TC6

Furthermore, TC6 commutes with R1, R2, R3, IOCO1, TC1 and TC2.

Lemma 108 (commutativity-TC6-R1)

\[ TC6 \circ R1 = R1 \circ TC6 \]

Proof.

\[ TC6(R1(P)) = \]
\[ = TC6(P \land (\text{tr} \leq \text{tr}')) \]
\[ = P \land (\text{tr} \leq \text{tr}') \land (\neg\text{wait} \Rightarrow (\text{pass'} \Rightarrow \neg\text{fail'})) \]
\[ = P \land (\neg\text{wait'} \Rightarrow (\text{pass'} \Rightarrow \neg\text{fail'})) \land (\text{tr} \leq \text{tr}') \]
\[ = TC6(P) \land (\text{tr} \leq \text{tr}') \]
\[ = R1(TC6(P)) \] (def. of R1

Lemma 109 (commutativity-TC6-R2)

\[ TC6 \circ R2 = R2 \circ TC6 \]

Proof.

\[ TC6(R2(P(\text{tr}, \text{tr}'))) = \]
\[ = TC6(P(\langle \rangle, \text{tr'} - \text{tr})) \]
\[ = P(\langle \rangle, \text{tr'} - \text{tr}) \land (\neg\text{wait'} \Rightarrow (\text{pass'} \Rightarrow \neg\text{fail'})) \]
\[ = (P \land (\neg\text{wait'} \Rightarrow (\text{pass'} \Rightarrow \neg\text{fail'})))(\langle \rangle, \text{tr'} - \text{tr}) \]
\[ = TC6(P)(\langle \rangle, \text{tr'} - \text{tr}) \]
\[ = R2(TC6(P)) \] (def. of R2
Lemma 110 (commutativity-TC6-IOCO1)

\[ TC6 \circ IOCO1 = IOCO1 \circ TC6 \]

Proof.

\[
\begin{align*}
TC6(IOCO1(P)) &= \\
&= TC6(P \land (ok \Rightarrow (wait' \lor ok')))) \quad \text{(def. of IOCO1)} \\
&= P \land (ok \Rightarrow (wait' \lor ok')) \land (\neg wait' \Rightarrow (pass' \Rightarrow \neg fail')) \quad \text{(def. of TC6)} \\
&= TC6(P) \land (ok \Rightarrow (wait' \lor ok')) \quad \text{(def. of IOCO1)} \\
&= IOCO1(TC6(P)) \\
\end{align*}
\]

Lemma 111 (commutativity-TC6-TC1)

\[ TC6 \circ TC1 = TC1 \circ TC6 \]

Proof.

\[
\begin{align*}
TC6(TC1(P)) &= \\
&= TC6(P \land (\exists n \in \mathbb{N} \cdot length(tr' - tr) \leq n)) \quad \text{(def. of TC1)} \\
&= P \land (\exists n \in \mathbb{N} \cdot length(tr' - tr) \leq n) \land (\neg wait' \Rightarrow (pass' \Rightarrow \neg fail')) \quad \text{(def. of TC6)} \\
&= TC6(P) \land (\exists n \in \mathbb{N} \cdot length(tr' - tr) \leq n) \quad \text{(def. of TC1)} \\
&= TC1(TC6(P)) \\
\end{align*}
\]

Lemma 112 (commutativity-TC6-TC2)

\[ TC6 \circ TC2 = TC2 \circ TC6 \]

Proof.

\[
\begin{align*}
TC6(TC2(P)) &= \\
&= TC6(P \land (\neg wait' \Rightarrow (wait' \Rightarrow (ref'_{in} = A_{in} \lor ref'_{in} = \emptyset)))) \quad \text{(def. of TC2)} \\
&= P \land (\neg wait' \Rightarrow (wait' \Rightarrow (ref'_{in} = A_{in} \lor ref'_{in} = \emptyset))) \land \\
&\quad (\neg wait' \Rightarrow (pass' \Rightarrow \neg fail')) \quad \text{(def. of TC6)} \\
&= TC6(P) \land (\neg wait' \Rightarrow (wait' \Rightarrow (ref'_{in} = A_{in} \lor ref'_{in} = \emptyset))) \quad \text{(def. of TC2)} \\
&= TC2(TC6(P)) \\
\end{align*}
\]

Lemma 113 (commutativity-TC6-TC3)

\[ TC6 \circ TC3 = TC3 \circ TC6 \]

Proof.

\[
\begin{align*}
TC6(TC3(P)) &= \\
&= TC6(P \land (\neg wait' \Rightarrow (wait' \Rightarrow (ref'_{out} \geq |A_{out}|-1)))) \quad \text{(def. of TC3)} \\
&= P \land (\neg wait' \Rightarrow (wait' \Rightarrow (ref'_{out} \geq |A_{out}|-1))) \land (\neg wait' \Rightarrow (pass' \Rightarrow \neg fail')) \quad \text{(def. of TC6)} \\
&= TC6(P) \land (\neg wait' \Rightarrow (wait' \Rightarrow (ref'_{out} \geq |A_{out}|-1))) \quad \text{(def. of TC3)} \\
&= TC3(TC6(P)) \\
\end{align*}
\]
Lemma 114 (commutativity-TC6-TC4)

\[ TC6 \circ TC4 = TC4 \circ TC6 \]

Proof.

\[
TC6( TC4(P) ) =
\]

\[
= TC6( P \land (\neg \text{wait} \Rightarrow (\lnot \text{ref}_{\text{out}} = |A_{\text{out}}| - 1) \Leftrightarrow \text{ref}_{\text{in}} = A_{\text{in}}))) \quad (\text{def. of TC4})
\]

\[
= P \land (\neg \text{wait} \Rightarrow (\lnot \text{ref}_{\text{out}} = |A_{\text{out}}| - 1) \Leftrightarrow \text{ref}_{\text{in}} = A_{\text{in}})) \land
\]

\[
(\neg \text{wait} \Rightarrow (\text{pass'} \Rightarrow \text{fail'})) \quad (\text{prop. calculus})
\]

\[
= P \land (\neg \text{wait} \Rightarrow (\lnot \text{ref}_{\text{out}} = |A_{\text{out}}| - 1) \Leftrightarrow \text{ref}_{\text{in}} = A_{\text{in}})) \land
\]

\[
= TC6( P ) \land (\neg \text{wait} \Rightarrow (\lnot \text{ref}_{\text{out}} = |A_{\text{out}}| - 1) \Leftrightarrow \text{ref}_{\text{in}} = A_{\text{in}})) \quad (\text{def. of TC4})
\]

