Relating Semantic Models of Compensating CSP

Shamim Ripon and Michael Butler

School of Electronics and Computer Science
University of Southampton.
{sr03r,mjb}@ecs.soton.ac.uk

Abstract. Building equivalences between different semantic models of a language strengthens the formal foundation of the language. This paper shows the derivation of denotational semantics from operational semantics of the language cCSP. The aim is to show the correspondence between the operational and trace semantics. We extract traces from operational rules and use structural induction to show the correspondence between the two semantics of cCSP.

1 Introduction

Operational and denotational semantics are two well-known methods of assigning meaning to programming languages and both semantics are useful for a full description of the language. Denotational semantics associates an element of a semantic domain to each expression in the language and the semantics is compositional. Traces are one of the ways to define denotational semantics. A trace gives the global picture of the behaviour of a system. The common way of defining operational semantics is to provide state transition systems for the language, where the transition system models the computation steps of expressions in the language and allows the formal analysis of the language. Potential use for operational semantics is for model checking.

Inspired by using process algebra, especially CSP [5], in transaction processing, compensating CSP (cCSP) was introduced and its trace semantics were defined in [2]. Following the introduction of trace semantics, the operational semantics of cCSP are defined by using labelled transition system [3]. Having defined both denotational and operational semantics, it is natural to see how these two semantics are related to each other.

This paper draws the correspondence of two different semantic representations of a language. In particular, the aim is to accomplish the unification between operational and denotational approach of cCSP. The unification is based on the approach where we use the transition rules from operational semantics to derive the traces and then show that these derived traces correspond to the original trace definitions of the operators by using induction over the definitions. Proving the correspondence means that any of the presentations can be accepted as a primary definition of the meaning of the language and each of the definitions can be correctly used at different times and for different purposes. cCSP is comprised of standard and compensable processes and for both of the processes we derive traces by using the operational rules and show that the operational semantics correspond to the denotational semantics.

The remainder of the paper is organised as follows. Section 2 recalls the traces and operational semantics of cCSP. Following this, Section 3 briefly outlines the approach taken to proving the correspondence and then describes the theorem and the supporting lemmas by which we can draw the correspondence of the two semantics. After briefly giving some related works, we then draw the conclusions and discuss our future works.

2 Background

The cCSP language was inspired by two main ideas: transaction processing features and process algebra, especially CSP. As in CSP, processes in cCSP are modelled in terms of the atomic events they can engage in and the operators provided by the language support sequencing, choice, parallel composition of processes. In order to support failed transactions, compensation operators are introduced. Processes are categorized into standard and compensable processes. Standard processes do not have any compensation but compensations are attached to compensable processes which are used to compensate the failure of transactions.

We use P, Q to identify standard processes and PP, QQ to identify compensable processes. This section summarises the basics of cCSP language with a brief description of the traces and operational semantics. The syntax of the language is summarised here.

Standard Processes:

$P, Q ::= A \mid P \; ; \; Q \mid P \| Q \mid [PP] \mid P \,\Box\, Q \mid P \rhd Q \mid \mathit{SKIP} \mid \mathit{THROW} \mid \mathit{YIELD}$

Compensable Processes:

$PP, QQ ::= P \div Q \mid PP ; QQ \mid PP \| QQ \mid PP \,\Box\, QQ \mid \mathit{SKIPP} \mid \mathit{THROWW} \mid \mathit{YIELDD}$

The basic unit of a standard process is the atomic event (A). Other operators of standard processes are the sequential ($P \; ; \; Q$) and parallel ($P \| Q$) composition of processes, the choice operator ($\Box$), interrupt handler ($\rhd$), the empty process *SKIP*, the raise of an interrupt *THROW* and the yield of an interrupt *YIELD*. In parallel processes, the whole group of parallel processes may fail when one throws an exception and all the other processes are willing to yield to that exception. A process that is ready to terminate is also willing to yield an interrupt. Yield points are inserted in a process through *YIELD*. In parallel composition, throwing an interrupt in one process synchronises with yielding in another process. The basic way of constructing a compensable process is through the compensation pair construct ($P \div Q$) where P is the forward behaviour and Q is its associated compensation which is designed to compensate the effect of P. Sequential composition of compensable processes is defined so that the compensations for the performed actions will be accumulated in reverse to their original performance. Parallel composition of compensable processes is defined so that compensations are accumulated in parallel. The current definition of parallel operator does not support synchronisation on normal events. By enclosing a compensable process PP in a transaction block $[PP]$ we get a complete transaction which converts the compensable process PP into a standard process. The behaviours of the block are defined in terms of the behaviour of PP. Successfully completed PP represents successful completion of the whole block and compensations are no longer required. When the forward behaviour of PP throws an interrupt, the compensations are executed in the appropriate order and the interrupt is not observable outside the block. $SKIPP$, $THROWW$, $YIELDD$ are the compensable counterpart of the standard primitive processes. Figure 1 presents an example of a transaction for processing customer order in a warehouse in cCSP language. The transaction is represented in a transaction block where the block consists of two processes composed sequentially. *RestockOrder* is the compensation of *AcceptOrder*. *FulfillOrder* consists of processes which is composed in parallel where the main tasks are booking a courier, packing the items and check credit in parallel. In case of failure, the semantics the block will ensure that the appropriate compensation will be invoked.

$$\begin{aligned}
\textbf{ProcessOrder} &= [\,(\mathit{AcceptOrder} \div \mathit{RestockOrder})\,;\ \textbf{FulfillOrder}\,] \\
\textbf{FulfillOrder} &= \mathit{BookCourier} \div \mathit{CancelCourier} \parallel \\
&\ \textbf{PackOrder} \parallel \\
&\ \mathit{CreditCheck}\,;\ (\ \mathit{Ok}\ ;\ \mathit{SKIPP}\ \Box\ \mathit{NotOk}\ ;\ \mathit{THROWW}\ ) \\
\textbf{PackOrder} &= \parallel i \in \mathit{Items}\ \bullet\ (\mathit{PackItem}(i) \div \mathit{UnpackItem}(i))
\end{aligned}$$

Fig. 1. Order transaction processing

2.1 Trace Semantics

A trace of a process records history of behaviour up to some point. The trace semantics of the cCSP language are summarised in Figure 2. We show the operators on traces which are then lifted to operators on set of traces. Traces considered for cCSP are non-empty sets.

The semantics of a standard process is a set of traces of the form $s\langle\omega\rangle$ where $s \in \Sigma^*$ (Σ is alphabet of normal events) and $\omega \in \Omega$ ($\Omega = \{\checkmark, !, ?\}$), which means all traces end with any of the events in Ω, which are called terminal events. The terminal events represent the termination of a process. Successful termination is shown by a $\checkmark$. Termination by either throwing or yielding an interrupt is shown by ! or ? respectively. The following healthiness condition

$$p\langle\checkmark\rangle \in P \text{ or } p\langle !\rangle \in P \text{ for some } p \in \Sigma^*$$

declares that all standard processes consists of some terminating or interrupting behaviour. Unlike standard CSP, the trace prefixes are not included in traces of cCSP. In sequential composition (p ; q), the traces are the concatenated observable traces of p and q, only when p terminates successfully,(ends with $\checkmark$), otherwise the trace is only p. The traces of two parallel processes are $p\langle\omega\rangle \| q\langle\omega'\rangle$

Standard Processes

Atomic Action

For $A \in \Sigma$, $\quad T(A) = \{\langle A, \checkmark\rangle\}$

Sequential Composition

$p\langle\checkmark\rangle \ ; \ q = p.q, \quad$ and $\quad p\langle\omega\rangle \ ; \ q = p\langle\omega\rangle$, where $\omega \neq \checkmark$

$T(P \ ; \ Q) = \{p \ ; \ q \mid p \in P \wedge q \in Q\}$

Parallel Composition

$p\langle\omega\rangle \| q\langle\omega'\rangle = \{r\langle\omega\&\omega'\rangle \mid r \in (p \ ||| \ q)\}$

$T(P\|Q) = \{r \mid r \in (p\|q) \wedge p \in P \wedge q \in Q\}$

where

ω	!	!	!	?	?	✓
ω'	!	?	✓	?	✓	✓
$\omega\&\omega'$	!	!	!	?	?	✓

Interrupt Handler

$p\langle !\rangle \ \rhd \ q \ = \ p.q$ and $p\langle\omega\rangle \ \rhd \ q \ = \ p\langle\omega\rangle$, **where** $\omega \neq !$

$P \ \rhd \ Q \ = \{p \ \rhd \ q \mid p \in P \ \wedge \ q \in Q\}$

Choice

$T(P \,\Box\, Q) \ = \ T(P) \cup T(Q)$

Transaction Block

$[p\langle !\rangle, p'] \ = \ p.p'$ and $[p\langle\checkmark\rangle, p'] = p\langle\checkmark\rangle$

$T([PP]) = \{[p, p'] \mid (p, p') \in PP \ \wedge \ last(p) \neq ?\}$

Basic Processes

$T(SKIP) \ = \ \{\langle\checkmark\rangle\}, \quad T(THROW) \ = \ \{\langle !\rangle\}, \quad T(YIELD) \ = \ \{\langle ?\rangle, \langle\checkmark\rangle\}$

Compensable Processes

Compensation Pair

$p\langle\checkmark\rangle \div q = (p\langle\checkmark\rangle, q)$ and $p\langle\omega\rangle \div q \ = \ (p\langle\omega\rangle, \langle\checkmark\rangle)$ where $\omega \neq \checkmark$

$T(P \div Q) = \{p \div q \mid p \in P \wedge q \in Q\}$

Sequential Composition

$(p\langle\checkmark\rangle, p') \ ; \ (q, q') = (p.q, q' \ ; \ p')$ and $(p\langle\omega\rangle, p') \ ; \ (q, q') = (p\langle\omega\rangle, p')$ where $\omega \neq \checkmark$

$T(PP \ ; \ QQ) = \{pp \ ; \ qq \mid pp \in PP \wedge qq \in QQ\}$

Parallel Composition

$(p, p')\|(q, q') = \{(r, r') \mid r \in (p\|q) \wedge r' \in (p'\|q')\}$

$T(PP\|QQ) = \{rr \mid rr \in (pp\|qq) \wedge pp \in PP \wedge qq \in QQ\}$

Compensable Choice

$T(PP \,\Box\, PQ) \ = \ T(PP) \cup T(QQ)$

Compensable Basic Processes

$T(SKIPP) = T(SKIP \div SKIP) \quad = \{(\langle ?\rangle, \langle\checkmark\rangle), (\langle\checkmark\rangle, \langle\checkmark\rangle)\}$

$T(THROWW) = T(THROW \div SKIP) = \{(\langle ?\rangle, \langle\checkmark\rangle), (\langle !\rangle, \langle\checkmark\rangle)\}$

$T(YIELDD) = T(YIELD \div SKIP) \quad = \{(\langle ?\rangle, \langle\checkmark\rangle)\}$

Fig. 2. Trace semantics of cCSP

which corresponds to the set $(p \ ||| \ q)$, the possible interleaving of traces of both processes and followed by $\omega\&\omega'$, the synchronisation of ω and ω'.

