Under consideration for publication in Formal Aspects of Computing

A UTP Semantics for Communicating Processes with Shared Variables and its Formal Encoding in PVS

Ling Shi[1], Yongxin Zhao[2], Yang Liu[3], Jun Sun[4], Jin Song Dong[1], and Shengchao Qin[5]

[1]School of Computing, National University of Singapore, Singapore
[2]Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, China
[3]School of Computer Engineering, Nanyang Technological University, Singapore
[4]Singapore University of Technology and Design, Singapore
[5]Teesside University, UK

1. Monotonicity of CSP# Process Combinators

In this section, we present the detailed proof of the monotonicity of the CSP# process constructs. Given any two processes P and Q such that $P \sqsupseteq Q$, then given any process R, the following auxiliary laws should be satisfied.

Law A.1

$$(P \wedge R) \sqsupseteq (Q \wedge R),\ \textit{provided that}\ P \sqsupseteq Q.$$

Proof:

$$\begin{array}{cll}
 & (P \wedge R) \sqsupseteq (Q \wedge R) & [\sqsupseteq] \\
= & [(P \wedge R) \Rightarrow (Q \wedge R)] & [\textit{propositional calculus}] \\
= & [((P \wedge R) \Rightarrow Q) \wedge ((P \wedge R) \Rightarrow R)] & [\textit{propositional calculus}] \\
= & [((P \Rightarrow Q) \vee (R \Rightarrow Q)) \wedge ((P \Rightarrow R) \vee (R \Rightarrow R))] & [\textit{assumption}] \\
= & [(\textit{true} \vee (R \Rightarrow Q)) \wedge ((P \Rightarrow R) \vee (R \Rightarrow R))] & [\textit{propositional calculus}] \\
= & [\textit{true} \wedge \textit{true}] & [\textit{propositional calculus}] \\
= & \textit{true} & \Box
\end{array}$$

Law A.2

$$(P \vee R) \sqsupseteq (Q \vee R),\ \textit{provided that}\ P \sqsupseteq Q.$$

Correspondence and offprint requests to: Yongxin Zhao, e-mail: yxzhao@sei.ecnu.edu.cn

Proof:

$$
\begin{array}{lll}
 & (P \vee R) \sqsupseteq (Q \vee R) & [\sqsupseteq] \\
= & [(P \vee R) \Rightarrow (Q \vee R)] & [\textit{propositional calculus}] \\
= & [(P \Rightarrow (Q \vee R)) \wedge (R \Rightarrow (Q \vee R))] & [\textit{propositional calculus}] \\
= & [((P \Rightarrow Q) \vee (P \Rightarrow R)) \wedge ((R \Rightarrow Q) \vee (R \Rightarrow R))] & [\textit{assumption}] \\
= & [\mathit{true} \vee (P \Rightarrow R)) \wedge ((R \Rightarrow Q) \vee (R \Rightarrow R))] & [\textit{propositional calculus}] \\
= & [\mathit{true} \wedge \mathit{true}] & [\textit{propositional calculus}] \\
= & \mathit{true} & \square
\end{array}
$$

The CSP# sequential composition construct is monotonic (see **Law A.3** and **Law A.4**).

Law A.3

$(P;\ R) \sqsupseteq (Q;\ R)$, *provided that* $P \sqsupseteq Q$.

Proof:

$$
\begin{array}{lll}
 & (P;\ R) \sqsupseteq (Q;\ R) & [\sqsupseteq] \\
= & \forall\, obs, obs' \bullet ((P;\ R) \Rightarrow (Q;\ R))^{1} & [3.3.2] \\
= & \forall\, obs, obs' \bullet \left(\begin{array}{l} \exists\, obs_0 \bullet (P[obs_0/obs'] \wedge R[obs_0/obs]) \\ \Rightarrow \\ \exists\, obs_0 \bullet (Q[obs_0/obs'] \wedge R[obs_0/obs]) \end{array} \right) & \left[\begin{array}{l} \textit{assumption},\ \sqsupseteq \\ \textit{and Lemma 1} \end{array} \right] \\
= & \mathit{true} & \square
\end{array}
$$

Lemma 1. $\forall\, obs, obs' \bullet (\exists\, m \bullet (P(obs, m) \wedge R(m, obs')) \Rightarrow \exists\, m \bullet (Q(obs, m) \wedge R(m, obs')))$ holds, provided that $\forall\, obs, obs' \bullet (P(obs, obs') \Rightarrow Q(obs, obs'))$.