As for specifications and implementations we make \( I \) suitable for test cases:

\[
I^{TC} = \{ -\text{ok} \land (\text{tr} \leq \text{tr}') \land (\exists n \in \mathbb{N} \cdot \text{length}(\text{tr}' - \text{tr}) \leq n) \land
\]

\[
(\text{wait'} \Rightarrow (\text{ref}_{\text{in}} = A_{\text{in}} \lor \text{ref}_{\text{in}}' = \emptyset)) \land
\]

\[
(\text{wait'} \Rightarrow ((\text{ref}_{\text{out}}' \geq |A_{\text{out}}| - 1))) \land
\]

\[
(\text{wait'} \Rightarrow ((\text{ref}_{\text{out}}' = |A_{\text{out}}| - 1) \Rightarrow \text{ref}_{\text{in}}' = A_{\text{in}})) \land
\]

\[
(\text{wait'} \Rightarrow ((\text{ref}_{\text{in}}' = \emptyset) \Rightarrow \text{ref}_{\text{out}}' = A_{\text{out}})) \land
\]

\[
(\neg \text{wait'} \Rightarrow (\text{pass'} \Rightarrow \text{fail'})) \}
\]

\( I^{TC} \) ensures that even in the case of divergence we respect the properties of test cases. For test cases \( R3 \) becomes \( R3^{TC} =_{\text{def}} I^{TC} \triangleleft \text{wait} \triangleright P \).

Given \( I^{TC} \) the healthiness condition \( R3 \) for test cases becomes \( R3^{TC} =_{\text{def}} I^{TC} \triangleleft \text{wait} \triangleright P \). All test case healthiness conditions commute with \( R3^{TC} \).

Lemma 115 (commutativity-TC6-TC5)

\[ TC6 \circ TC5 = TC5 \circ TC6 \]

Proof.

\[
TC6( TC5(P) ) =
\]

\[
= TC6( P \land (\neg \text{wait} \Rightarrow ((\text{ref}_{\text{out}} = \emptyset) \Leftrightarrow \text{ref}_{\text{out}} = A_{\text{out}}))) \quad (\text{def. of TC5})
\]

\[
= P \land (\neg \text{wait} \Rightarrow ((\text{ref}_{\text{out}} = \emptyset) \Leftrightarrow \text{ref}_{\text{out}} = A_{\text{out}})) \land
\]

\[
(\neg \text{wait} \Rightarrow (\text{pass'} \Rightarrow \text{fail'})) \quad (\text{prop. calculus})
\]

\[
= P \land (\neg \text{wait} \Rightarrow ((\text{ref}_{\text{out}} = \emptyset) \Leftrightarrow \text{ref}_{\text{out}} = A_{\text{out}})) \land
\]

\[
= TC6( P ) \land (\neg \text{wait} \Rightarrow ((\text{ref}_{\text{out}} = \emptyset) \Leftrightarrow \text{ref}_{\text{out}} = A_{\text{out}})) \quad (\text{def. of TC6})
\]

\[
= TC5( TC6(P) )
\]
Lemma 117 (commutativity-TC2-R3 TC)

\[ \text{TC}_2 \circ \text{R}_3^{\text{TC}} = \text{R}_3^{\text{TC}} \circ \text{TC}_2 \]

Proof.
\[ \text{TC}_2(\text{R}_3^{\text{TC}}(P)) = \]
\[ = \text{TC}_2(\text{R}_3^{\text{TC}} \circ \text{wait} \circ P) \]
\[ = (\text{if}^{\text{TC}} \circ \text{wait} \circ P) \land (\exists n \in N \cdot \text{length}(tr' - tr) \leq n) \]
\[ = (\exists n \in N \cdot \text{length}(tr' - tr) \leq n) \land \text{wait} \]
\[ = (\exists n \in N \cdot \text{length}(tr' - tr) \leq n) \land \text{wait} \]
\[ = \text{TC}_2(\text{R}_3^{\text{TC}}(P)) \]
\[ = \text{R}_3^{\text{TC}}(\text{TC}_2(P)) \]
\[ = \text{R}_3^{\text{TC}}(\text{TC}_2(P)) \]

Lemma 118 (commutativity-TC3-R3 TC)

\[ \text{TC}_3 \circ \text{R}_3^{\text{TC}} = \text{R}_3^{\text{TC}} \circ \text{TC}_3 \]

Proof.
\[ \text{TC}_3(\text{R}_3^{\text{TC}}(P)) = \]
\[ = \text{TC}_3(\text{R}_3^{\text{TC}} \circ \text{wait} \circ P) \]
\[ = (\text{if}^{\text{TC}} \circ \text{wait} \circ P) \land (\exists n \in N \cdot \text{length}(tr' - tr) \leq n) \]
\[ = (\text{if}^{\text{TC}} \circ \text{wait} \circ P) \land (\exists n \in N \cdot \text{length}(tr' - tr) \leq n) \]
\[ = (\exists n \in N \cdot \text{length}(tr' - tr) \leq n) \land \text{wait} \]
\[ = \text{TC}_3(\text{R}_3^{\text{TC}}(P)) \]
\[ = \text{R}_3^{\text{TC}}(\text{TC}_3(P)) \]
\[ = \text{R}_3^{\text{TC}}(\text{TC}_3(P)) \]

□
Lemma 120 (commutativity-TC5-R3)

Proof.