Compensable processes are comprised of forward and compensation behaviour and the traces of compensable processes consist of a pair of traces of the form $(s\langle\omega\rangle, s'\langle\omega'\rangle)$, where $s\langle\omega\rangle$ is the forward behaviour and $s'\langle\omega'\rangle$ is the compensation behaviour. In sequential composition, the forward traces correspond to the original forward behaviour and it is then followed by the traces of the compensation. Traces of parallel composition are defined as the interleaving of forward traces followed by the interleaving of compensations. Traces of the compensation pair are the traces of both of the processes of the pair when the forward process (P) terminate with a $\langle \checkmark \rangle$, otherwise it is traces of the forward process followed by only a $\langle \checkmark \rangle$. Traces of a transaction block are only the traces of the compensable process inside the block when the process terminates with a $\langle \checkmark \rangle$, otherwise when the forward process terminates with a $\langle ! \rangle$ the traces of the block are the traces of the forward process followed by the traces of the compensation. Similar to standard processes, the following healthiness condition

$$\begin{aligned} (p\langle \checkmark \rangle, p'\langle \omega \rangle) \in PP \quad \text{or} \quad & (p\langle ! \rangle, p'\langle \checkmark \rangle) \in PP \\ \text{or} \quad & (p\langle ! \rangle, p'\langle \checkmark \rangle) \in PP \quad \text{for some} \quad p, p' \end{aligned}$$

states that compensable processes consist of some terminating or interrupting behaviour which ensures that traces of processes are non-empty and this condition is preserved by all the operators.

2.2 Operational Semantics

By using labelled transition systems [10], the operational semantics specifies the relation between states of a program. We extend the terms of the language to define the operational semantics with 0 and $\langle PP, P \rangle$, where 0 represents the null process which cannot perform any event and $\langle PP, P \rangle$ is an auxiliary construct which is derived during defining the operational semantics of compensable sequential composition. These auxiliary terms have no corresponding definitions in the trace definition of the language. The compensation pair defined here has a subtle difference to that presented in the original trace definition [2], where an extra behaviour was included, which allows the operator to yield immediately with an empty compensation. The same behaviour can be obtained from the definition presented here by adding a *YIELD* sequentially followed by the forward behaviour of the pair as follows:

$$P \div' Q \;\hat{=}\; (\mathit{YIELD} \;;\; P) \div Q$$

YIELD can either yield (?) or terminate with a $\checkmark$. When it yields the above definition gives the required extra behaviour of yield with an empty compensation of the original trace model and when *YIELD* terminates with a $\checkmark$, the above definition gives the same behaviour presented in this paper.

Each process has two different kind of transition: by normal events and by terminal events. Normal events make the transition of a process from one state to another state. For example, the normal event a makes the transition of a standard process from P to P' and a compensable process from PP to PP'.

$$\begin{aligned} P \xrightarrow{a} P' \quad & (P' \text{ is a standard process}) \\ PP \xrightarrow{a} PP' \quad & (PP' \text{ is a compensable process}) \end{aligned}$$

A terminal event (ω) causes the standard processes to terminate with a null (0) process. But the effect of a terminal event is different in a compensable process, where the compensable process terminates and the attached compensation,

Standard Processes

Atomic Action: $A \xrightarrow{A} SKIP \quad (A \in \Sigma)$

Basic Processes: $SKIP \xrightarrow{\checkmark} 0,\ THROW \xrightarrow{!} 0,\ YILED \xrightarrow{\checkmark} 0,\ YIELD \xrightarrow{?} 0$

Sequential Composition: $S1 : \dfrac{P \xrightarrow{a} P'}{P\ ;\ Q \xrightarrow{a} P'\ ;\ Q} \quad (a \in \Sigma) \qquad S2 : \dfrac{P \xrightarrow{\omega} 0}{P\ ;\ Q \xrightarrow{\omega} 0} \quad (\omega \neq \checkmark)$

$$S3 : \frac{P \xrightarrow{\checkmark} 0 \wedge Q \xrightarrow{a} Q'}{P\ ;\ Q \xrightarrow{a} Q'} \quad (a \in \Sigma \cup \Omega)$$

Parallel Composition: $P1 : \dfrac{P \xrightarrow{a} P'}{P \| Q \xrightarrow{a} P' \| Q} \qquad P2 : \dfrac{Q \xrightarrow{a} Q'}{P \| Q \xrightarrow{a} P \| Q'} \quad (a \in \Sigma)$

$$P3 : \frac{P \xrightarrow{\omega} 0\ \wedge\ Q \xrightarrow{\omega'} 0}{P \| Q \xrightarrow{\omega \& \omega'} 0} \quad \text{where} \begin{array}{c|cccccc} \omega & ! & ! & ! & ? & ? & \checkmark \\ \omega' & ! & ? & \checkmark & ? & \checkmark & \checkmark \\ \hline \omega \& \omega' & ! & ! & ! & ? & ? & \checkmark \end{array}$$

Choice: $C1 : \dfrac{P \xrightarrow{a} P'}{P \Box Q \xrightarrow{a} P'} \qquad C2 : \dfrac{Q \xrightarrow{a} Q'}{P \Box Q \xrightarrow{a} Q'} \quad (a \in \Sigma \cup \Omega)$

Interrupt Handler: $IH1 : \dfrac{P \xrightarrow{a} P'}{P \rhd Q \xrightarrow{a} P' \rhd Q} \quad (a \in \Sigma) \quad IH2 : \dfrac{P \xrightarrow{\omega} 0}{P \rhd Q \xrightarrow{\omega} 0} \quad (\omega \in \Sigma \wedge \omega \neq !)$

$$IH3 : \frac{P \xrightarrow{!} 0\ \wedge\ Q \xrightarrow{a} Q'}{P \rhd Q \xrightarrow{a} Q'} \quad (a \in \Sigma \cup \Omega)$$

Transaction Block: $T1 : \dfrac{PP \xrightarrow{a} PP'}{[PP] \xrightarrow{a} [PP']} \quad (a \in \Sigma) \quad T2 : \dfrac{PP \xrightarrow{\checkmark} P}{[PP] \xrightarrow{\checkmark} 0}$

$$T3 : \frac{PP \xrightarrow{!} P \wedge P \xrightarrow{a} P'}{[PP] \xrightarrow{a} P'} \quad (a \in \Sigma \cup \Omega)$$

Compensable Processes

Compensation Pair: $R1 : \dfrac{P \xrightarrow{a} P'}{P \div Q \xrightarrow{a} P' \div Q} \ (a \in \Sigma) \quad R2 : \dfrac{P \xrightarrow{\checkmark} 0}{P \div Q \xrightarrow{\checkmark} Q}$

$$R3 : \frac{P \xrightarrow{\omega} 0}{P \div Q \xrightarrow{\omega} SKIP} \quad (\omega \neq \checkmark)$$

Sequential Composition: $CS1 : \dfrac{PP \xrightarrow{a} PP'}{PP\ ;\ QQ \xrightarrow{a} PP'\ ;\ QQ} \ (a \in \Sigma) \quad CS2 : \dfrac{PP \xrightarrow{\omega} P}{PP\ ;\ QQ \xrightarrow{\omega} P} \ (\omega \neq \checkmark)$

$$CS3 : \frac{PP \xrightarrow{\checkmark} P \wedge QQ \xrightarrow{\omega} Q}{PP\ ;\ QQ \xrightarrow{\omega} Q\ ;\ P} \ (\omega \in \Omega) \quad CS4 : \frac{PP \xrightarrow{\checkmark} P \wedge QQ \xrightarrow{a} QQ'}{PP\ ;\ QQ \xrightarrow{a} \langle QQ', P \rangle} \ (a \in \Sigma)$$

$$CS5 : \frac{QQ \xrightarrow{a} QQ'}{\langle QQ, P \rangle \xrightarrow{a} \langle QQ', P \rangle} \ (a \in \Sigma) \qquad CS6 : \frac{QQ \xrightarrow{\omega} Q}{\langle QQ, P \rangle \xrightarrow{\omega} Q\ ;\ P} \ (\omega \in \Omega)$$

Parallel Composition: $CP1 : \dfrac{PP \xrightarrow{a} PP'}{PP \| QQ \xrightarrow{a} PP' \| QQ} \quad CP2 : \dfrac{QQ \xrightarrow{a} QQ'}{PP \| QQ \xrightarrow{a} PP \| QQ'} \quad (a \in \Sigma)$

$$CP3 : \frac{PP \xrightarrow{\omega} P \wedge QQ \xrightarrow{\omega'} Q}{PP \| QQ \xrightarrow{\omega \& \omega'} P \| Q}$$

Compensable Choice: $CC1 : \dfrac{PP \xrightarrow{a} PP'}{PP \Box\, QQ \xrightarrow{a} PP'} \quad CC2 : \dfrac{QQ \xrightarrow{a} QQ'}{PP \Box\, QQ \xrightarrow{a} QQ'} \ (a \in \Sigma)$

$$CC3 : \frac{PP \xrightarrow{\omega} P}{PP \Box\, QQ \xrightarrow{\omega} P} \qquad CC4 : \frac{QQ \xrightarrow{\omega} Q}{PP \Box\, QQ \xrightarrow{\omega} Q} \ (\omega \in \Omega)$$

Fig. 3. Operational semantics of cCSP

which is a standard process, is stored for future use.

$$\begin{aligned} P &\xrightarrow{\omega} 0 \\ PP &\xrightarrow{\omega} P \qquad (P \text{ is the compensation}) \end{aligned}$$

The operational semantics of cCSP are summarised in Figure 3 . Considering standard processes, in sequential composition ($P \,;\, Q$), the second process Q in the sequence can start only when the first process P terminates successfully (with $\checkmark$), otherwise the first process will terminate with ! or ? and the second process will not start. In parallel composition each process can evolve independently and processes synchronise only on terminal events.

Compensable process has a forward behaviour and a compensation of the forward behaviour which will store after the completion of the forward behaviour for future use. In compensable sequential composition ($PP \,;\, QQ$), while the first process (PP) terminates, its compensation (P) will be stored and the second process (QQ) will start. In this scenario we get an auxiliary construct ($\langle QQ, P\rangle$) and it will be described later in the proof. After termination of the second process (QQ) its compensation (Q) will accumulate in front of P, i.e., ($Q \,;\, P$). In parallel composition the basic difference with standard processes is that after termination, compensations are accumulated and in parallel. In compensation pair, after successful completion of the forward behaviour the compensation will be stored for future use, however, unsuccessful termination, i.e, termination by ! or ? results an empty compensation. Transaction block converts a compensable process into a standard process. A non-terminal event changes the state of the process inside the block. Successful completion of the forward process inside the block means completion of the whole block, but throwing an interrupt by the compensable process inside the block causes the compensation to run.

3 Correspondence

This section shows the correspondence between the operational and trace semantics. The correspondence is shown in two steps. In the first step, traces are derived from operational rules and in the next step, we show the correspondence between the derived traces and the original definitions of the trace semantics.