Proof:

1		$\forall\, obs, obs' \bullet (P(obs, obs') \Rightarrow Q(obs, obs'))$	premise
2	obs_1	$\forall\, obs' \bullet (P(obs_1, obs') \Rightarrow Q(obs_1, obs'))$	$\forall\, obs\ e\ 1$
3	obs'_1	$P(obs_1, obs'_1) \Rightarrow Q(obs_1, obs'_1)$	$\forall\, obs'\ e\ 2$
4		$\exists\, m \bullet (P(obs_1, m) \wedge R(m, obs'_1))$	assumption
5	m_0	$P(obs_1, m_0) \wedge R(m_0, obs'_1)$	$\exists\, m\ e\ 4$
6		$P(obs_1, m_0) \Rightarrow Q(obs_1, m_0)$	$\forall\, obs'\ e\ 2$
7		$P(obs_1, m_0)$	$\wedge e_1\ 5$
8		$Q(obs_1, m_0)$	$\Rightarrow e\ 6, 7$
9		$R(m_0, obs'_1)$	$\wedge e_2\ 5$
10		$Q(obs_1, m_0) \wedge R(m_0, obs'_1)$	$\wedge i\ 8, 9$
11		$\exists\, m \bullet (Q(obs_1, m) \wedge R(m, obs'_1))$	$\exists\, m\ i\ 10$
12		$\exists\, m \bullet (Q(obs_1, m) \wedge R(m, obs'_1))$	$\exists\, m\ 4, 5-11$
13		$\exists\, m \bullet (P(obs_1, m) \wedge R(m, obs'_1)) \Rightarrow$ $\exists\, m \bullet (Q(obs_1, m) \wedge R(m, obs'_1))$	$\Rightarrow i\ 4-12$
14		$\forall\, obs' \bullet (\exists\, m \bullet (P(obs_1, m) \wedge R(m, obs')) \Rightarrow$ $\exists\, m \bullet (Q(obs_1, m) \wedge R(m, obs')))$	$\forall\, obs'\ i\ 3-13$
15		$\forall\, obs, obs' \bullet (\exists\, m \bullet (P(obs, m) \wedge R(m, obs')) \Rightarrow$ $\exists\, m \bullet (Q(obs, m) \wedge R(m, obs')))$	$\forall\, obs\ i\ 2-14$

Law A.4

$(R;\ P) \sqsupseteq (R;\ Q)$, *provided that* $P \sqsupseteq Q$.

[1] The term *obs* represents the set of observational variables *ok*, *wait*, *tr*, and *ref*, as is the case of obs'.

Proof:

$$
\begin{array}{lll}
 & (R;\ P) \sqsupseteq (R;\ Q) & [\sqsupseteq] \\
= & \forall\, obs, obs' \bullet ((R;\ P) \Rightarrow (R;\ Q)) & [3.3.2] \\
= & \forall\, obs, obs' \bullet \left(\begin{array}{l} \exists\, obs_0 \bullet (R[obs_0/obs'] \wedge P[obs_0/obs]) \\ \Rightarrow \\ \exists\, obs_0 \bullet (R[obs_0/obs'] \wedge Q[obs_0/obs]) \end{array} \right) & \left[\begin{array}{l} assumption,\ \sqsupseteq \\ and\ Lemma\ 2 \end{array} \right] \\
= & true & \square
\end{array}
$$

Lemma 2. $\forall\, obs, obs' \bullet (\exists\, m \bullet (R(obs, m) \wedge P(m, obs')) \Rightarrow \exists\, m \bullet (R(obs, m) \wedge P(m, obs')))$ holds, provided that $\forall\, obs, obs' \bullet (P(obs, obs') \Rightarrow Q(obs, obs'))$.