\[
\begin{align*}
\text{TC}_3(\text{R}_3^{TC}(P)) & = & (\text{def. of } \text{R}_3^{TC}) \\
= & \text{TC}_3(\text{I}^{TC} \triangleright wait \triangleright P) & (\text{def. of } \text{TC}_3) \\
= & (\text{I}^{TC} \triangleright wait \triangleright P) \land (\lnot wait \Rightarrow (wait' \Rightarrow (|ref'_{out}| \geq |A_{out}| - 1))) & (\land\text{-if-distr}) \\
= & (\text{I}^{TC} \land (\lnot wait \Rightarrow (wait' \Rightarrow (|ref'_{out}| \geq |A_{out}| - 1)))) \lhd wait > \\
(P \land (\lnot wait \Rightarrow (wait' \Rightarrow (|ref'_{out}| \geq |A_{out}| - 1)))) & (\text{def. of if and } \lnot wait) \\
= & \text{I}^{TC} \lhd wait > (P \land (\lnot wait \Rightarrow (wait' \Rightarrow (|ref'_{out}| \geq |A_{out}| - 1)))) & (\text{def. of } \text{TC}_3) \\
= & \text{I}^{TC} \lhd wait > (\text{TC}_3(P)) & (\text{def. of } \text{R}_3^{TC}) \\
= & \text{R}_3^{TC}(\text{TC}_3(P))
\end{align*}
\]

Lemma 119 (commutativity-TC4-R3^{TC})

\[
\text{TC}_4 \circ \text{R}_3^{TC} = \text{R}_3^{TC} \circ \text{TC}_4
\]

Proof.

\[
\begin{align*}
\text{TC}_4(\text{R}_3^{TC}(P)) & = & (\text{def. of } \text{R}_3^{TC}) \\
= & \text{TC}_4(\text{I}^{TC} \triangleright wait \triangleright P) & (\text{def. of } \text{TC}_4) \\
= & (\text{I}^{TC} \triangleright wait \triangleright P) \land (\lnot wait \Rightarrow (wait' \Rightarrow (|ref'_{out}| = |A_{out}| - 1) \Leftrightarrow ref'_{in} = \text{A}_{in}))) & (\land\text{-if-distr}) \\
= & (\text{I}^{TC} \land (\lnot wait \Rightarrow (wait' \Rightarrow (|ref'_{out}| = |A_{out}| - 1) \Leftrightarrow ref'_{in} = \text{A}_{in}))) \lhd wait > \\
(P \land (\lnot wait \Rightarrow (wait' \Rightarrow (|ref'_{out}| = |A_{out}| - 1) \Leftrightarrow ref'_{in} = \text{A}_{in}))) & (\text{def. of if and } \lnot wait) \\
= & \text{I}^{TC} \lhd wait > (P \land (\lnot wait \Rightarrow (wait' \Rightarrow (|ref'_{out}| = |A_{out}| - 1) \Leftrightarrow ref'_{in} = \text{A}_{in})))) & (\text{def. of } \text{TC}_4) \\
= & \text{I}^{TC} \lhd wait > (\text{TC}_4(P)) & (\text{def. of } \text{R}_3^{TC}) \\
= & \text{R}_3^{TC}(\text{TC}_4(P))
\end{align*}
\]

Lemma 120 (commutativity-TC5-R3^{TC})

\[
\text{TC}_5 \circ \text{R}_3^{TC} = \text{R}_3^{TC} \circ \text{TC}_5
\]

Proof.

\[
\begin{align*}
\text{TC}_5(\text{R}_3^{TC}(P)) & = & (\text{def. of } \text{R}_3^{TC}) \\
= & \text{TC}_5(\text{I}^{TC} \triangleright wait \triangleright P) & (\text{def. of } \text{TC}_5) \\
= & (\text{I}^{TC} \triangleright wait \triangleright P) \land (\lnot wait \Rightarrow (wait' \Rightarrow ((ref'_{in} = \emptyset) \Leftrightarrow ref'_{out} = \text{A}_{out}))) & (\land\text{-if-distr}) \\
= & (\text{I}^{TC} \land (\lnot wait \Rightarrow (wait' \Rightarrow ((ref'_{in} = \emptyset) \Leftrightarrow ref'_{out} = \text{A}_{out})))) \lhd wait > \\
(P \land (\lnot wait \Rightarrow (wait' \Rightarrow ((ref'_{in} = \emptyset) \Leftrightarrow ref'_{out} = \text{A}_{out})))) & (\text{def. of if and } \lnot wait) \\
= & \text{I}^{TC} \lhd wait > (P \land (\lnot wait \Rightarrow (wait' \Rightarrow ((ref'_{in} = \emptyset) \Leftrightarrow ref'_{out} = \text{A}_{out})))) & (\text{def. of } \text{TC}_5) \\
= & \text{I}^{TC} \lhd wait > (\text{TC}_5(P)) & (\text{def. of } \text{R}_3^{TC}) \\
= & \text{R}_3^{TC}(\text{TC}_5(P))
\end{align*}
\]

Lemma 121 (commutativity-TC6-R3^{TC})

\[
\text{TC}_6 \circ \text{R}_3^{TC} = \text{R}_3^{TC} \circ \text{TC}_6
\]
Proof.

\[ \text{TC6(R3}^{\text{TC}}(P)) = \]
\[ = \text{TC6(R3}^{\text{TC}}(\text{TC} \land \lnot \text{wait} \Rightarrow P) \land (\text{wait} \Rightarrow \text{pass}' \Rightarrow \lnot \text{fail}'))) \]
\[ = (\text{TC6}^{\text{TC}}(\text{TC} \land \lnot \text{wait} \Rightarrow (\text{pass}' \Rightarrow \lnot \text{fail}')) \land \ldots \land (\text{wait} \Rightarrow (\text{pass}' \Rightarrow \lnot \text{fail}'))) \land \ldots \]
\[ = (\text{TC6}^{\text{TC}}(\text{TC} \land \lnot \text{wait} \Rightarrow (\text{pass}' \Rightarrow \lnot \text{fail}')) \land \ldots \land (\text{wait} \Rightarrow (\text{pass}' \Rightarrow \lnot \text{fail}'))) \land \ldots \]
\[ \quad \quad \quad \quad \text{def. of } \text{TC6} \text{ and prop. calculus} \]
\[ \quad \quad \quad \quad \text{prop. calculus and def. of if} \]
\[ \quad \quad \quad \quad \text{prop. calculus} \]
\[ \quad \quad \quad \quad \text{prop. calculus and def. of if} \]
\[ \quad \quad \quad \quad \text{def. of } \text{TC}^{\text{TC}} \]
\[ \quad \quad \quad \quad \text{def. of } \text{R3}^{\text{TC}} \]