Operational semantics defines the lifted transition relation labelled by sequence of events. The derived traces of a standard process P is defined as $DT(P)$, if we let $t \in DT(P)$ then we get the following definition:

$$t \in DT(P) \;=\; P \xrightarrow{t} 0$$

where t is the derived trace which consists of a sequence of events followed by a terminal event $\omega \in \{\checkmark, !, ?\}$. By applying induction over the traces, where $\langle\omega\rangle$ is considered as the basic step and $\langle a\rangle t$ is considered as the inductive step, we show that

$$\begin{aligned} P \xrightarrow{\langle\omega\rangle} 0 &\;=\; P \xrightarrow{\omega} 0 \\ P \xrightarrow{\langle a\rangle t} 0 &\;=\; \exists\, P' \cdot P \xrightarrow{a} P' \;\wedge\; P' \xrightarrow{t} 0 \end{aligned}$$

Compensable processes are modelled by using pair of traces: one for forward behaviour and another for compensation. The following definition represents a completed behaviour of the forward process:

$$PP \xrightarrow{t} R$$

where t is the trace of the forward behaviour and R is the compensation. When the behaviour of the compensation is added we get the following definition:

$$PP \xrightarrow{(t,t')} 0 \;=\; \exists\, R \cdot PP \xrightarrow{t} R \;\wedge\; R \xrightarrow{t'} 0$$

where t' is the trace of the compensation. By using these two definitions we get the trace derivation rule of a compensable process PP defined as:

$$(t,t') \in DT(PP) \;=\; PP \xrightarrow{(t,t')} 0$$

Induction over traces are applied here similar to standard processes. After deriving the traces, we then show the correspondence. The traces of standard and compensable processes are presented as $T(P)$ and $T(PP)$ respectively. Recall that $T(P)$ and $T(PP)$ are defined directly for the constructs of cCSP (Fig 2) while $DT(P)$ and $DT(PP)$ are defined indirectly via operational rules. By structural induction over the derived traces, we show that $DT(P) = T(P)$ and $DT(PP) = T(PP)$. The paper shows the proof of the following theorem:

Theorem 1.
$DT(P) = T(P)$ *for all standard processes* P, *not containing* 0
$DT(PP) = T(PP)$ *for all compensable processes* PP, *not containing* $\langle PP, P\rangle$ *or* 0

The theorem is proved by structural induction over terms of the language. In the following sections, we show the correspondence of the two semantics of the cCSP operators for both standard and compensable processes.

There is another theorem that is required during the proofs to show the correspondence.

Theorem 2. *For any process term* P *(not containing* 0*)*
$\forall\, P \cdot \exists\, t \cdot P \xrightarrow{t} 0$

This theorem can be proved by induction over process terms.

3.1 Sequential Composition

This section demonstrates the correspondence of sequential composition. The correspondence is drawn separately for standard and compensable processes.

Standard Process
The correspondence between derived traces and the original defined traces are drawn by showing that $DT(P\ ;\ Q) = T(P\ ;\ Q)$ and to prove it consider that $DT(P) = T(P)$ and $DT(Q) = T(Q)$. We let $t \in DT(P\ ;\ Q)$ and following the trace derivation rule we get

$$t \in DT(P\ ;\ Q) \ = \ (P\ ;\ Q) \xrightarrow{t} 0$$

On the other hand, let $t \in T(P\ ;\ Q)$ and from the definition of trace semantics

$$\begin{aligned} & t \in T(P\ ;\ Q) \\ = \quad & \exists\, p, q \cdot t = (p\ ;\ q) \ \wedge \ p \in T(P) \ \wedge \ q \in T(Q) \quad \text{[trace definition]} \\ = \quad & \exists\, p, q \cdot t = (p\ ;\ q) \ \wedge \ p \in DT(P) \ \wedge \ q \in DT(Q) \quad \text{[induction assumption]} \\ = \quad & \exists\, p, q \cdot t = (p\ ;\ q) \ \wedge \ P \xrightarrow{p} 0 \ \wedge \ Q \xrightarrow{q} 0 \quad \text{[trace derivation rule]} \end{aligned}$$

This leads to the following lemma:

Lemma 3
$(P\ ;\ Q) \xrightarrow{t} 0 \quad = \quad \exists\, p, q \cdot \ t = (p\ ;\ q) \ \wedge \ P \xrightarrow{p} 0 \ \wedge \ Q \xrightarrow{q} 0$

The lemma is proved by induction over the traces. The induction is based on the following two cases:

- case $\langle \omega \rangle$ - the lemma is proved for trace $\langle \omega \rangle$ (basic step).
- case $\langle a \rangle t$ - the lemma is proved for the trace $\langle a \rangle t$ considering that the lemma holds for trace t (inductive step).

The following equations are derived from operational rules which will support the proof of the above lemma.

- $(P\ ;\ Q) \xrightarrow{\omega} 0 \quad = \quad P \xrightarrow{\checkmark} 0 \wedge Q \xrightarrow{\omega} 0 \ \ \vee \ \ P \xrightarrow{\omega} 0 \wedge \omega \neq \checkmark$
- $(P\ ;\ Q) \xrightarrow{a} R \quad = \quad (\exists\, P' \cdot P \xrightarrow{a} P' \wedge R = (P'\ ;Q)) \ \ \vee \ \ P \xrightarrow{\checkmark} 0 \ \wedge \ Q \xrightarrow{a} R$

The basic step of the inductive proof of the lemma is trivial. The inductive step is given here:
Case - $\langle a \rangle t$:
$(P\ ;Q) \xrightarrow{\langle a \rangle t} 0 \ \ = \ \exists\, R \cdot (P\ ;\ Q) \xrightarrow{a} R \ \wedge \ R \xrightarrow{t} 0$
"From operational rules $(S1)$ and $(S3)$"

$$\begin{aligned} = \quad & \exists\, P' \cdot P \xrightarrow{a} P' \ \wedge \ (P'\ ;\ Q) \xrightarrow{t} 0 & (1) \\ & \vee\, P \xrightarrow{\checkmark} 0 \ \wedge \ Q \xrightarrow{a} R \ \wedge \ R \xrightarrow{t} 0 & (2) \end{aligned}$$

From (1)

$$
\begin{aligned}
& \exists P' \cdot P \xrightarrow{a} P' \ \wedge\ (P' \ ; \ Q) \xrightarrow{t} 0 \\
= \ & \text{“inductive hypothesis”} \\
& \exists P' \cdot P \xrightarrow{a} P' \ \wedge\ \exists p', q \cdot\ t = (p' \ ; \ q) \ \wedge\ P' \xrightarrow{p'} 0 \ \wedge\ Q \xrightarrow{q} 0 \\
= \ & \text{“combining existential quantifications”} \\
& \exists p', q \cdot\ t = (p' \ ; \ q) \ \wedge\ P \xrightarrow{\langle a \rangle p'} 0 \ \wedge\ Q \xrightarrow{q} 0 \\
= \ & \text{“using trace rule } \langle a \rangle t = \langle a \rangle (p' \ ; \ q) = (\langle a \rangle p') \ ; \ q\text{”} \\
& \exists p', q \cdot \langle a \rangle t = (\langle a \rangle p' \ ; \ q) \ \wedge\ P \xrightarrow{\langle a \rangle p'} 0 \ \wedge\ Q \xrightarrow{q} 0 \\
= \ & \exists p, q \cdot p = \langle a \rangle p' \ \wedge\ \langle a \rangle t = (p \ ; \ q) \ \wedge\ P \xrightarrow{p} 0 \ \wedge\ Q \xrightarrow{q} 0
\end{aligned}
$$

From (2)

$$
\begin{aligned}
& P \xrightarrow{\checkmark} 0 \ \wedge\ Q \xrightarrow{a} R \ \wedge\ R \xrightarrow{t} 0 \\
= \ & P \xrightarrow{\checkmark} 0 \ \wedge\ Q \xrightarrow{\langle a \rangle t} 0 \\
= \ & \exists p, q \cdot\ p = \langle \checkmark \rangle \ \wedge\ q = \langle a \rangle t \ \wedge\ P \xrightarrow{p} 0 \ \wedge\ Q \xrightarrow{q} 0 \\
= \ & \text{“}\langle \checkmark \rangle \ ; \ q = q\text{”} \\
& \exists p, q \cdot \langle a \rangle t = (p \ ; \ q) \ \wedge\ p = \langle \checkmark \rangle \ \wedge\ P \xrightarrow{p} 0 \ \wedge\ Q \xrightarrow{q} 0
\end{aligned}
$$

Therefore, for $\langle a \rangle t$, from (1) $\vee$ (2)

$$
\begin{aligned}
& \quad \exists p, q \cdot p = \langle a \rangle p' \ \wedge\ \langle a \rangle t = (p \ ; \ q) \ \wedge\ P \xrightarrow{p} 0 \ \wedge\ Q \xrightarrow{q} 0 \\
& \vee \exists p, q \cdot\ p = \langle \checkmark \rangle \ \wedge\ \langle a \rangle t = (p \ ; \ q) \ \wedge\ P \xrightarrow{p} 0 \ \wedge\ Q \xrightarrow{q} 0 \\
= \ & \text{“combining existential quantifications”} \\
& \exists p, q \cdot\ (p = \langle \checkmark \rangle \ \vee\ p = \langle a \rangle p') \ \wedge\ \langle a \rangle t = (p \ ; \ q) \ \wedge\ P \xrightarrow{p} 0 \ \wedge\ Q \xrightarrow{q} 0 \\
= \ & \text{“}p \ ; \ q = \langle a \rangle t \ \Rightarrow\ p \neq \langle ! \rangle \ \wedge\ p \neq \langle ? \rangle\text{”} \\
& \exists p, q \cdot \langle a \rangle t = (p \ ; \ q) \ \wedge\ P \xrightarrow{p} 0 \ \wedge\ Q \xrightarrow{q} 0
\end{aligned}
$$

This completes the proof of the lemma. We follow the same approach to prove all the other lemmas in the rest of the paper. The proof of the basic step of the induction of the above lemma and the proofs of all the other lemmas in the paper can be found in the appendix.

Compensable Process

Compensable processes have both forward and compensation behaviour. Compensable processes consist of a pair of traces of the form $(p\langle \omega \rangle, p'\langle \omega' \rangle)$, where $p\langle \omega \rangle$ represents the trace of the forward behaviour and $p'\langle \omega' \rangle$ represents the compensation. Let $(t, t') \in DT(PP \ ; \ QQ)$ and according to trace derivation rules we get

$$(t, t') \in DT(PP \ ; \ QQ) \ = \ \exists R \cdot (PP \ ; \ QQ) \xrightarrow{t} R \wedge R \xrightarrow{t'} 0$$

The following lemma allows us to derive the lifted forward traces from sequential composition of compensable processes.

Lemma 4

$$
\begin{aligned}
& (PP \ ; \ QQ) \xrightarrow{t} R \\
= \ & \exists P, Q, p, q \cdot\ t = (p \ ; \ q) \ \wedge\ PP \xrightarrow{p} P \ \wedge\ QQ \xrightarrow{q} Q \\
& \qquad \wedge R = COND\,(last(p) = \checkmark, \ (Q \ ; \ P), \ P)
\end{aligned}
$$

Here, $COND(true, e1, e2) = e1$ and $COND(false, e1, e2) = e2$.