Proof:

1		$\forall\, obs, obs' \bullet (P(obs, obs') \Rightarrow Q(obs, obs'))$	premise
2	obs_1'	$\forall\, obs \bullet (P(obs, obs_1') \Rightarrow Q(obs, obs_1'))$	$\forall\, obs'\ e\ 1$
3	obs_1	$P(obs_1, obs_1') \Rightarrow Q(obs_1, obs_1')$	$\forall\, obs\ e\ 2$
4		$\exists\, m \bullet (R(obs_1, m) \wedge P(m, obs_1'))$	assumption
5	m_0	$R(obs_1, m_0) \wedge P(m_0, obs_1')$	$\exists\, m\ e\ 4$
6		$P(m_0, obs_1') \Rightarrow Q(m_0, obs_1')$	$\forall\, obs\ e\ 2$
7		$P(m_0, obs_1')$	$\wedge e_2\ 5$
8		$Q(m_0, obs_1')$	$\Rightarrow e\ 6, 7$
9		$R(obs_1, m_0)$	$\wedge e_1\ 5$
10		$R(obs_1, m_0) \wedge Q(m_0, obs_1')$	$\wedge i\ 8, 9$
11		$\exists\, m \bullet (R(obs_1, m) \wedge Q(m, obs_1'))$	$\exists\, m\ i\ 10$
12		$\exists\, m \bullet (R(obs_1, m) \wedge Q(m, obs_1'))$	$\exists\, m\ 4, 5-11$
13		$\exists\, m \bullet (R(obs_1, m) \wedge P(m, obs_1')) \Rightarrow$ $\exists\, m \bullet (R(obs_1, m) \wedge Q(m, obs_1'))$	$\Rightarrow i\ 4-12$
14		$\forall\, obs \bullet (\exists\, m \bullet (R(obs, m) \wedge P(m, obs_1')) \Rightarrow$ $\exists\, m \bullet (R(obs, m) \wedge Q(m, obs_1')))$	$\forall\, obs\ i\ 3-13$
15		$\forall\, obs, obs' \bullet (\exists\, m \bullet (R(obs, m) \wedge P(m, obs')) \Rightarrow$ $\exists\, m \bullet (R(obs, m) \wedge Q(m, obs')))$	$\forall\, obs'\ i\ 2-14$

Synchronous output/input is monotonic (see **Law A.5** and **Law A.6**).

Law A.5

$$(ch!exp \rightarrow P) \sqsupseteq (ch!exp \rightarrow Q),\ provided\ that\ P \sqsupseteq Q.$$

Proof:

$$
\begin{array}{lll}
 & (ch!exp \rightarrow P) & [3.3.4] \\
= & \mathbf{H}\left(ok' \wedge \left(\begin{array}{l} ch? \notin ref' \wedge tr' = tr \\ \lhd wait' \rhd \\ \exists\, s \in \mathsf{S} \bullet tr' = tr \frown \langle (s, ch!\mathcal{A}[\![exp]\!](s)) \rangle \end{array} \right) \right);\ P & \left[\begin{array}{l} assumption \\ and\ A.4 \end{array} \right] \\
\sqsupseteq & \mathbf{H}\left(ok' \wedge \left(\begin{array}{l} ch? \notin ref' \wedge tr' = tr \\ \lhd wait' \rhd \\ \exists\, s \in \mathsf{S} \bullet tr' = tr \frown \langle (s, ch!\mathcal{A}[\![exp]\!](s)) \rangle \end{array} \right) \right);\ Q & [3.3.4] \\
= & ch!exp \rightarrow Q & \square
\end{array}
$$

Law A.6

$$(ch?m \rightarrow P(m)) \sqsupseteq (ch?m \rightarrow Q(m)),\ provided\ that\ \forall\, m \in \mathsf{T} \bullet P(m) \sqsupseteq Q(m).$$

Proof:

$$
\begin{array}{lll}
 & ch?m \rightarrow P(m) & [3.3.4] \\
= & \exists\, v \in \mathsf{T} \bullet \left(\mathbf{H} \left(ok' \wedge \left(\begin{array}{l} ch! \notin ref' \wedge tr' = tr \\ \lhd wait' \rhd \\ tr' = tr \frown \langle (s, ch?v) \rangle \end{array} \right) \right) ;\ P(v) \right) & \left[\begin{array}{l} assumption,\ A.4, \\ and\ predicate \\ calculus \end{array} \right] \\
\sqsupseteq & \exists\, v \in \mathsf{T} \bullet \left(\mathbf{H} \left(ok' \wedge \left(\begin{array}{l} ch! \notin ref' \wedge tr' = tr \\ \lhd wait' \rhd \\ tr' = tr \frown \langle (s, ch?v) \rangle \end{array} \right) \right) ;\ Q(v) \right) & [3.3.4] \\
= & ch?m \rightarrow Q(m) & \Box
\end{array}
$$

The CSP# data operation prefixing construct is monotonic (see **Law A.7**).