**Definition 34 (Test process)** A test process \( P \) is a reactive process, which satisfies the healthiness conditions \( \text{TC1…TC6} \) and \( \text{IOCO1} \). The set of events in which a test case can potentially engage is given by \( \mathcal{A} \), where \( \mathcal{A} = \mathcal{A}_{\text{out}} \cup \mathcal{A}_{\text{in}}, \mathcal{A}_{\text{out}} \cap \mathcal{A}_{\text{in}} = \emptyset \) and \( \emptyset \in \mathcal{A}_{\text{in}} \). The observations are extended by: \( \text{pass}, \text{fail} : \text{Bool} \), which denote the pass and fail verdicts, respectively.

**Theorem 5** Test cases are closed under \( \{+;\} \).

Proof. Idempotence of healthiness conditions and healthiness conditions are preserved (see lemmas).

**Remark 4** Due to the results of Petrenko et al. [PYH03], the properties of test cases have been changed recently [Tre08]. Test cases are now input-enabled, i.e. they are not able to block any input (i.e. outputs of the IUT) anymore. Hence, test cases accept every output of the IUT in every state. Note that this conflicts with healthiness condition \( \text{TC5} \) and \( \text{TC6} \). During the test execution one has now to decide non-deterministically whether to send an input or to wait for an output. However, we use the original version of \text{ioco} in this paper.
Lemma 122 (closure-γ-TC1)

Test cases are reactive processes expressed in terms of $do_A(a)$. We use the following abbreviations for indicating pass (√) and fail (×) verdicts.

$\sqrt{\vDash} =_{df} (\neg:\text{wait}' \Rightarrow \text{pass}')$

$\times =_{df} (\neg:\text{wait}' \Rightarrow \text{fail}')$

Due to the properties of test cases the only choices of a test case are choices between inputs. Since the chosen input to a test case, i.e. output of the IUT, are up to the IUT this choice is given in terms of an external choice:

$$P + Q =_{df} P \land Q \triangleleft \delta \triangleright P \lor Q \quad \text{with} \quad \delta =_{df} R3(tr' = tr \land \text{wait}')$$

The healthiness conditions for test cases are preserved by the operators used for test cases, i.e., by sequential composition and by the external choice:

**Lemma 122 (closure-γ-TC1)**

$$TC1(P; Q) = P; Q \text{ provided } P \text{ and } Q \text{ are } TC1 \text{ healthy}$$

**Proof.**

$$TC1(P; Q) =$$

$$= TC1(\exists v_0, tr_0 \bullet P[v_0, tr_0/v', tr'] \land Q[v_0, tr_0/v, tr])$$

($$\text{def. of } :$$)

$$= \exists v_0, tr_0 \bullet P[v_0, tr_0/v', tr'] \land Q[v_0, tr_0/v, tr] \land (\exists n \in N \bullet length(tr' - tr) \leq n)$$

($$\text{def. of } TC1$$)

$$= \exists v_0, tr_0 \bullet TC1(P)[v_0, tr_0/v', tr] \land TC1(Q)[v_0, tr_0/v, tr] \land (\exists n \in N \bullet length(tr' - tr) \leq n)$$

($$\text{assumption}$$)

$$\text{Lemma 123 (closure-γ-TC2)}$$

$$TC2(P; Q) = P; Q \text{ provided that } P \text{ and } Q \text{ are } TC2 \text{ healthy and } Q \text{ is } R3^\delta \text{ healthy}$$
Proof.