The following equations are derived from the operational rules which will support to prove the above lemma:

- $(PP \; ; \; QQ) \xrightarrow{\omega} R \;=\; \exists\, P, Q \cdot PP \xrightarrow{\checkmark} P \;\wedge\; QQ \xrightarrow{a} Q \;\wedge\; R = (Q \; ; \; P) \;\vee\; PP \xrightarrow{\omega} R \;\wedge\; \omega \neq \checkmark$
- $(PP \; ; \; QQ) \xrightarrow{a} RR \;=\; \exists\, PP' \cdot PP \xrightarrow{a} PP' \;\wedge\; RR = (PP' \; ; \; QQ) \;\vee\; \exists\, P, QQ' \cdot PP \xrightarrow{\checkmark} P \;\wedge\; QQ \xrightarrow{a} QQ' \;\wedge\; R = \langle QQ', P\rangle$

In the inductive proof of the above lemma we get an intermediate step involving the auxiliary construct $\langle QQ, P\rangle$.

$$\begin{aligned}
(PP \; ; \; QQ) \xrightarrow{\langle a\rangle t} R \;&=\; \exists\, RR \cdot (PP \; ; \; QQ) \xrightarrow{a} RR \wedge RR \xrightarrow{t} R \\
&=\; \exists\, PP' \cdot \; PP \xrightarrow{a} PP' \;\wedge\; (PP' \; ; \; QQ) \xrightarrow{t} R \qquad (3)\\
&\vee\; \exists\, P, QQ' \cdot \; PP \xrightarrow{\checkmark} P \;\wedge\; QQ \xrightarrow{a} QQ' \;\wedge\; \langle QQ', P\rangle \xrightarrow{t} R \qquad (4)
\end{aligned}$$

To deal with this we need another lemma which will support to removing the auxiliary construct in equation 4. This lemma will deal with the situation where the forward behaviour of the first process of sequential composition is terminated with ✓ and its compensation is stored and the second process of the composition has started.

Lemma 5
$\langle QQ, P\rangle \xrightarrow{t} R \;=\; \exists\, Q \cdot \; QQ \xrightarrow{t} Q \;\wedge\; R = (Q \; ; P)$

The lemma is proved by induction over t and helps to prove Lemma 4.

3.2 Parallel Composition

Parallel composition of two processes is defined to be the interleaving of their observable events followed by the synchronisation of their terminal events. For example, considering asynchronous events, the execution of $A\|B$ will be either A followed by B or B followed by A. For traces p and q, we write $(p \;|||\; q)$ to denote the set of interleaving of p and q and it has the following definition:

$$\begin{aligned}
\langle\rangle \in (p \;|||\; q) \;&=\; p = \langle\rangle \;\wedge\; q = \langle\rangle \\
\langle a\rangle t \in (p \;|||\; q) \;&=\; \exists\, p' \cdot \; p = \langle a\rangle p' \;\wedge\; t \in (p' \;|||\; q) \\
&\vee\; \exists\, q' \cdot \; q = \langle a\rangle q' \;\wedge\; t \in (p \;|||\; q')
\end{aligned}$$

Standard Processes
Similar to our earlier approach of standard processes, we let $t \in DT(P\|Q)$ and then we get:

$$t \in DT(P\|Q) \;=\; (P\|Q) \xrightarrow{t} 0$$

We need to prove of the following lemma to draw the correspondence.

Lemma 6
$(P \| Q) \xrightarrow{t} 0 \quad = \quad \exists\, p, q \cdot\; t \in (p \| q) \;\wedge\; P \xrightarrow{p} 0 \;\wedge\; Q \xrightarrow{q} 0$

From the operational rules we derive the following equations to help us prove the above lemma.

- $P \| Q \xrightarrow{\omega} 0 \;\; = \;\; P \xrightarrow{\omega 1} 0 \;\wedge\; Q \xrightarrow{\omega 2} 0 \;\wedge\; \omega \;=\; \omega 1 \& \omega 2$
- $P \| Q \xrightarrow{a} R \;\; = \;\; (\exists\, P' \cdot P \xrightarrow{a} P' \;\wedge\; R = (P' \| Q)) \;\; \vee \;\; (\exists\, Q' \cdot Q \xrightarrow{a} Q' \;\wedge\; R = (P \| Q'))$

Compensable Processes
For the parallel composition of compensable we let $(t, t') \in DT(PP \| QQ)$ and then by following trace derivation rule we get,

$$(t, t') \in DT(PP \| QQ) \quad = \quad \exists\, R \cdot (PP \| QQ) \xrightarrow{t} R \;\wedge\; R \xrightarrow{t'} 0$$

The supporting lemma that we need is as follows:

Lemma 7
$(PP \parallel QQ) \xrightarrow{t} R$
$= \;\; \exists\, P, Q, p, q \cdot\; t \in (p \| q) \;\wedge\; PP \xrightarrow{p} P \;\wedge\; QQ \xrightarrow{q} P \;\wedge\; R = P \parallel Q$

From the operational rules for parallel composition of compensable processes we get the following equations which will help to prove the above lemma.

- $PP \| QQ \xrightarrow{\omega} R = (\exists\, P, Q \cdot PP \xrightarrow{\omega 1} P \;\;\wedge\;\; QQ \xrightarrow{\omega 2} Q \;\wedge\; R = (P \| Q) \;\wedge\; \omega = \omega 1 \& \omega 2)$
- $PP \| QQ \xrightarrow{a} RR = (\exists\, PP' PP \xrightarrow{a} PP' \wedge R = (PP' \| QQ))$
  $\vee \;\; (\exists\, QQ' \cdot QQ \xrightarrow{a} QQ' \wedge R = (PP \| QQ'))$

3.3 Transaction Block

This section derives the correspondence between the two semantics of transaction block. Transaction block is a standard process and we let $t \in DT([PP])$. Following the trace derivation rule we get

$$t \in DT([PP]) \quad = \quad [PP] \xrightarrow{t} 0$$

The correspondence proof needs the following supporting lemma:

Lemma 8
$[PP] \xrightarrow{t} 0 \quad = \quad \exists\, p, p' \cdot t = [p, p'] \;\wedge\; PP \xrightarrow{p, p'} 0 \;\wedge\; \mathit{last}(p) \neq ?$

The operational semantics entail the following equations which help to prove the above lemma:

- $[PP] \xrightarrow{\omega} 0 \quad = \quad PP \xrightarrow{\checkmark} P \;\vee\; PP \xrightarrow{!} P \;\wedge\; P \xrightarrow{\omega} 0$
- $[PP] \xrightarrow{a} R \quad = \quad (\exists\, PP' \cdot PP \xrightarrow{a} PP' \wedge R = [PP']) \;\vee\; (\exists\, P \cdot PP \xrightarrow{!} P \;\wedge\; P \xrightarrow{a} R)$

The transaction block operator runs the compensation of a terminating forward behaviour and discards the compensation of successfully completed forward behaviour. It removes the traces of an yielding forward behaviour.

3.4 Compensation Pair

Compensation pair is a compensable process. Let $(t, t') \in (P \div Q)$ and following the trace derivation rule we have

$$(t, t') \in DT(P \div Q) \;=\; (P \div Q) \xrightarrow{(t,t')} 0$$

Then we need to prove the following lemma:

Lemma 9
$(P \div Q) \xrightarrow{(t,t')} 0 \quad = \quad \exists\, p, q \cdot (t, t') = (p \div q) \;\wedge\; (P \div Q) \xrightarrow{p,q} 0$

The following supporting equations are derived from operational rules of compensation pair:

- $(P \div Q) \xrightarrow{\omega} R \quad = \quad P \xrightarrow{\checkmark} 0 \wedge R = Q \;\vee\; P \xrightarrow{\omega} 0 \;\wedge\; R = \mathit{SKIP} \wedge \omega \neq \checkmark$
- $(P \div Q) \xrightarrow{a} RR \quad = \quad P \xrightarrow{a} P' \;\wedge\; R = P' \div Q$

SKIP,*THROW* and *YIELD* are the primitive processes of cCSP and their compensable counterparts are defined as:

$$\begin{gathered} \mathit{SKIPP} = \mathit{SKIP} \div \mathit{SKIP} \\ \mathit{THROWW} = \mathit{THROW} \div \mathit{SKIP} \\ \mathit{YIELDD} = \mathit{YIELD} \div \mathit{SKIP} \end{gathered}$$

The correspondence proofs are trivial and omitted from this paper.

We left out two of the operators from the correspondence proof shown here. One of them is choice operator $(P \,\Box\, Q)$. Correspondence proof of this operator is fairly since traces of choice is just the union of the traces of each process. Another operator is the interrupt handler $(P \rhd Q)$. Its definition is similar to standard sequential composition except that the flow of control from first to second process is caused by a throw (!) rather than a ✓ and its correspondence proof is dual of the proof of sequential composition.

3.5 Correspondence Derivation

The correspondence is shown separately for standard and compensable processes. For each term of the language the correspondence between the two semantics are shown by using structural induction. For example, we have shown that $DT(P \,;\, Q) \;=\; T(P \,;\, Q)$ and in order to prove this we consider $DT(P) = T(P)$ and $DT(Q) = T(Q)$. Similarly, for all the terms of the language we have shown the inductive proof assuming the base case. Having the proofs we can show that for a standard processes P (not containing 0)

$$t \in DT(P) \;=\; t \in T(P)$$

And similarly for a compensable process PP (not containing 0 or $\langle PP, P\rangle$)

$$(t, t') \in DT(PP) \;=\; (t, t') \in T(PP)$$

4 Lessons Learned

We have adopted a systematic approach to show the correspondence between the two semantics of cCSP. Traces are derived from the operational rules and then by applying induction over the derived traces we showed the correspondence.

In the original definition of trace semantics each process was defined as a non-empty set of traces followed by a trace of a terminal event ($\checkmark, !, ?$) and the nature of a trace is indicated by this final symbol. We defined the operational semantics by using labelled transition systems. Transition rules for normal events and terminal events were defined by separate symbols. Having separate symbols as labels allows us to extract traces of normal events and terminal events easily and helps to prove the correspondence.

The correspondence of the two semantics was proved by structural induction. Two levels of induction was applied in the proof. In one level induction was applied on individual derived traces and in another level induction was on process terms of the language. For example, Lemma 3 was on sequential operator for individual traces. This was then easily lifted to the set of traces to prove the correspondence.

5 Related Work

Hoare and He [6] presented the idea of unifying different programming paradigms and showed the way of deriving operational semantics from its denotational presentation of a sequential language. They derive algebraic presentation from the denotational definition and then derive the operational semantics from the algebraic laws. Similar to our work, Huibiao *et al.* [13] derived denotational semantics from operational semantics for a subset of Verilog [4]. However the derivation was done in a different way than our method where the authors defined transitional condition and phase semantics from the operational semantics. The denotational semantics are derived from the sequential composition of the phase semantics. The authors also derived operational semantics from denotational semantics [12].

Unlike our approach, the unification between the two semantics was shown in [11], by extending the operational semantics to incorporate the denotational properties. The equivalence was shown for a language having simple models without any support for concurrency. Similar problem was also investigated in [7] for a simple sequential language, which support recursion and synchronisation in the form of interleaving. The relation between operational and denotational semantics is obtained via an intermediate semantics. A comparison of the operators of cCSP with another language having similar operators including compensation pairs and transaction blocks can be found in [1].

6 Conclusions and Future Work

Demonstrating the relationship between the two semantics of a language ensures the consistency of the whole semantic description of the language. The main contribution of this paper is to show the correspondence between the operational and the trace semantics of cCSP. The correspondence is shown by deriving the traces from the operational rules and then applying the induction over the derived traces. Two level of induction is applied. In one level, induction is applied over the terms of the language and in the next level induction is applied over the derived traces.

The correspondence shown here are completely done by hand and there are strong possibilities to miss some of the important parts during the proof. As part of the future work our goal is to use an automated/mechanized prover which will help us to use mathematical induction, and prove the theorems automatically. We are investigating the use of Prototype Verification System (PVS) [8][9] for our purpose. The specification language of PVS is based on classical, typed, high order logic and contains the constructs intended to ease the natural development of specification.

The parallel operator of cCSP does not support synchronization on normal events. Synchronization of events is significant for the development of a language. Currently we are working on adding synchronization to cCSP. Adding synchronization and then using mechanized theorem prover to show the correspondence will strengthen the formal foundation of the language.