Law A.7

$$(\{prog\} \rightarrow P) \sqsupseteq (\{prog\} \rightarrow Q),\ provided\ that\ P \sqsupseteq Q.$$

Proof:

$$
\begin{array}{lll}
 & \{prog\} \rightarrow P & [3.3.5] \\
= & \mathbf{H} \left(ok' \wedge \left(\begin{array}{l} \exists\, s \in \mathsf{S} \bullet (tr' = tr \frown \langle (s, \perp) \rangle \wedge (s, \perp) \in \mathcal{C}[\![prog]\!]) \\ \lhd wait' \rhd \\ \exists\, s, s' \in \mathsf{S} \bullet (tr' = tr \frown \langle (s, s') \rangle \wedge (s, s') \in \mathcal{C}[\![prog]\!] \\ \wedge (s, \perp) \notin \mathcal{C}[\![prog]\!]) \end{array} \right) \right) ;\ P & \left[\begin{array}{l} assumption \\ and\ A.4 \end{array} \right] \\
\sqsupseteq & \mathbf{H} \left(ok' \wedge \left(\begin{array}{l} \exists\, s \in \mathsf{S} \bullet (tr' = tr \frown \langle (s, \perp) \rangle \wedge (s, \perp) \in \mathcal{C}[\![prog]\!]) \\ \lhd wait' \rhd \\ \exists\, s, s' \in \mathsf{S} \bullet (tr' = tr \frown \langle (s, s') \rangle \wedge (s, s') \in \mathcal{C}[\![prog]\!] \\ \wedge (s, \perp) \notin \mathcal{C}[\![prog]\!]) \end{array} \right) \right) ;\ Q & [3.3.5] \\
= & \{prog\} \rightarrow Q & \Box
\end{array}
$$

The CSP# state guard is monotonic (see **Law A.8**).

Law A.8

$$[b]P \sqsupseteq [b]Q,\ provided\ that\ P \sqsupseteq Q.$$

Proof:

$$
\begin{array}{lll}
 & [b]P \sqsupseteq [b]Q & [3.3.7\ and\ \sqsupseteq] \\
= & \left[\begin{array}{l} \widehat{P} \lhd \mathcal{B}(b)(\pi_1(head(tr' - tr))) = true \wedge tr < tr') \rhd Stop \\ \Rightarrow \\ \widehat{Q} \lhd \mathcal{B}(b)(\pi_1(head(tr' - tr))) = true \wedge tr < tr') \rhd Stop \end{array} \right] & \left[\begin{array}{l} predicate \\ calculus \end{array} \right]
\end{array}
$$

$$
= \left[\begin{array}{l}
\left(\begin{array}{l}
\left(\begin{array}{l}
(\widehat{P} \wedge \mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr') \\
\Rightarrow \\
(\widehat{Q} \wedge \mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr')
\end{array}\right) \\
\quad \vee \\
\left(\begin{array}{l}
(\widehat{P} \wedge \mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr') \\
\Rightarrow \\
(Stop \wedge \neg(\mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr'))
\end{array}\right)
\end{array}\right) \\
\wedge \\
\left(\begin{array}{l}
\left(\begin{array}{l}
(Stop \wedge \neg(\mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr')) \\
\Rightarrow \\
(\widehat{Q} \wedge \mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr')
\end{array}\right) \\
\quad \vee \\
\left(\begin{array}{l}
(Stop \wedge \neg(\mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr')) \\
\Rightarrow \\
(Stop \wedge \neg(\mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr'))
\end{array}\right)
\end{array}\right)
\end{array}\right]
\qquad \left[\begin{array}{l} \mathit{predicate} \\ \mathit{calculus} \end{array}\right]
$$

$$
= \left[\begin{array}{l}
\left(\begin{array}{l}
(\widehat{P} \Rightarrow \widehat{Q}) \\
\quad \vee \\
((\mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr') \Rightarrow \widehat{Q}) \\
\quad \vee \\
\left(\begin{array}{l}
(\widehat{P} \wedge \mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr') \\
\Rightarrow \\
(Stop \wedge \neg(\mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr'))
\end{array}\right)
\end{array}\right) \\
\wedge \\
\left(\begin{array}{l}
\left(\begin{array}{l}
(Stop \wedge \neg(\mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr')) \\
\Rightarrow \\
(\widehat{Q} \wedge \mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr')
\end{array}\right) \\
\quad \vee \\
true
\end{array}\right)
\end{array}\right]
\qquad \left[\begin{array}{l} \mathit{predicate} \\ \mathit{calculus} \end{array}\right]
$$