\[ \text{TC2}(P; Q) = \text{(assumption and def. of R3}^{TC}) \]
\[ = \text{TC2}(P; (\exists^{TC} \land \text{wait} \Rightarrow Q)) \quad \text{(def. of ;)} \]
\[ = \text{TC2}(\exists v_0 \land P[v_0/v'] \land (\exists^{TC} \land \text{wait} \Rightarrow Q)[v_0/v]) \quad \text{(def. TC2, def. if, and substitution)} \]
\[ = \exists v_0 \land P[v_0/v'] \land (\exists^{TC} \land \text{wait} \Rightarrow Q)[v_0/v] \land \text{wait} \Rightarrow \text{wait} \land Q[v_0/v] \land (\text{wait} \lor \text{wait}' \lor \text{ref}_{in} = A_{in} \lor \text{ref}_{in}' = 0) \quad \text{(def. of } \exists^{TC} \text{ and prop. calculus)} \]
\[ = \exists v_0 \land \left( P[v_0/v'] \land \text{wait} \lor \text{ok} \land \ldots \land (\text{wait} \lor \text{ref}_{in} = A_{in} \lor \text{ref}_{in}' = 0) \lor (\text{wait} \land \text{wait}' \lor \text{ref}_{in} = 0) \lor (\text{wait} \lor \text{wait}' \lor \text{ref}_{in} = 0) \lor (\text{ref}_{in} = 0) \lor (\text{ref}_{in}' = 0) \right) \quad \text{(substitution and prop. calculus)} \]
\[ = \exists v_0 \land \left( P[v_0/v'] \land \text{wait} \lor \text{ok} \land \ldots \land (\text{wait} \lor \text{ref}_{in} = A_{in} \lor \text{ref}_{in}' = 0) \lor (\text{wait} \lor \text{wait}' \lor \text{ref}_{in} = 0) \lor (\text{ref}_{in} = 0) \lor (\text{ref}_{in}' = 0) \right) \quad \text{(prop. calculus)} \]
\[ = \exists v_0 \land \left( P[v_0/v'] \land \text{wait} \lor \text{ok} \land \ldots \land (\text{wait} \lor \text{ref}_{in} = A_{in} \lor \text{ref}_{in}' = 0) \lor (\text{wait} \lor \text{wait}' \lor \text{ref}_{in} = 0) \lor (\text{ref}_{in} = 0) \lor (\text{ref}_{in}' = 0) \right) \quad \text{(substitution)} \]
\[ = \exists v_0 \land \left( P[v_0/v'] \land \text{wait} \lor \text{ok} \land \ldots \land (\text{wait} \lor \text{ref}_{in} = A_{in} \lor \text{ref}_{in}' = 0) \lor (\text{wait} \lor \text{wait}' \lor \text{ref}_{in} = 0) \lor (\text{ref}_{in} = 0) \lor (\text{ref}_{in}' = 0) \right) \quad \text{(def. of TC2 and assumption)} \]
\[ = \exists v_0 \land \left( P[v_0/v'] \land \text{wait} \lor \text{ok} \land \ldots \land (\text{wait} \lor \text{ref}_{in} = A_{in} \lor \text{ref}_{in}' = 0) \lor (\text{wait} \lor \text{wait}' \lor \text{ref}_{in} = 0) \lor (\text{ref}_{in} = 0) \lor (\text{ref}_{in}' = 0) \right) \quad \text{(prop. calculus and def. of } \exists^{TC} \text{ and assumption)} \]
\[ = \exists v_0 \land \left( P[v_0/v'] \land \text{wait} \lor \text{ok} \land \ldots \land (\text{wait} \lor \text{ref}_{in} = A_{in} \lor \text{ref}_{in}' = 0) \lor (\text{wait} \lor \text{wait}' \lor \text{ref}_{in} = 0) \lor (\text{ref}_{in} = 0) \lor (\text{ref}_{in}' = 0) \right) \quad \text{(def. TC2 and renaming)} \]
\[ = \exists v_0 \land \left( P[v_0/v'] \land \text{wait} \lor \text{ok} \land \ldots \land (\text{wait} \lor \text{ref}_{in} = A_{in} \lor \text{ref}_{in}' = 0) \lor (\text{wait} \lor \text{wait}' \lor \text{ref}_{in} = 0) \lor (\text{ref}_{in} = 0) \lor (\text{ref}_{in}' = 0) \right) \quad \text{(prop. calculus)} \]
\[ = \exists v_0 \land \left( P[v_0/v'] \land \text{wait} \lor \text{ok} \land \ldots \land (\text{wait} \lor \text{ref}_{in} = A_{in} \lor \text{ref}_{in}' = 0) \lor (\text{wait} \lor \text{wait}' \lor \text{ref}_{in} = 0) \lor (\text{ref}_{in} = 0) \lor (\text{ref}_{in}' = 0) \right) \quad \text{(prop. calculus)} \]
\[ = \exists v_0 \land \left( P[v_0/v'] \land \text{wait} \lor \text{ok} \land \ldots \land (\text{wait} \lor \text{ref}_{in} = A_{in} \lor \text{ref}_{in}' = 0) \lor (\text{wait} \lor \text{wait}' \lor \text{ref}_{in} = 0) \lor (\text{ref}_{in} = 0) \lor (\text{ref}_{in}' = 0) \right) \quad \text{(substitution and def. of TC2)} \]
\[ = \exists v_0 \land P[v_0/v'] \land \text{wait} \land \text{ref}_{in} \lor P[v_0/v'] \land \text{wait} \land \text{ref}_{in} \quad \text{(assumption and def. of R3}^{TC}) \]
\[ = \exists v_0 \land P[v_0/v'] \land R3^{TC} \land Q[v_0/v] \quad \text{(assumption and def. of ;)} \]

\[ = P; Q \]
Lemma 124 (closure-;TC3)

\[ \text{TC3}(P; Q) = P; Q \text{ provided } Q \text{ is } \text{TC3} \text{ healthy} \]

**Proof.** Similar to the proof of Lemma 123. □

Lemma 125 (closure-;TC4)

\[ \text{TC4}(P; Q) = P; Q \text{ provided } Q \text{ is } \text{TC4} \text{ healthy} \]

**Proof.** Similar to the proof of Lemma 123. □

Lemma 126 (closure-;TC5)

\[ \text{TC5}(P; Q) = P; Q \text{ provided } Q \text{ is } \text{TC5} \text{ healthy} \]

**Proof.** Similar to the proof of Lemma 123. □

Lemma 127 (closure-;TC6)

\[ \text{TC6}(P; Q) = P; Q \text{ provided } Q \text{ is } \text{TC6} \text{ healthy} \]

**Proof.** Similar to the proof of Lemma 123. □

Lemma 128 (closure-+-IOCO1)

\[ \text{IOCO1}(P + Q) = P + Q \text{ provided } P \text{ and } Q \text{ are } \text{IOCO1} \text{ healthy} \]

**Proof.**

\[
\begin{align*}
\text{IOCO1}(P + Q) &= \quad \text{(def. of +)} \\
&= \text{IOCO1}((P \land Q) <\delta > (P \lor Q)) \quad \text{(def. of IOCO1)} \\
&= (P \land Q) <\delta > (P \lor Q) \land (ok \Rightarrow (wait' \lor ok')) \quad \text{(distr. of \land over if)} \\
&= (P \land Q \land (ok \Rightarrow (wait' \lor ok'))) <\delta > \quad \text{(prop. calculus)} \\
&= (P \land (ok \Rightarrow (wait' \lor ok'))) \land (Q \land (ok \Rightarrow (wait' \lor ok'))) <\delta > \\
&= (P \land (ok \Rightarrow (wait' \lor ok'))) \lor (Q \land (ok \Rightarrow (wait' \lor ok'))) \quad \text{(def. of +)} \\
&= \text{IOCO1}(P) + \text{IOCO1}(Q) \quad \text{(def. of IOCO1)} \\
P + Q &= \quad \text{(assumption)} \\
\end{align*}
\]