7 Acknowledgements

Special thanks to Tony Hoare, Roberto Bruni, Hernán Melgratti.

References

1. Roberto Bruni, Michael Butler, Carla Ferreira, Tony Hoare, Hernan Melgratti, and Ugo Montanari. Comparing two approaches to compensable flow composition. In Martn Abadi and Luca de Alfaro, editors, *CONCUR 2005*, volume 3653 of *LNCS*, pages 383–397, 2005.
2. Michael Butler, Tony Hoare, and Carla Ferreira. A trace semactics for long-running transaction. In A.E. Abdallah, C.B. Jones, and J.E. Sanders, editors, *Proceedings of 25 Years of CSP*, volume 3525 of *LNCS*, London, 2004. Springer-Verlag.
3. Michael Butler and Shamim Ripon. Executable semantics for compensating CSP. In Mario Bravetti, Leïla Kloul, and Gianluigi Zavattaro, editors, *WS-FM 2005*, volume 3670 of *LNCS*, pages 243–256, Versailles, France, September 1-3 2005. Springer-Verlag.
4. Mike Gordon. The semantic challenge of Verilog HDL. In *Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science (LICS '95:)*, pages 136–145. IEEE Computer Society, June 1995.
5. C.A.R. Hoare. *Communicating Sequential Process.* Prentice Hall, 1985.
6. C.A.R. Hoare and He Jifeng. *Unifying Theories of Programming.* Prentice Hall International Series in Computer Science, 1998.
7. J.-J. Ch. Meyer and E.P.de Vink. *On Relating Denotational and Operational Semantics for Programming Languages with Recursion and Concurrency*, chapter 24, pages 387–406. Elsevier, 1990.

8. S. Owre, J.M. Rushby, and N Shankar. PVS: A Prototype Verification System. In Deepak Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *LNCS*, pages 748–752. Springer-Verlag, June 1992.
9. Sam Owre, John Rushby, Natarajan Shankar, and Friedrich von Henke. Formal verification for fault-tolerant architectures: Prolegomena to the design of PVS. *IEEE Transactions on Software Engineering*, 21(2):107–125, February 1995.
10. G. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Aarhus University, Computer Science Department, September 1981.
11. Scott F. Smith. From operational to denotational semantics. In *Proceedings of the 7th International Conference on Mathematical Foundations of Programming Semantics*, volume 598 of *LNCS*, pages 54–76, 1992.
12. Huibiao Zhu, Jonathan P. Bowen, and Jifeng He. Deriving operational semantics from denotational semantics for Verilog. In *8th Asia-Pacific Software Engineering Conference (APSEC 2001)*, pages 177–184. IEEE Computer Society, 4-7 Dec 2001.
13. Huibiao Zhu, Jonathan P. Bowen, and Jifeng He. From operational semantics to denotational semantics for Verilog. In Tiziana Margaria and Thomas F. Melham, editors, *CHARME 2001*, volume 2144 of *LNCS*, pages 449–466, 2001.

A Appendix

(For reviewer's only, not for the final version)

Here we show the proof of the lemmas shown in the main text in Section 3. By using the lemmas the correspondence derivation are given here as well.

A.1 Standard Sequential Composition

Lemma 3
$(P\ ;\ Q) \xrightarrow{t} 0 = \exists\, p, q \cdot\ t = (p\ ;\ q)\ \wedge\ P \xrightarrow{p} 0\ \wedge\ Q \xrightarrow{q} 0$

Basic step: Case - $\langle\omega\rangle$
$(P\ ;\ Q) \xrightarrow{\langle\omega\rangle} 0 = P\ ;\ Q \xrightarrow{\omega} 0$
"From operational rules $(S2)$, $(S3)$ of standard sequential composition"

$$= \quad P \xrightarrow{\checkmark} 0\ \wedge\ Q \xrightarrow{\omega} 0 \tag{5}$$
$$\vee\, P \xrightarrow{\omega} 0\ \wedge\ \omega \neq \checkmark \tag{6}$$

From (5)
$\quad P \xrightarrow{\checkmark} 0\ \wedge\ Q \xrightarrow{\omega} 0$
$=\ \exists\, p, q \cdot\ p = \langle\checkmark\rangle\ \wedge\ q = \langle\omega\rangle\ \wedge\ \langle\omega\rangle = (p\ ;\ q)\ \wedge\ P \xrightarrow{p} 0\ \wedge\ Q \xrightarrow{q} 0$
$=\ \exists\, p, q \cdot \langle\omega\rangle = (p\ ;\ q)\ \wedge\ p = \langle\checkmark\rangle\ \wedge\ P \xrightarrow{p} 0\ \wedge\ Q \xrightarrow{q} 0$

From (6)
$\quad P \xrightarrow{\omega} 0\ \wedge\ \omega \neq \checkmark$
$=\ \exists\, p, q \cdot p = \langle\omega\rangle\ \wedge\ \omega \neq \checkmark\ \wedge\ \langle\omega\rangle = (p\ ;\ q)\ \wedge\ P \xrightarrow{p} 0\ \wedge\ Q \xrightarrow{q} 0$
$=\ \exists\, p, q \cdot \langle\omega\rangle = (p\ ;\ q)\ \wedge\ p \neq \langle\checkmark\rangle\ \wedge\ P \xrightarrow{p} 0\ \wedge\ Q \xrightarrow{q} 0$

Therefore, for $\langle\omega\rangle$
$\quad \exists\, p, q \cdot \langle\omega\rangle = (p\ ;\ q)\ \wedge\ p = \langle\checkmark\rangle\ \wedge\ P \xrightarrow{p} 0\ \wedge\ Q \xrightarrow{q} 0$
$\vee\ \ \exists\, p, q \cdot \langle\omega\rangle = (p\ ;\ q)\ \wedge\ p \neq \langle\checkmark\rangle\ \wedge\ P \xrightarrow{p} 0\ \wedge\ Q \xrightarrow{q} 0$
$=\ \exists\, p, q \cdot \langle\omega\rangle = (p\ ;\ q)\ \wedge\ P \xrightarrow{p} 0\ \wedge\ Q \xrightarrow{q} 0$

A.2 Compensable Sequential Composition

Lemma 4
$\quad (PP\ ;\ QQ) \xrightarrow{t} R$
$= \exists\, P, Q, p, q \cdot\ t = (p\ ;\ q)\ \wedge\ PP \xrightarrow{p} P\ \wedge\ QQ \xrightarrow{q} Q$
$\qquad\qquad \wedge\ R = \mathit{COND}\,(\mathit{last}(p) = \checkmark,\ (Q\ ;\ P),\ P)$

Basic step: Case - $\langle\omega\rangle$
$(PP\ ;\ QQ) \xrightarrow{\langle\omega\rangle} R\ =\ (PP\ ;\ QQ) \xrightarrow{\omega} R$

From operational rules $(CS2)$ and $(CS3)$ of compensable sequential processes

$$= \exists\, P, Q \cdot PP \xrightarrow{\checkmark} P \ \wedge\ QQ \xrightarrow{\omega} Q \ \wedge\ R = (Q\ ;\ P) \tag{7}$$
$$\vee\ \exists\, P \cdot PP \xrightarrow{\omega} P \ \wedge\ \omega \neq \checkmark \ \wedge\ R = P \tag{8}$$

From (7)
$\exists\, P, Q \cdot PP \xrightarrow{\checkmark} P \ \wedge\ QQ \xrightarrow{\omega} Q \ \wedge\ R = Q\ ;\ P$
$= \exists\, P, Q, p, q \cdot p = \langle \checkmark \rangle \ \wedge\ q = \langle \omega \rangle \ \wedge\ PP \xrightarrow{p} P \ \wedge\ QQ \xrightarrow{q} Q \ \wedge\ R = (Q\ ;\ P)$
$= \exists\, P, Q, p, q \cdot \langle \omega \rangle = (p\ ;\ q) \ \wedge\ PP \xrightarrow{p} P \ \wedge\ QQ \xrightarrow{q} Q \ \wedge\ R = (Q\ ;\ P)$

From (8)
$\exists\, P \cdot PP \xrightarrow{\omega} P \ \wedge\ \omega \neq \checkmark \ \wedge\ R = P$
$= \exists\, P, p \cdot p = \langle \omega \rangle \ \wedge\ \omega \neq \checkmark \ \wedge\ PP \xrightarrow{p} P \ \wedge\ R = P$
$= \exists\, P, p \cdot \langle \omega \rangle = p \ \wedge\ PP \xrightarrow{p} P \ \wedge\ R = P$

Therefore, for $\langle \omega \rangle$, from (7) $\vee$ (8)
$\exists\, P, Q, p, q \cdot \langle \omega \rangle = (p\ ;\ q) \ \wedge\ PP \xrightarrow{p} P \ \wedge\ QQ \xrightarrow{q} Q \ \wedge\ R = (Q\ ;\ P)$
$\vee\ \exists\, P, p \cdot \langle \omega \rangle = p \ \wedge\ PP \xrightarrow{p} P \ \wedge\ R = P$

The main difference between the two equation of the above disjunction is that whether or not, $last(p) = \checkmark$. The sequence operator handles this in such a way that when $last(p) = \checkmark$, then behaviour of QQ is accepted and is augmented with behaviour of PP, otherwise behaviour of QQ is discarded. The above two equations can be combined by the expression $COND$.

$$= \quad \exists\, P, Q, p, q \cdot \langle \omega \rangle = (p\ ;\ q) \ \wedge\ PP \xrightarrow{p} P \ \wedge\ QQ \xrightarrow{q} Q$$
$$\wedge\ R = COND(last(p) = \checkmark, (Q\ ;\ P), P)$$

Inductive step: Case - $\langle a \rangle t$
$(PP\ ;\ QQ) \xrightarrow{\langle a \rangle t} R = \exists\, RR \cdot (PP\ ;\ QQ) \xrightarrow{a} RR \ \wedge\ RR \xrightarrow{t} R$
From operational rules $(CS1)$ and $(CS4)$ of compensable sequential processes

$$= \exists\, PP' \cdot PP \xrightarrow{a} PP' \ \wedge\ (PP'\ ;\ QQ) \xrightarrow{t} R \tag{9}$$
$$\vee\ \exists\, P, QQ' \cdot PP \xrightarrow{\checkmark} P \ \wedge\ QQ \xrightarrow{a} QQ' \ \wedge\ \langle QQ', P \rangle \xrightarrow{t} R \tag{10}$$

From (9)
$\exists\, PP' \cdot PP \xrightarrow{a} PP' \ \wedge\ (PP'\ ;\ QQ) \xrightarrow{t} R$
$=$ " by inductive hypothesis"
$\exists\, PP' \cdot PP \xrightarrow{a} PP' \ \wedge\ \exists\, P, Q, p', q \cdot t = (p'\ ;\ q) \ \wedge\ PP' \xrightarrow{p'} P \ \wedge\ QQ \xrightarrow{q} Q$
$\wedge\ R = COND(last(p) = \checkmark, (Q\ ;\ P), P)$
Here $COND$ expression handles both the cases, whether or not $last(p) = \checkmark$.