$$
= \left[\begin{array}{l}
\left(\begin{array}{l}
\left(\begin{array}{l}
(P \wedge tr < tr' \vee P(tr,tr) \wedge \exists\, s \in \mathsf{S} \cdot tr' - tr = \langle (s,s) \rangle) \\
\Rightarrow \\
(Q \wedge tr < tr' \vee Q(tr,tr) \wedge \exists\, s \in \mathsf{S} \cdot tr' - tr = \langle (s,s) \rangle)
\end{array}\right) \vee \\
((\mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr') \Rightarrow \widehat{Q}) \\
\quad \vee \\
\left(\begin{array}{l}
(\widehat{P} \wedge \mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr') \\
\Rightarrow \\
(Stop \wedge \neg(\mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr'))
\end{array}\right)
\end{array}\right) \\
\wedge \\
true
\end{array}\right]
\qquad \left[\begin{array}{l} \mathit{predicate} \\ \mathit{calculus} \end{array}\right]
$$

$$
= \left[\left(\begin{array}{l}
\left(\begin{array}{l}
((P \wedge tr < tr') \Rightarrow \\
\quad (Q \wedge tr < tr' \vee Q(tr,tr) \wedge \exists\, s \in \mathsf{S} \cdot tr' - tr = \langle (s,s) \rangle)) \\
\wedge \\
((P(tr,tr)\, \exists\, s \in \mathsf{S} \cdot tr' - tr = \langle (s,s) \rangle) \Rightarrow \\
\quad (Q \wedge tr < tr' \vee Q(tr,tr) \wedge \exists\, s \in \mathsf{S} \cdot tr' - tr = \langle (s,s) \rangle))
\end{array}\right) \vee \\
((\mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr') \Rightarrow \widehat{Q}) \\
\quad \vee \\
\left(\begin{array}{l}
(\widehat{P} \wedge \mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr') \\
\Rightarrow \\
(Stop \wedge \neg(\mathcal{B}(b)(\pi_1(head(tr'-tr))) = true \wedge tr < tr'))
\end{array}\right)
\end{array}\right)\right]
\qquad \left[\begin{array}{l} \mathit{assumption} \\ \mathit{and\ predicate} \\ \mathit{calculus} \end{array}\right]
$$

$$
= \quad true \wedge true \qquad \left[\begin{array}{l} \mathit{propositional} \\ \mathit{calculus} \end{array}\right]
$$

$$
= \quad true \qquad \Box
$$

$$
\begin{array}{ll}
\mathit{where} & \widehat{P} \mathrel{\widehat{=}} P \wedge tr < tr' \vee P(tr,tr) \wedge \exists\, s \in \mathsf{S} \cdot tr' - tr = \langle (s,s) \rangle \\
& \widehat{Q} \mathrel{\widehat{=}} Q \wedge tr < tr' \vee Q(tr,tr) \wedge \exists\, s \in \mathsf{S} \cdot tr' - tr = \langle (s,s) \rangle
\end{array}
$$

The CSP# parallel composition is monotonic (see **Law A.9** and **Law A.10**).

Law A.9

$$P \parallel_{(X_1,X_2)} R \sqsupseteq Q \parallel_{(X_1,X_2)} R$$

provided that $P \sqsupseteq Q$.
Proof:

$$\begin{array}{llr}
 & P \sqsupseteq Q & [\sqsupseteq] \\
= & [P \Rightarrow Q] & [\mathit{predicate\ calculus}] \\
= & [(\exists\, 0.obs \bullet P[0.obs/obs']) \Rightarrow (\exists\, 0.obs \bullet Q[0.obs/obs'])]^2 & [\sqsupseteq] \\
= & (\exists\, 0.obs \bullet P[0.obs/obs']) \sqsupseteq (\exists\, 0.obs \bullet Q[0.obs/obs']) & [\mathit{Law\ A}.1] \\
= & \begin{array}{l}(\exists\, 0.obs, 1.obs \bullet (P[0.obs/obs'] \wedge R[1.obs/obs'])) \\ \sqsupseteq \\ (\exists\, 0.obs, 1.obs \bullet (Q[0.obs/obs'] \wedge R[1.obs/obs']))\end{array} & [\mathit{Law\ A}.3] \\
\Rightarrow & \begin{array}{l}(\exists\, 0.obs, 1.obs \bullet (P[0.obs/obs'] \wedge R[1.obs/obs'] \wedge M(X_1, X_2))) \\ \sqsupseteq \\ (\exists\, 0.obs, 1.obs \bullet (Q[0.obs/obs'] \wedge R[1.obs/obs'] \wedge M(X_1, X_2)))\end{array} & [\mathit{predicate\ calculus}] \\
= & P \parallel_{(X_1,X_2)} R \sqsupseteq Q \parallel_{(X_1,X_2)} R & \square
\end{array}$$