Lemma 129 (closure-+-TC1)

\[ \text{TC1}(P + Q) = P + Q \text{ provided } P \text{ and } Q \text{ are } \text{TC1} \text{ healthy} \]
Proof.

\[ TC1(P + Q) = \]
\[ = TC1((P \land Q) \triangleleft \delta \triangleright (P \lor Q)) \] (def. of +)
\[ = (P \land Q) \triangleleft \delta \triangleright (P \lor Q) \land (\exists n \in \mathbb{N} \bullet length(tr' - tr) \leq n) \] (def. of TC1)
\[ = (P \land Q \land (\exists n \in \mathbb{N} \bullet length(tr' - tr) \leq n)) \triangleleft \delta \triangleright \] (distr. of \& over if)
\[ ((P \lor Q) \land (\exists n \in \mathbb{N} \bullet length(tr' - tr) \leq n)) \] (prop. calculus)
\[ = (P \land (\exists n \in \mathbb{N} \bullet length(tr' - tr) \leq n) \land \]
\[ Q \land (\exists n \in \mathbb{N} \bullet length(tr' - tr) \leq n)) \triangleleft \delta \triangleright \]
\[ ((P \lor Q) \land (\exists n \in \mathbb{N} \bullet length(tr' - tr) \leq n)) \] (def. of +)
\[ (P \land (\exists n \in \mathbb{N} \bullet length(tr' - tr) \leq n)) + (Q \land (\exists n \in \mathbb{N} \bullet length(tr' - tr) \leq n)) \] (def. of TC1)

\[ TC1(P) + TC1(Q) \] (assumption)

\[ P + Q \]

\[ \square \]

Lemma 130 (closure-++TC2)

\[ TC2(P + Q) = P + Q \] provided \( P \) and \( Q \) are \( TC2 \) healthy

Proof.

\[ TC2(P + Q) = \]
\[ = TC2((P \land Q) \triangleleft \delta \triangleright (P \lor Q)) \] (def. of +)
\[ = (P \land Q) \triangleleft \delta \triangleright (P \lor Q) \land (\neg \text{wait} \implies (\text{wait}' \implies (ref'_in = A_{in} \lor ref_{in}' = \emptyset)))) \] (distr. of \& over if)
\[ = (P \land Q \land (\neg \text{wait} \implies (\text{wait}' \implies (ref'_in = A_{in} \lor ref_{in}' = \emptyset)))) \triangleleft \delta \triangleright \]
\[ ((P \lor Q) \land (\neg \text{wait} \implies (\text{wait}' \implies (ref'_in = A_{in} \lor ref_{in}' = \emptyset)))) \] (prop. calculus)
\[ = (P \land (\neg \text{wait} \implies (\text{wait}' \implies (ref'_in = A_{in} \lor ref_{in}' = \emptyset)))) \land \]
\[ Q \land (\neg \text{wait} \implies (\text{wait}' \implies (ref'_in = A_{in} \lor ref_{in}' = \emptyset)))) \triangleleft \delta \triangleright \]
\[ ((P \land (\neg \text{wait} \implies (\text{wait}' \implies (ref'_in = A_{in} \lor ref_{in}' = \emptyset)))) \lor \]
\[ (Q \land (\neg \text{wait} \implies (\text{wait}' \implies (ref'_in = A_{in} \lor ref_{in}' = \emptyset)))) \] (def. of +)
\[ (P \land (\neg \text{wait} \implies (\text{wait}' \implies (ref'_in = A_{in} \lor ref_{in}' = \emptyset)))) + \]
\[ (Q \land (\neg \text{wait} \implies (\text{wait}' \implies (ref'_in = A_{in} \lor ref_{in}' = \emptyset)))) \] (def. of TC2)

\[ TC2(P) + TC2(Q) \] (assumption)

\[ P + Q \]

\[ \square \]

Lemma 131 (closure-++TC3)

\[ TC3(P + Q) = P + Q \] provided \( P \) and \( Q \) are \( TC3 \) healthy
Lemma 132 (closure-+TC4)

\( TC4(P + Q) = P + Q \) provided \( P \) and \( Q \) are \( TC4 \) healthy

Proof.

\[
\begin{align*}
TC3(P + Q) &= \quad \text{(def. of +)} \\
&= TC3((P \land Q) < \delta \triangleright (P \lor Q)) \quad \text{(def. of TC3)} \\
&= (P \land Q) < \delta \triangleright (P \lor Q) \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (\ref{f}_{\text{out}}' \geq |\mathcal{A}_{\text{out}}| - 1))) \quad \text{(distr. of \land over if)} \\
&= (P \land Q \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (\ref{f}_{\text{out}}' \geq |\mathcal{A}_{\text{out}}| - 1)))) < \delta \triangleright \\
&\quad (P \lor Q) \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (\ref{f}_{\text{out}}' \geq |\mathcal{A}_{\text{out}}| - 1)))) < \delta \triangleright \\
&\quad (P \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (\ref{f}_{\text{out}}' \geq |\mathcal{A}_{\text{out}}| - 1)))) \lor \\
&\quad (Q \land (\neg \text{wait} \Rightarrow (\text{wait'} \Rightarrow (\ref{f}_{\text{out}}' \geq |\mathcal{A}_{\text{out}}| - 1)))) \quad \text{(def. of +)} \\
&\quad TC3(P) + TC3(Q) \quad \text{(assumption)} \\
&= P + Q
\end{align*}
\]

Lemma 133 (closure-+TC5)

\( TC5(P + Q) = P + Q \) provided \( P \) and \( Q \) are \( TC5 \) healthy
Proof.