$= $ "Combining existential quantifications"

$\exists\, P, Q, p', q \cdot t = (p' \;;\; q) \;\wedge\; PP \xrightarrow{\langle a\rangle p'} P \;\wedge\; QQ \xrightarrow{q} Q$
$\qquad\qquad\qquad \wedge\; R = COND(last(p) = \checkmark, (Q \;;\; P), P)$

$= \exists\, P, Q, p, q \cdot\; p = \langle a\rangle p' \;\wedge\; t = (p' \;;\; q) \;\wedge\; PP \xrightarrow{p} P \;\wedge\; QQ \xrightarrow{q} Q$
$\qquad\qquad\qquad \wedge\; R = COND(last(p) = \checkmark, (Q \;;\; P), P)$

$= $ "$\langle a\rangle t \;=\; \langle a\rangle(p' \;;\; q) \;=\; (\langle a\rangle p') \;;\; q \;=\; (p \;;\; q)$"

$\exists\, P, Q, p, q \cdot\; \langle a\rangle t = (p \;;\; q) \;\wedge\; PP \xrightarrow{p} P \;\wedge\; QQ \xrightarrow{q} Q$
$\qquad\qquad\qquad \wedge\; R = COND(last(p) = \checkmark, (Q \;;\; P), P)$

From (10)

$\exists\, P, QQ' \cdot\; PP \xrightarrow{\checkmark} P \;\wedge\; QQ \xrightarrow{a} QQ' \;\wedge\; \langle QQ', P\rangle \xrightarrow{t} R$

$= $ " by using Lemma 5"

$\exists\, P, QQ' \cdot\; PP \xrightarrow{\checkmark} P \;\wedge\; QQ \xrightarrow{a} QQ' \;\wedge\; \exists\, Q \cdot QQ' \xrightarrow{t} Q \;\wedge\; R = (Q \;;\; P)$

$= $ "Combining existential quantifications"

$\exists\, P, Q \cdot\; PP \xrightarrow{\checkmark} P \;\wedge\; QQ \xrightarrow{\langle a\rangle t} Q \;\wedge\; R = (Q \;;\; P)$

$= \exists\, P, Q, p, q \cdot\; p = \langle \checkmark \rangle \;\wedge\; q = \langle a\rangle t \;\wedge\; \langle a\rangle t = (p \;;\; q) \;\wedge\; PP \xrightarrow{p} P \;\wedge\; QQ \xrightarrow{q} Q \;\wedge\; R = (Q \;;\; P)$

Therefore, for $\langle a\rangle t$, from (9) $\vee$ (10)

$\exists\, P, Q, p, q \cdot\; \langle a\rangle t = (p \;;\; q) \;\wedge\; PP \xrightarrow{p} P \;\wedge\; QQ \xrightarrow{q} Q$
$\qquad\qquad\qquad \wedge\; R = COND(last(p) = \checkmark, (Q \;;\; P), P)$

$\vee\; \exists\, P, Q, p, q \cdot\; \langle a\rangle t = (p \;;\; q) \;\wedge\; PP \xrightarrow{p} P \;\wedge\; QQ \xrightarrow{q} Q \;\wedge\; R = (Q \;;\; P)$

"2nd part of the disjunction is same as the 1st part when $last(p) = \checkmark$ in $COND$ expression.

$= \exists\, P, Q, p, q \cdot\; \langle a\rangle t = (p \;;\; q) \;\wedge\; PP \xrightarrow{p} P \;\wedge\; QQ \xrightarrow{q} Q$
$\qquad\qquad\qquad \wedge\; R = COND(last(p) = \checkmark, (Q \;;\; P), P)$

Lemma 5:

$\langle QQ, P\rangle \xrightarrow{t} R$

$= \exists\, Q \cdot\; QQ \xrightarrow{t} Q \;\wedge\; R = Q \;; P$

Basic step: Case - $\langle \omega \rangle$

$\langle QQ, P\rangle \xrightarrow{\langle\omega\rangle} R = \langle QQ, P\rangle \xrightarrow{\omega} R$

$= $ "From operational rule ($CS6$)"

$\exists\, Q \cdot\; QQ \xrightarrow{\omega} Q \;\wedge\; R = Q \;; P$

Inductive step: Case - $\langle a\rangle t$

$\langle QQ, P\rangle \xrightarrow{\langle a\rangle t} R$

$= \exists\, RR \cdot \langle QQ, P\rangle \xrightarrow{a} RR \;\wedge\; RR \xrightarrow{t} R$

$= $ "From operational rule ($CS5$) "

$\exists\, QQ' \cdot\; QQ \xrightarrow{a} QQ' \;\wedge\; \langle QQ', P\rangle \xrightarrow{t} R$

$= $ "inductive hypothesis"

$\exists\, QQ' \cdot\; QQ \xrightarrow{a} QQ' \;\wedge\; \exists\, Q \cdot\; QQ' \xrightarrow{t} Q \;\wedge\; R = Q \;;\; P$

$= $ "combining existential quantification"

$\exists\, Q \cdot\; QQ \xrightarrow{\langle a\rangle t} Q \;\wedge\; R = Q \;;\; P$

Deriving correspondence:

$(t, t') \in DT(PP \ ; \ QQ)$

$= \quad (PP \ ; \ QQ) \xrightarrow{(t,t')} 0$

$= \quad \exists\, R \cdot (PP \ ; QQ) \xrightarrow{t} R \ \wedge \ R \xrightarrow{t'} 0$

$=$ "by using Lemma 4"

$\exists\, P, Q, p, q \cdot t = (p \ ; \ q) \ \wedge \ PP \xrightarrow{p} P \ \wedge \ QQ \xrightarrow{q} Q$

$\wedge \ R = \ COND(last(p) = \checkmark,\ (Q \ ; \ P),\ P) \ \wedge \ R \xrightarrow{t'} 0$

We now consider both the cases where p ends with and without $\checkmark$ and we separate these two conditions. The sequential composition operator is defined in a way so that when $last(p) \neq \checkmark$, the traces of QQ are discarded

$= \quad \exists\, P, Q, p, q \cdot \ t = (p\langle\checkmark\rangle \ ; \ q) \ \wedge \ PP \xrightarrow{p\langle\checkmark\rangle} P \ \wedge \ QQ \xrightarrow{q} Q \ \wedge \ (Q \ ; \ P) \xrightarrow{t'} 0$

$\vee \ \exists\, P, p \cdot \ t = p\langle\omega\rangle \ \wedge \ \omega \neq \checkmark \ \wedge \ PP \xrightarrow{t} P \ \wedge \ P \xrightarrow{t'} 0$

$=$ "by using Lemma 3"

$\exists\, P, Q, p, q \cdot \ t = (p\langle\checkmark\rangle \ ; \ q) \ \wedge \ PP \xrightarrow{p\langle\checkmark\rangle} P \ \wedge \ QQ \xrightarrow{q} Q$

$\wedge \ \exists\, p', q' \cdot \ t' = (q' \ ; \ p') \ \wedge \ Q \xrightarrow{q'} 0 \ \wedge \ P \xrightarrow{p'} 0$

$\vee \ \exists\, P, p, p' \cdot \ t = p\langle\omega\rangle \ \wedge \ t' = p' \ \wedge \ PP \xrightarrow{t} P \ \wedge \ P \xrightarrow{t'} 0 \ \wedge \ \omega \neq \checkmark$

$=$ "combining existential quantification"

$\exists\, p, p', q, q' \cdot \ t = (p\langle\checkmark\rangle \ ; \ q) \ \wedge \ t' = (q' \ ; \ p') \ \wedge \ PP \xrightarrow{p\langle\checkmark\rangle, p'} 0 \ \wedge \ QQ \xrightarrow{q, q'} 0$

$\vee \ \exists\, p, p' \cdot \ t = p\langle\omega\rangle \ \wedge \ t' = p' \ \wedge \ PP \xrightarrow{p\langle\omega\rangle, p'} 0 \ \wedge \ \omega \neq \checkmark$

$=$ "using the rules for derived traces "

$\exists\, p, p', q, q' \cdot \ t = (p\langle\checkmark\rangle \ ; \ q) \ \wedge \ t' = (q' \ ; \ p') \ \wedge \ (p\langle\checkmark\rangle, p') \in DT(PP) \ \wedge \ (q, q') \in DT(QQ)$

$\vee \ \exists\, p, p' \cdot \ t = p\langle\omega\rangle \ \wedge \ \omega \neq \checkmark \ \wedge \ t' = p' \ \wedge \ (p\langle\omega\rangle, p') \in DT(PP)$

$=$ "Structural Induction"

$\exists\, p, p', q, q' \cdot \ t = (p\langle\checkmark\rangle \ ; \ q) \ \wedge \ t' = (q' \ ; \ p') \ \wedge \ (p\langle\checkmark\rangle, p') \in T(PP) \ \wedge \ (q, q') \in T(QQ)$

$\vee \ \exists\, p, p' \cdot \ t = p\langle\omega\rangle \ \wedge \ \omega \neq \checkmark \wedge \ t' = p' \ \wedge (p\langle\omega\rangle, p') \in T(PP)$

We have $(t, t') = (p\langle\checkmark\rangle \ ; \ q),\ (q' \ ; \ p')$ or $(t, t') = (p\langle\omega\rangle, p')$

Using trace rules we can write that:

$(p\langle\checkmark\rangle \ ; \ q), (q' \ ; \ p') = (p\langle\checkmark\rangle, p') \ ; \ (q, q')$

Similarly, using the definition of sequential composition over traces

$p\langle\omega\rangle, p' = (p\langle\omega\rangle, p') \ ; \ (q, q')$ **where** $\omega \neq \checkmark$

$= \quad (t, t') \in T(PP \ ; \ QQ)\square$

A.3 Standard Parallel Composition

Lemma 6
$P \| Q \xrightarrow{t} 0 \;=\; \exists\, p, q \cdot\; t \in (p \| q) \;\wedge\; P \xrightarrow{p} 0 \;\wedge\; Q \xrightarrow{q} 0$

Basic step: Case - $\langle \omega \rangle$

$P \| Q \xrightarrow{\langle \omega \rangle} 0 = P \| Q \xrightarrow{\omega} 0$
$=$ "From operational rule $(P3)$ "
$\exists\, \omega 1, \omega 2 \cdot P \xrightarrow{\langle \omega 1 \rangle} 0 \;\wedge\; Q \xrightarrow{\langle \omega 2 \rangle} 0 \;\wedge\; \omega = \omega 1 \& \omega 2$
$= \exists\, p, q \cdot\; p = \langle \omega 1 \rangle \;\wedge\; q = \langle \omega 2 \rangle \;\wedge\; \langle \omega \rangle \in (p \| q) \;\wedge\; P \xrightarrow{p} 0 \;\wedge\; Q \xrightarrow{q} 0$
$= \exists\, p, q \cdot\; \langle \omega \rangle \in (p \| q) \;\wedge\; P \xrightarrow{p} 0 \;\wedge\; Q \xrightarrow{q} 0$