Law A.10

$$R \parallel_{(X_1,X_2)} P \sqsupseteq R \parallel_{(X_1,X_2)} Q,$$

provided that $P \sqsupseteq Q$.
Proof:

$$\begin{array}{llr}
 & P \sqsupseteq Q & [\sqsupseteq] \\
= & [P \Rightarrow Q] & [\mathit{predicate\ calculus}] \\
= & [(\exists\, 1.obs \bullet P[1.obs/obs']) \Rightarrow (\exists\, 1.obs \bullet Q[1.obs/obs'])]^3 & [\sqsupseteq] \\
= & (\exists\, 1.obs \bullet P[1.obs/obs']) \sqsupseteq (\exists\, 1.obs \bullet Q[1.obs/obs']) & [A.1] \\
= & \begin{array}{l}(\exists\, 0.obs, 1.obs \bullet (P[1.obs/obs'] \wedge R[0.obs/obs'])) \\ \sqsupseteq \\ (\exists\, 0.obs, 1.obs \bullet (Q[1.obs/obs'] \wedge R[0.obs/obs']))\end{array} & [\mathit{predicate\ calculus}] \\
= & \begin{array}{l}(\exists\, 0.obs, 1.obs \bullet (R[0.obs/obs'] \wedge P[1.obs/obs'])) \\ \sqsupseteq \\ (\exists\, 0.obs, 1.obs \bullet (R[0.obs/obs'] \wedge Q[1.obs/obs']))\end{array} & [\mathit{predicatecalculus}] \\
\Rightarrow & \begin{array}{l}(\exists\, 0.obs, 1.obs \bullet (R[0.obs/obs'] \wedge P[1.obs/obs'] \wedge M(X_1, X_2))) \\ \sqsupseteq \\ (\exists\, 0.obs, 1.obs \bullet (R[0.obs/obs'] \wedge Q[1.obs/obs'] \wedge M(X_1, X_2)))\end{array} & [3.3.8] \\
= & R \parallel_{(X_1,X_2)} P \sqsupseteq R \parallel_{(X_1,X_2)} Q & \square
\end{array}$$

Since the semantics of other CSP# processes (i.e., event prefixing, external/internal choice and recursion) is the same as that of CSP, the proof is ommited here.

[2] The term *obs* represents the set of observational variables *ok*, *wait*, *tr*, and *ref*, as is the case of obs'.
[3] The term *obs* represents the set of observational variables *ok*, *wait*, *tr*, and *ref*, as is the case of obs'. The term in the following sections represent the same meaning.

2. Proof of Algebraic Laws

In this section, we present the proofs of laws in Section 4.
Law guard - 1

$$[b_1]([b_2]P) \;=\; [b_1 \wedge b_2]P$$

Proof:

$$\begin{array}{lll}
 & [b_1]([b_2]P) & [3.3.7] \\
= & \begin{array}{l}(\widehat{P} \lhd (\mathcal{B}(b_2)(\pi_1(head(\widehat{tr' - tr}))) = true \wedge tr < tr') \rhd Stop) \\ \lhd(\mathcal{B}(b_1)(\pi_1(head(tr' - tr))) = true \wedge tr < tr') \rhd Stop\end{array} & [predicate\ calculus] \\
= & \begin{array}{l}\left(\begin{array}{l}\widehat{P} \wedge \mathcal{B}(b_2)(\pi_1(head(tr' - tr))) = true \wedge \\ tr < tr' \wedge \\ \mathcal{B}(b_1)(\pi_1(head(tr' - tr))) = true\end{array}\right) \\ \vee \\ \left(\begin{array}{l}Stop \wedge \neg(\mathcal{B}(b_2)(\pi_1(head(tr' - tr))) = true \wedge \\ \qquad tr < tr' \wedge \\ \qquad \mathcal{B}(b_1)(\pi_1(head(tr' - tr))) = true)\end{array}\right)\end{array} & [Def.\ 2] \\
= & \begin{array}{l}(\widehat{P} \wedge \mathcal{B}(b_2 \wedge b_1)(\pi_1(head(tr' - tr))) = true \wedge tr < tr') \\ \vee \\ (Stop \wedge \neg(\mathcal{B}(b_2 \wedge b_1)(\pi_1(head(tr' - tr))) = true \wedge tr < tr'))\end{array} & [3.3.7] \\
= & [b_1 \wedge b_2]P & \Box \\
where & \widehat{P} \mathrel{\widehat=} P \wedge tr < tr' \vee P(tr, tr) \wedge \exists\, s \in \mathsf{S} \cdot tr' - tr = \langle (s, s) \rangle &
\end{array}$$