$$\text{TC5}(P + Q) =$$

$$= \text{TC5}((P \land Q) \land (P \lor Q))$$

$$= (P \land Q) \land ((-\text{wait} \Rightarrow (\text{wait}' \Rightarrow ((\text{ref}'_i = \emptyset) \Rightarrow \text{ref}'_o = \text{A}_o)))$$

$$= (P \land Q) \land (\text{ref}'_i (\emptyset) \Rightarrow \text{ref}'_o = \text{A}_o))) < \delta >$$

$$= (P \land Q) \land (\text{ref}'_i (\emptyset) \Rightarrow \text{ref}'_o = \text{A}_o))) < \delta >$$

$$= (P \land Q) \land (\text{ref}'_i (\emptyset) \Rightarrow \text{ref}'_o = \text{A}_o))) < \delta >$$

$$= \text{prop. calculus}$$

$$\text{TC5}(P + \text{TC5}(Q))$$

$$= \text{assumption}$$

$$P + Q$$

Lemma 134 (closure+-TC5)

$$\text{TC6}(P + Q) = P + Q$$ provided $$P$$ and $$Q$$ are TC6 healthy

Proof.

$$\text{TC6}(P + Q) =$$

$$= \text{TC6}((P \land Q) \land (P \lor Q))$$

$$= (P \land Q) \land ((-\text{wait} \Rightarrow (\text{wait}' \Rightarrow (\text{pass}' \Rightarrow \text{ref}')))$$

$$= (P \land Q) \land (\text{pass}' (\text{ref}'_i) \Rightarrow \text{ref}'_o \\text{TC6}(Q))$$

$$= \text{prop. calculus}$$

$$\text{TC6}((P \land Q) < \delta \triangleright (P \lor Q))$$

$$= (P \land Q) \land (\text{ref}'_i (\emptyset) \Rightarrow \text{ref}'_o = \text{A}_o))) < \delta >$$

$$= (P \land Q) \land (\text{ref}'_i (\emptyset) \Rightarrow \text{ref}'_o = \text{A}_o))) < \delta >$$

$$= (P \land Q) \land (\text{ref}'_i (\emptyset) \Rightarrow \text{ref}'_o = \text{A}_o))) < \delta >$$

$$= \text{prop. calculus}$$

$$\text{TC6}(P) + \text{TC6}(Q)$$

$$= \text{assumption}$$

$$P + Q$$

Example 7 For example, a valid test case $$T$$ with the alphabet $$A_{in} = \{c, t, \theta\}$$ and $$A_{out} = \{1\}$$ that sends a stimulus and subsequently accepts a c-response but neither accepts $$t$$ nor $$\theta$$ is given by

$$T = \langle 1; ((?c \land \lor) + (?t \land x) + (?t \land x) \rangle = \{\text{def. of do_A, } \lor, \land, \text{ and } +\}$$

$$= \langle 1 ! \not \in r i \land t' = t \lor v \rangle \langle ! \not \in r i \land t' = t \lor v \rangle \langle 1, ?c \rangle \land \text{pass}' \lor v \rangle$$

$$\langle 1, ?t \rangle \land \text{fail}' v \rangle \langle t' = t \lor v \rangle \langle 1, ?c \rangle \land \text{pass}' \lor v \rangle$$

A test suite is a set of test cases. Because of the use of global verdicts (see Definition 38), a test suite is given by the nondeterministic choice of a set of test cases.

Definition 35 (Test suite) Given a set of $$N$$ test processes $$T_1, \ldots, T_N$$, then a test suite $$TS$$ is defined as:

$$TS = \bigcap_{i=1,...,N} T_i$$
3.6 Testing Implementations

Test case execution in the input output conformance testing framework is modeled by executing the test case in parallel to the system under test. We model this parallel execution again as parallel merge (see Section 3.2).

The execution of a test case $t$ on an implementation $i$ is denoted by $t||i$. This new process $t||i$ consists of all traces present in both, the test case and the implementation. Furthermore, $t||i$ gives $fail'$ and $pass'$ verdicts after termination.

Such an execution operator is inspired by CSPs parallel composition [Hoa85], i.e, the parallel composition of a test case and an implementation can only engage in a particular action if both processes participate in the communication.

Since a test case swaps inputs and outputs of the IUT we need to rename alphabets. Therefore, we define an alphabet renaming operator for a process $P$ denoted by $\tilde{P}$ as follows: $\tilde{AP} = _{\theta} AP; \tilde{A_{out}P} = _{\theta} A_{in}P; \tilde{A_{in}P} = _{\theta} A_{out}P$.

Definition 36 (Test case execution) Let $TC$ be a test case process and $IUT$ be an implementation process, then

$$A(TC||IUT) = _{\theta} ATC \cup AIUT$$

$$TC||IUT = _{\theta} (TC \triangleleft IUT); M_{ti}$$

The relation $M_{ti}$ merges the traces of the test case and the implementation. The result comprises the pass and fail verdicts of the test case as well as traces that are allowed in both, the test case and the implementation. Because of our representation of quiescence, there is no $\theta$ that indicates termination of the IUT, i.e., $\neg1.wait$. $M_{ti}$ takes care of that when merging the traces.

Definition 37 (Test case/impl. merge) $M_{ti} = _{\theta} pass' = 0.pass \land fail' = 0.fail \land wait' = (0.wait \land 1.wait) \land$

$ref' = (0.ref \lor 1.ref) \land ok' = (0.ok \land 1.ok) \land$

$(\exists u \bullet ((u = (0.tr - tr) \land u = (1.tr - tr) \land tr' = tr^\sim u) \lor$

$(u \bullet A_{in}^{\sim}(\theta) = (0.tr - tr) \land u = (1.tr - tr \land tr' = tr^\sim (u,\theta)) \land \neg1.wait))$

Due to the lack of symmetry of our merge operator the test case execution operator $||$ is not symmetric. However, it still satisfies one important law. Let $T_1, T_2$ be test cases and let $P$ be an implementation, then

LAW L2 $$(T_1 \cap T_2)||P = (T_1||P) \cap (T_2||P)$$

Proof.

$$(T_1 \cap T_2)||P =$$

$=$

$((T_1 \cap T_2) \triangleleft P); M_{ti}$

$($definition of $||)$

$=$

$((T_1 \cap T_2) \triangleleft P); M_{ti}$

($definition of renaming operator$)

$=$

$((T_1 \cap T_2) \triangleleft P); M_{ti}$

($commutativity of \cap over \triangleleft$)

$=$

$((T_1 \cap T_2) \triangleleft P); M_{ti}$

($commutativity of \cap over \cap$)

$=$

$((T_1 \cap T_2) \triangleleft P); M_{ti}$

($definition of ||)$

This law allows one to run a set of $N$ test cases $T_1, \ldots, T_N$, i.e. a test suite $TS = _{\theta} \cap_{i=1,...,N} T_i$, against an implementation process $P$: $TS||P = \cap_{i=1,...,N} T_i||P = (T_1 \cap \ldots \cap T_N)||P$. 