Inductive step: Case - $\langle a \rangle t$

$P \| Q \xrightarrow{\langle a \rangle t} 0$
$= \exists\, R \cdot P \| Q \xrightarrow{\langle a \rangle} R \;\wedge\; R \xrightarrow{t} 0$
$=$ "Using the operational rules $(P1)$ and $(P2)$"
$\exists\, P' \cdot P \xrightarrow{a} P' \;\wedge\; P' \| Q \xrightarrow{t} 0$
$\vee\; \exists\, Q' \cdot Q \xrightarrow{a} Q' \;\wedge\; P \| Q' \xrightarrow{t} 0$
$=$ "Inductive hypothesis"
$\exists\; P' \cdot\; P \xrightarrow{a} P' \;\wedge\; \exists\, p', q \cdot\; t \in (p' \| q) \;\wedge\; P' \xrightarrow{p'} 0 \;\wedge\; Q \xrightarrow{q} 0$
$\vee\; \exists\; Q' \cdot\; Q \xrightarrow{a} Q' \;\wedge\; \exists\, p, q' \cdot\; t \in (p \| q') \;\wedge\; P \xrightarrow{p} 0 \;\wedge\; Q' \xrightarrow{q'} 0$
$=$ "Combining existential quantifications"
$= \exists\, p', q \cdot\; t \in (p' \| q) \;\wedge\; P \xrightarrow{\langle a \rangle p'} 0 \;\wedge\; Q \xrightarrow{q} 0$
$\vee\; \exists\, p, q' \cdot\; t \in (p \| q') \;\wedge\; P \xrightarrow{p} 0 \;\wedge\; Q \xrightarrow{\langle a \rangle q'} 0$
$= \exists\, p, q \cdot p = \langle a \rangle p' \;\wedge\; t \in (p' \| q) \;\wedge\; P \xrightarrow{p} 0 \;\wedge\; Q \xrightarrow{q} 0$
$\vee\; \exists\, p, q \cdot q = \langle a \rangle q' \;\wedge\; t \in (p \| q') \;\wedge\; P \xrightarrow{p} 0 \;\wedge\; Q \xrightarrow{q} 0$
$=$ "Combining"
$\exists\, p, q \cdot\; (p = \langle a \rangle p' \;\wedge\; t \in (p' \| q) \;\vee\; q = \langle a \rangle q' \;\wedge\; t \in (p \| q')) \;\wedge\; P \xrightarrow{p} 0 \;\wedge\; Q \xrightarrow{q} 0$
$=$ "By the definition the interleaving of traces"
$\exists\, p, q \cdot \langle a \rangle t \in (p \| q) \;\wedge\; P \xrightarrow{p} 0 \;\wedge\; Q \xrightarrow{q} 0$

Deriving Correspondence:

$t \in DT(P \parallel Q) = P \parallel Q \xrightarrow{t} 0$
$=$ "Using Lemma 6"
$\exists\, p, q \cdot\; t \in (p \| q) \;\wedge\; P \xrightarrow{p} 0 \;\wedge\; Q \xrightarrow{q} 0$
$=$ "using the rules for derived traces"

$\exists\, p, q \cdot t \in (p \| q) \;\wedge\; p \in DT(P) \;\wedge\; q \in DT(Q)$

$=$ "structural induction"

$\exists\, p, q \cdot t \in (p \| q) \wedge p \in T(P) \wedge q \in T(Q)$

$=$ "by trace rule"

$t \in T(P \parallel Q)\Box$

A.4 Compensable Parallel Composition

Lemma 7

$PP \| QQ \xrightarrow{t} R$

$= \exists\, P, Q, p, q \cdot\; t \in (p \| q) \;\wedge\; PP \xrightarrow{p} P \;\wedge\; QQ \xrightarrow{q} P \;\wedge\; R = P \| Q$

Basic step: Case- $\langle \omega \rangle$

$PP \| QQ \xrightarrow{\langle \omega \rangle} R$

$= PP \| QQ \xrightarrow{\omega} R$

$=$ "From operational rule ($CP3$) of parallel composition"

$\exists\, P, Q \cdot \omega = \omega 1 \& \omega 2 \;\wedge\; \omega 1 \& \omega 2 \in (\langle \omega 1 \rangle \| \langle \omega 2 \rangle) \;\wedge\; PP \xrightarrow{\omega 1} P \;\wedge\; QQ \xrightarrow{\omega 2} Q \;\wedge\; R = (P \| Q)$

$= \exists\, P, Q, p, q \cdot\; p = \langle \omega 1 \rangle \;\wedge\; q = \langle \omega 2 \rangle \;\wedge\; \langle \omega \rangle \in (p \| q) \;\wedge\; PP \xrightarrow{p} P \;\wedge\; QQ \xrightarrow{q} Q \;\wedge\; R = (P \| Q)$

$= \exists\, P, Q, p, q \cdot\; \langle \omega \rangle \in (p \| q) \;\wedge\; PP \xrightarrow{p} P \;\wedge\; QQ \xrightarrow{q} Q \;\wedge\; R = (P \| Q)$

Inductive step: Case- $\langle a \rangle t$

$PP \| QQ \xrightarrow{\langle a \rangle t} R$

$= \exists\, RR \cdot\; PP \| QQ \xrightarrow{a} RR \;\wedge\; RR \xrightarrow{t} R$

$=$ "From operational rules ($CP1$) and ($CP2$)"

$\exists\, PP' \cdot\; PP \xrightarrow{a} PP' \;\wedge\; PP' \| QQ \xrightarrow{t} R$

$\vee\; \exists\, QQ' \cdot\; QQ \xrightarrow{a} QQ' \;\wedge\; PP \| QQ' \xrightarrow{t} R$

$=$ "By inductive hypothesis"

$\exists\, PP' \cdot PP \xrightarrow{a} PP' \;\wedge\; \exists\, P, Q, p', q \cdot t \in (p' \| q) \;\wedge\; PP' \xrightarrow{p'} P \;\wedge\; QQ \xrightarrow{q} Q \;\wedge\; R = (P \| Q)$

$\vee\; \exists\, QQ' \cdot QQ \xrightarrow{a} QQ' \;\wedge\; \exists\, P, Q, p, q' \cdot t \in (p \| q') \;\wedge\; PP \xrightarrow{p} P \;\wedge\; QQ \xrightarrow{q'} Q \;\wedge\; R = (P \| Q)$

$=$ "Combining existential quantifications"

$\exists\, P, Q, p', q \cdot t \in (p' \| q) \;\wedge\; PP \xrightarrow{\langle a \rangle p'} P \;\wedge\; QQ \xrightarrow{q} Q \;\wedge\; R = (P \| Q)$

$\vee\; \exists\, P, Q, p, q' \cdot t \in (p \| q') \;\wedge\; PP \xrightarrow{p} P \;\wedge\; QQ \xrightarrow{\langle a \rangle q'} Q \;\wedge\; R = (P \| Q)$

$= \exists\, P, Q, p, q \cdot\; p = \langle a \rangle p' \;\wedge\; t \in (p' \| q) \;\wedge\; PP \xrightarrow{p} P \;\wedge\; QQ \xrightarrow{q} Q \;\wedge\; R = (P \| Q)$

$\vee\; \exists\, P, Q, p, q \cdot\; q = \langle a \rangle q' \;\wedge\; t \in (p \| q') \;\wedge\; PP \xrightarrow{p} P \;\wedge\; QQ \xrightarrow{q} Q \;\wedge\; R = (P \| Q)$

$=$ "Combining existential quantifications"

$\exists\, P, Q, p, q \cdot\; (p = \langle a \rangle p' \;\wedge\; t \in (p' \| q) \;\vee\; q = \langle a \rangle q' \;\wedge\; t \in (p \| q'))$
$\quad \wedge PP \xrightarrow{p} P \;\wedge\; QQ \xrightarrow{q} Q \;\wedge\; R = (P \| Q)$

$=$ "By the definition of interleaving"

$\exists\, P, Q, p, q \cdot\; \langle a \rangle t \in (p \| q) \;\wedge\; PP \xrightarrow{p} P \;\wedge\; QQ \xrightarrow{q} Q \;\wedge\; R = (P \| Q)$

Deriving Corrrespondence:

$(t, t') \in DT(PP \| QQ)$

$= (PP \| QQ) \xrightarrow{(t,t')} 0$

$= \exists\, R \cdot (PP \| QQ) \xrightarrow{t} R \;\wedge\; R \xrightarrow{t'} 0$

= "by Lemma 7"

$\exists\, P, Q, p, q \cdot\; t \in (p \| q) \;\wedge\; PP \xrightarrow{p} P \;\wedge\; QQ \xrightarrow{q} Q \;\wedge\; (P \| Q) \xrightarrow{t'} 0$

= "by Lemma 6"

$\exists\, P, Q, p, q \cdot\; t \in (p \| q) \;\wedge\; PP \xrightarrow{p} P \;\wedge\; QQ \xrightarrow{q} Q$

$\wedge \exists\, p', q' \cdot t' \in (p' \| q') \;\wedge\; P \xrightarrow{p'} 0 \;\wedge\; Q \xrightarrow{q'} 0$

= "combining existential quantifications"

$\exists\, p, p', q, q' \cdot\; t \in (p \| q) \;\wedge\; t' \in (p' \| q') \;\wedge\; PP \xrightarrow{p,p'} 0 \;\wedge\; QQ \xrightarrow{q,q'} 0$

= "Trace derivation rule"

$\exists\, p, p', q, q' \cdot\; t \in (p \| q) \;\wedge\; t' \in (p' \| q') \;\wedge\; (p, p') \in DT(PP) \;\wedge\; (q, q') \in DT(QQ)$

= "Structural induction"

$\exists\, p, p', q, q' \cdot\; t \in (p \| q) \wedge t' \in (p' \| q') \;\wedge\; (p, p') \in T(PP) \;\wedge\; (q, q') \in T(QQ)$

= "by trace rules of parallel composition

$(t, t') \in T(PP \| QQ) \square$

A.5 Transaction Block

Lemma 8

$[PP] \xrightarrow{t} 0 = \exists\, p, p' \cdot t = [p, p'] \;\wedge\; PP \xrightarrow{p,p'} 0 \;\wedge\; \mathit{last}(p) \neq ?$

Basic step: Case- $\langle \omega \rangle$

$[PP] \xrightarrow{\omega} 0$

"using operational rules $(T2)$ and $(T3)$"

$$= \exists\, P, p' \cdot PP \xrightarrow{\checkmark} P \;\wedge\; P \xrightarrow{p'} 0 \qquad (11)$$

$$\vee\, \exists\, P \cdot PP \xrightarrow{!} P \;\wedge\; P \xrightarrow{\omega} 0 \qquad (12)$$

From (11)

$\exists\, P, p' \cdot PP \xrightarrow{\checkmark} P \;\wedge\; P \xrightarrow{p'} 0$

$= \exists\, p, p' \cdot p = \langle\rangle \;\wedge\; PP \xrightarrow{p\langle\checkmark\rangle, p'} 0$

= "By the definition of block operation"

$\exists\, p, p' \cdot \langle \omega \rangle = p\langle\checkmark\rangle \;\wedge\; PP \xrightarrow{p\langle\checkmark\rangle, p'} 0$

From (12)

$\exists\, P \cdot PP \xrightarrow{!} P \;\wedge\; P \xrightarrow{\omega} 0$

$= \exists\, P, p, p' \cdot p = \langle\rangle \wedge p' = \langle \omega \rangle \;\wedge\; PP \xrightarrow{p\langle ! \rangle} P \;\wedge\; P \xrightarrow{p'} 0$

$= \exists\, p, p' \cdot \langle \omega \rangle = p.p' \;\wedge\; PP \xrightarrow{p\langle ! \rangle, p'} 0$

Therefore, for $\langle\omega\rangle$, from (11) $\vee$ (12)

$\exists\, p, p' \cdot \langle\omega\rangle = p\langle\checkmark\rangle \;\wedge\; PP \overset{p\langle\checkmark\rangle, p'}{\longrightarrow} 0$

$\vee\; \exists\, p, p' \cdot \langle\omega\rangle = p.p' \;\wedge\; PP \overset{p\langle!\rangle, p'}{\longrightarrow} 0$

$=$ "By the definition of block operator"