Law guard - 2

$$[b](P_1 \ \mathsf{op}\ P_2) \;=\; [b]P_1 \ \mathsf{op}\ [b]P_2 \qquad \text{where, } \mathsf{op} \in \{\|, \Box, \sqcap\}$$

Proof: The guard b_1 constrains that the pre-state of initial observation of composition process should satisfies the condition, since the pre-state of the initial observation of the composition process can be from either process P_1 or P_2 (see Section 3.3.6, 3.3.8), so the condition should be satisfied by the initial observation of both processes. □

Law par - 1

$$P_1 \|_{(X_1,X_2)} P_2 = P_2 \|_{(X_1,X_2)} P_1$$

Proof:

$$\begin{array}{lll}
 & P_1 \|_{(X_1,X_2)} P_2 & [3.3.8] \\
= & \exists\, 0.ons, 1.obs \bullet (P_1[0.obs/obs'] \wedge P_2[1.obs/obs'] \wedge M(X_1, X_2)) & \left[\begin{array}{l}symmetry\ of\ M(X_1, X_2)\ and \\ predicate\ calculus\end{array}\right] \\
= & \exists\, 0.ons, 1.obs \bullet (P_2[0.obs/obs'] \wedge P_1[1.obs/obs'] \wedge M(X_1, X_2)) & [3.3.8] \\
= & P_2 \|_{(X_1,X_2)} P_1 & \Box
\end{array}$$

Law par - 2

$$(P_1 \|_{(X_1,X_2)} P_2) \|_{(X_1,X_2)} P_3 \;=\; P_1 \|_{(X_1,X_2)} (P_2 \|_{(X_1,X_2)} P_3),$$

Proof:[4]

$$
\begin{array}{lll}
& (P_1 \parallel_{(X_1,X_2)} P_2) \parallel_{(X_1,X_2)} P_3 & [3.3.8] \\
= & \left(\begin{array}{l} (P_1[0.obs/obs'] \wedge P_2[1.obs/obs'] \wedge M(X_1,X_2))[0.obs/obs'] \\ \wedge \\ P_3[1.obs/obs'] \end{array} \right) \wedge M(X_1,X_2) & \left[\begin{array}{l} associativity \\ of\ M(X_1,X_2), \\ predicate\ calculus \end{array} \right] \\
= & \left(\begin{array}{l} P_1[0.obs/obs'] \\ \wedge \\ (P_2[0.obs/obs'] \wedge P_3[1.obs/obs'] \wedge M(X_1,X_2))[1.obs/obs'] \end{array} \right) \wedge M(X_1,X_2) & [3.3.8] \\
= & P_1 \parallel_{(X_1,X_2)} (P_2 \parallel_{(X_1,X_2)} P_3) & \square
\end{array}
$$

Law par - 3

$$Skip \parallel_{(X_1,X_2)} P = P = P \parallel_{(X_1,X_2)} Skip,$$

given that set X_1 is an empty set ($\varnothing$) and X_2 is a set of channel outputs and inputs.
Proof:

$$
\begin{array}{lll}
& Skip \parallel_{(X_1,X_2)} P & [\mathbf{par-1}] \\
= & P \parallel_{(X_1,X_2)} Skip & \square
\end{array}
$$