Since our test cases do not consist of a single trace but of several traces there may be different verdicts given at the end of different traces. An implementation passes a test case if all possible test runs lead to the verdict pass:

**Definition 38 (Global verdict)** Given a test process (or a test suite) \( T \) and an implementation process \( IUT \), then

\[
\begin{align*}
IUT \text{ passes } T & =_d \forall r \in A^\theta_\mathcal{A} \bullet (((T||IUT)[r/tr'] \land \neg\text{wait'}) \Rightarrow \text{pass'}) \\
IUT \text{ fails } T & =_d \exists r \in A^\theta_\mathcal{A} \bullet (((T||IUT)[r/tr'] \land \neg\text{wait'}) \Rightarrow \text{fail'})
\end{align*}
\]

**Example 8** Now we can calculate verdicts by executing test cases on implementations. For example, consider the test case of Example 7, i.e. \( T = !1; ((?c \land \mathcal{V}) + (?t \land \mathcal{X}) + (\emptyset \land \mathcal{X})) \), and the IUT \( P_w = i^\theta_\mathcal{A}(?1); i^\theta_\mathcal{A}(lc) \) (representing the IUT \( w \) of Figure 3). Executing \( T \) on the IUT \( P_w \), i.e. \( T||P_w \), is conducted as follows:

\[
\begin{align*}
T||P_w & = \text{def. of } | | \text{ and renaming} \\
& = (?1; ((!c \land \mathcal{V}) + (?t \land \mathcal{X}) + (\emptyset \land \mathcal{X})) \triangleleft i^\theta_\mathcal{A}(?1); i^\theta_\mathcal{A}(lc); M_{i1} \quad \{ \text{def. of } \triangleleft \text{ and } M_{i1} \} \\
& = \left( \begin{array}{l}
?1 \notin ref' \land \text{tr'} = \text{tr} \lor \\
\text{tr} \notin ref' \land \text{tr'} = \text{tr} \triangleright (?1) \end{array} \right) < \text{wait'} \triangleright (\text{tr'} = \text{tr} \triangleright (?1, ?c) \land \text{pass'})
\end{align*}
\]

Thus, we have \( P_w \) passes \( T \) because

\[
\begin{align*}
P_w \text{ passes } T & = \forall r \in A^\theta_\mathcal{A} \bullet (((T||P_w)[r/tr'] \land \neg\text{wait'}) \Rightarrow \text{pass'}) \\
& = \forall r \in A^\theta_\mathcal{A} \bullet (((\text{prop. calc.}) \\
& = \forall r \in A^\theta_\mathcal{A} \bullet \text{TRUE} = \text{TRUE} \quad \Box
\end{align*}
\]

### 4 Conclusion and Future Work

This paper lifts the input output conformance (\( \text{ioco} \)) theory of Tretmans [Tre96] for functional black-box testing of reactive systems to UTP’s reactive processes [HH98].

The presented operators make the absence of output events, i.e. quiescence, observable for reactive processes. Furthermore, we show how to express input enabled processes and introduced a formal notion of fairness. By the use of specification processes and implementation processes we define \( \sqsubseteq_{\text{ioco}} \). This conformance relation gives a notion of correctness of an implementation with respect to a specification in terms of UTP’s reactive processes.

Although, the presented theory is more complex than Tretmans’ original formulation there are many benefits in embedding \( \text{ioco} \) in UTP. First, the presented healthiness conditions are mostly simple and formalize the assumptions behind \( \text{ioco} \). To the best of our knowledge, this is the first time that these assumptions have been presented in a formal way. Second, we can formally prove properties of \( \sqsubseteq_{\text{ioco}} \), and check proofs automatically. For example, particular steps of some proofs of [Wei08] have been checked using satisfiability solvers (e.g. [DdM08]). Thanks to the predicative style of \( \sqsubseteq_{\text{ioco}} \) such decision procedures cannot only be used for proof checking, but also for test case generation by expressing test case generation as a satisfiability problem. Finally, formulating \( \text{ioco} \) in terms of UTP make specifications with a UTP semantics useable for \( \text{ioco} \) testing.

Although, this paper gives the basic notion of specifications, implementations, test cases and conformance there is plenty of work left. While we related refinement and \( \sqsubseteq_{\text{ioco}} \), there are many other interesting laws that should be investigated. Another open task is to instantiate this framework for a particular process algebra, e.g., CSP. Furthermore, there are many extensions to the original input output conformance. For example, Lestiennes and Gaudel [LG05] presented \( \text{rioco} \), which relaxes the property of input-enabledness. Another variation of \( \text{ioco} \) is conformance under the presence of time, i.e. \( \text{tioco} \) [KT04]. It would be interesting to study these conformance relations in terms of UTP and compare arising healthiness conditions to the healthiness conditions presented in this paper.
Acknowledgments  The research herein is conducted within the competence network Softnet Austria and funded by the Austrian Federal Ministry of Economics (bm:wa), the province of Styria, the Steirische Wirtschaftsförderungsgesellschaft mbH. (SFG), and the city of Vienna in terms of the center for innovation and technology (ZIT). This research is partially funded by the EU FP7 project MOGENTES ICT-216679.

References