$\vee\, \exists\, p, p' \cdot \langle\omega\rangle = [p, p'] \;\wedge\; PP \xrightarrow{p,p'} 0 \;\wedge\; \mathit{last}(p) \neq ?$

Inductive step: Case - $\langle a\rangle t$

$[PP] \xrightarrow{\langle a\rangle t} 0$

$= \exists\, R \cdot [PP] \xrightarrow{a} R \wedge R \xrightarrow{t} 0$

$=$ "using operational rules $(T1)$ and $(T3)$"

$$\exists\, PP' \cdot PP \xrightarrow{a} PP' \;\wedge\; [PP'] \xrightarrow{t} 0 \tag{13}$$

$$\vee\, \exists\, P' \cdot PP \xrightarrow{!} P \;\wedge\; P \xrightarrow{a} P' \;\wedge\; P' \xrightarrow{t} 0 \tag{14}$$

From (13)

$\exists\, PP' \cdot PP \xrightarrow{a} PP' \;\wedge\; [PP'] \xrightarrow{t} 0$

$=$ "by inductive hypothesis"

$\exists\, PP' \cdot PP \xrightarrow{a} PP' \;\wedge\; \exists\, p'', p' \cdot t = [p''.p'] \;\wedge\; PP' \xrightarrow{p'', p'} 0 \;\wedge\; \mathit{last}(p'') \neq ?$

$=$ "Combining existential quantifications"

$\exists\, p'', p' \cdot t = [p'', p'] \;\wedge\; PP \overset{\langle a\rangle p'', p'}{\longrightarrow} 0 \;\wedge\; \mathit{last}(p'') \neq ?$

$= \exists\, p, p' \cdot p = \langle a\rangle p'' \;\wedge\; t = [p'', p'] \;\wedge\; PP \xrightarrow{p,p'} 0 \;\wedge\; \mathit{last}(p'') \neq ?$

$=$ "$\langle a\rangle t \;=\; \langle a\rangle[p'', p'] \;=\; [\langle a\rangle p'', p'] \;=\; [p, p']$ "

$\exists\, p, p' \cdot \langle a\rangle t = [p, p'] \;\wedge\; PP \xrightarrow{p,p'} 0 \;\wedge\; \mathit{last}(p) \neq ?$

From (14)

$\exists\, P \cdot PP \xrightarrow{!} P \;\wedge\; P \xrightarrow{\langle a\rangle t} 0$

$= \exists\, P, p, p' \cdot p = \langle\rangle \;\wedge\; p' = \langle a\rangle t \wedge PP \xrightarrow{p\langle!\rangle} P \;\wedge\; P \xrightarrow{p'} 0$

$=$ "by using trace rules"

$\exists\, p, p' \cdot \langle a\rangle t = [p.p'] \;\wedge\; p = \langle\rangle \;\wedge\; PP \overset{p\langle!\rangle, p'}{\longrightarrow} 0$

Therefore, for $\langle a\rangle t$ from (13) $\vee$ (14)

$\exists\, p, p' \cdot \langle a\rangle t = [p, p'] \;\wedge\; PP \xrightarrow{p,p'} 0 \;\wedge\; \mathit{last}(p) \neq ?$

$\vee\; \exists\, p, p' \cdot \langle a\rangle t = [p.p'] \;\wedge\; p = \langle\rangle \;\wedge\; PP \overset{p\langle!\rangle, p'}{\longrightarrow} 0$

$=$ "Combining existential quantifications"

$\exists\, p, p' \cdot \langle a\rangle t = [p, p'] \;\wedge\; PP \xrightarrow{p,p'} 0 \;\wedge\; \mathit{last}(p) \neq ?$

Deriving correspondence:

$t \in DT([PP])$

$= [PP] \xrightarrow{t} 0$

$=$ "From Lemma 8"

$\exists\, p, p' \cdot t = [p, p'] \;\wedge\; PP \xrightarrow{p,p'} 0 \;\wedge\; last(p) \neq ?$
$=$ "By trace derivation rule"
$\exists\, p, p' \cdot\; t = [p, p'] \;\wedge\; (p, p) \in DT([PP]) \;\wedge\; last(p) \neq ?$
$=$ "Structural induction"
$\exists\, p, p' \cdot\; t = [p, p'] \;\wedge\; (p, p) \in T([PP]) \;\wedge\; last(p) \neq ?$
$= t \in T([PP])\Box$

A.6 Compensation Pair

Lemma 9
$P \div Q \xrightarrow{(t,t')} 0 \;=\; \exists\, p, q \cdot (t, t') = (p \div q) \;\wedge\; (P \div Q) \xrightarrow{p,q} 0$

Basic step: Case - $\langle\omega\rangle$ and $\langle\omega\rangle$
$P \div Q \xrightarrow{(t,t')} 0$
$= \exists\, R \cdot P \div Q \xrightarrow{\omega} R \;\wedge\; R \xrightarrow{\omega} 0$
$=$ "Using operational rules (R2) and (R3) of compensation pair"

$$P \xrightarrow{\checkmark} 0 \;\wedge\; Q \xrightarrow{\omega} 0 \tag{15}$$
$$\vee\; P \xrightarrow{\omega} 0 \;\wedge\; \omega \neq \checkmark \;\wedge\; SKIP \xrightarrow{\checkmark} 0 \tag{16}$$

From (15)
$P \xrightarrow{\checkmark} 0 \;\wedge\; Q \xrightarrow{\omega} 0$
$= \exists\, p, q \cdot p = \langle\rangle \;\wedge q = \langle\omega\rangle \;\wedge\; P \xrightarrow{p\langle\checkmark\rangle} 0 \;\wedge\; Q \xrightarrow{q} 0$
$=$ "By trace rule"
$\exists\, p, q \cdot (p\langle\checkmark\rangle \div q) = (\omega, \omega) \;\wedge\; P \div Q \xrightarrow{p\langle\checkmark\rangle,q} 0$

From (16)
$P \xrightarrow{\omega} 0 \;\wedge\; \omega \neq \checkmark \;\wedge\; SKIP \xrightarrow{\checkmark} 0$
$= \exists\, p, q \cdot p = \langle\rangle \;\wedge\; q = \langle\checkmark\rangle \;\wedge\; P \div Q \xrightarrow{p\langle\omega\rangle,q} 0$
$=$ "By trace rule"
$\exists\, p, q \cdot (p\langle\omega\rangle \div q) = (\langle\omega\rangle, \langle\checkmark\rangle) \;\wedge\; \omega \neq \checkmark \;\wedge\; P \div Q \xrightarrow{p\langle\omega\rangle,q} 0$

Therefore, from (15) $\vee$ (16)
$\exists\, p, q \cdot (p\langle\checkmark\rangle \div q) = (\omega, \omega) \;\wedge\; P \div Q \xrightarrow{p\langle\checkmark\rangle,q} 0$
$\vee\; \exists\, p, q \cdot (p\langle\omega\rangle \div q) = (\langle\omega\rangle, \langle\checkmark\rangle) \;\wedge\; \omega \neq \checkmark \;\wedge\; P \div Q \xrightarrow{p\langle\omega\rangle,q} 0$
$=$ "Combining and using trace rules"
$\exists\, p, q \cdot (\langle\omega\rangle, \langle\omega\rangle) = (p \div q) \;\wedge\; P \div Q \xrightarrow{p,q} 0$

Inductive step: For inductive case consider either $\langle a\rangle t$ or $\langle a\rangle t'$

$P \div Q \xrightarrow{(t,t')} 0$

$= \exists\, R \cdot P \div Q \xrightarrow{t} R \wedge R \xrightarrow{t'} 0$

"Using operational rules ($R1$) and ($R2$) and applying induction"

$$= \exists\, P' \cdot P \xrightarrow{a} P' \ \wedge\ (P' \div Q) \xrightarrow{(t,t')} 0 \tag{17}$$

$$\vee\ \exists\, Q' \cdot P \xrightarrow{\checkmark} 0 \ \wedge\ Q \xrightarrow{a} Q' \ \wedge\ Q' \xrightarrow{t'} 0 \tag{18}$$

From (17)

$\exists\, P' \cdot P \xrightarrow{a} P' \ \wedge\ (P' \div Q) \xrightarrow{(t,t')} 0$

$=$ "By inductive hypothesis"

$\exists\, P' \cdot P \xrightarrow{a} P' \ \wedge\ \exists\, p', q \cdot (t, t') = (p' \div q) \ \wedge\ (P' \div Q) \xrightarrow{p',q} 0$

$=$ "Combining existential quantification"

$\exists\, p', q \cdot (t, t') = (p' \div q) \ \wedge\ (P \div Q) \xrightarrow{\langle a\rangle p',q} 0$

$= \exists\, p, q \cdot p = \langle a\rangle p' \ \wedge\ (t, t') = (p' \div q) \ \wedge\ (P \div Q) \xrightarrow{p,q} 0$

$=$ "$\langle a\rangle(t, t') \ = \ (\langle a\rangle t, t') \ = \ (\langle a\rangle p') \div q \ = \ (p \div q)$"

$\exists\, p, q \cdot (\langle a\rangle t, t') = (p \div q) \ \wedge\ (P \div Q) \xrightarrow{p,q} 0$

From (18)

$\exists\, Q' \cdot P \xrightarrow{\checkmark} 0 \ \wedge\ Q \xrightarrow{a} Q' \ \wedge\ Q' \xrightarrow{t'} 0$

$= P \xrightarrow{\checkmark} 0 \ \wedge\ Q \xrightarrow{\langle a\rangle t'} 0$

$= \exists\, p, q \cdot p = \langle\rangle \ \wedge\ q = \langle a\rangle t' \ \wedge\ P \xrightarrow{p\langle\checkmark\rangle} 0 \ \wedge\ Q \xrightarrow{q} 0$

$=$ "By trace rule"

$\exists\, p, q, t \cdot t = p\langle\checkmark\rangle \ \wedge\ (p\langle\checkmark\rangle \div q) = (t, \langle a\rangle t') \ \wedge\ (P \div Q) \xrightarrow{p\langle\checkmark\rangle,q} 0$

Therefore, from (17) $\vee$ (18)

$\exists\, p, q \cdot (\langle a\rangle t, t') \ = \ (p \div q) \ \wedge\ (P \div Q) \xrightarrow{p\langle\omega\rangle,q} 0$

$\vee\ \exists\, p, q \cdot (t, \langle a\rangle t') \ = \ (p\langle\checkmark\rangle \div q) \ \wedge\ (P \div Q) \xrightarrow{p\langle\checkmark\rangle,q} 0$

"Combining and using trace rule"

$= \exists\, p, q \cdot (t, t') = (p \div q) \ \wedge\ (P \div Q) \xrightarrow{p,q} 0$

Deriving correspondence:

$(t, t') \ \in \ DT(P \div Q)$

$= (P \div Q) \xrightarrow{(t,t')} 0$

$=$ "From Lemma 9"

$\exists\, p, q \cdot \ (t, t') \ = \ (p \div q) \ \wedge\ (P \div Q) \xrightarrow{p,q} 0$

$=$ "By trace derivation rules"

$\exists\, p, q \cdot \ (t, t') \ = \ (p \div q) \ \wedge\ (p, q) \ \in \ DT(P \div Q)$

$=$ "Structural induction"

$\exists\, p, q \cdot \ (t, t') \ = \ (p \div q) \ \wedge\ (p, q) \ \in \ T(P \div Q)$

$= (t, t') \ \in \ T(P \div Q)\square$