$$
\begin{array}{llll}
& P \parallel_{(X_1,X_2)} Skip & & [3.3.8] \\
= & (P[0.obs/obs']) \wedge (Skip[1.obs/obs']) \wedge M(X_1,X_2) & & [3.3.1] \\
= & (P[0.obs/obs']) \wedge (\mathbf{H}(\exists\, ref \bullet II)[1.obs/obs']) \wedge M(X_1,X_2) & & [\mathbf{H}] \\
= & \left(\begin{array}{l} ((P[0.obs/obs']) \wedge \\ \left(\begin{array}{l} (wait \wedge \neg ok \wedge tr \le tr') \vee \\ (wait \wedge ok' \wedge tr' = tr \wedge wait' = wait \wedge ref' = ref) \vee \\ (\neg wait \wedge \neg ok \wedge tr \le tr') \vee \\ (\neg wait \wedge \exists\, ref \bullet (ok' \wedge tr' = tr \wedge wait' = wait \\ \quad \wedge\, ref' = ref)) \end{array} \right) [1.obs/obs'] \end{array} \right) \wedge M(X_1,X_2) & & \left[\begin{array}{l} propos- \\ itional \\ calculus \end{array} \right] \\
= & \left(\begin{array}{l} ((P[0.obs/obs']) \wedge \\ \left(\begin{array}{l} (\neg ok \wedge tr \le tr') \vee \\ (wait \wedge ok' \wedge tr' = tr \wedge wait' = wait \wedge ref' = ref) \vee \\ (\neg wait \wedge ok' \wedge tr' = tr \wedge wait' = wait) \end{array} \right) [1.obs/obs'] \end{array} \right) \wedge M(X_1,X_2) & & \left[\begin{array}{l} P\ is \\ \mathbf{CSP1} \end{array} \right] \\
= & \left(\begin{array}{l} ((\mathbf{CSP1}(P)[0.obs/obs']) \wedge \\ \left(\begin{array}{l} (\neg ok \wedge tr \le tr') \vee \\ (wait \wedge ok' \wedge tr' = tr \wedge wait' = wait \wedge ref' = ref) \vee \\ (\neg wait \wedge ok' \wedge tr' = tr \wedge wait' = wait) \end{array} \right) [1.obs/obs'] \end{array} \right) \wedge M(X_1,X_2) & & [\mathbf{CSP1}] \\
= & \left(\begin{array}{l} ((P \vee \neg ok \wedge tr \le tr')[0.obs/obs']) \wedge \\ \left(\begin{array}{l} (\neg ok \wedge tr \le tr') \vee \\ (wait \wedge ok' \wedge tr' = tr \wedge wait' = wait \wedge ref' = ref) \vee \\ (\neg wait \wedge ok' \wedge tr' = tr \wedge wait' = wait) \end{array} \right) [1.obs/obs'] \end{array} \right) \wedge M(X_1,X_2) & & [P\ is\ \mathbf{R3}] \\
= & \left(\begin{array}{l} ((\mathbf{R3}(P) \vee \neg ok \wedge tr \le tr')[0.obs/obs']) \wedge \\ \left(\begin{array}{l} (\neg ok \wedge tr \le tr') \vee \\ (wait \wedge ok' \wedge tr' = tr \wedge wait' = wait \wedge ref' = ref) \vee \\ (\neg wait \wedge ok' \wedge tr' = tr \wedge wait' = wait) \end{array} \right) [1.obs/obs'] \end{array} \right) \wedge M(X_1,X_2) & & [\mathbf{R3}]
\end{array}
$$

[4] We will omit the extensional qualification of $0.obs$ and $1.obs$ in the sequent proofs.

$$
= \left(\begin{array}{l}
\left(\begin{array}{l} \mathit{wait} \wedge \mathit{II} \vee \\ \neg \mathit{wait} \wedge P \vee \\ \neg \mathit{ok} \wedge \mathit{tr} \leq \mathit{tr}' \end{array} \right) [0.\mathit{obs}/\mathit{obs}'])\\
\wedge \\
\left(\begin{array}{l} (\neg \mathit{ok} \wedge \mathit{tr} \leq \mathit{tr}') \vee \\ (\mathit{wait} \wedge \mathit{ok}' \wedge \mathit{tr}' = \mathit{tr} \wedge \mathit{wait}' = \mathit{wait} \wedge \mathit{ref}' = \mathit{ref}) \vee \\ (\neg \mathit{wait} \wedge \mathit{ok}' \wedge \mathit{tr}' = \mathit{tr} \wedge \mathit{wait}' = \mathit{wait}) \end{array} \right) [1.\mathit{obs}/\mathit{obs}']
\end{array} \right) \wedge M(X_1, X_2) \qquad \left[\begin{array}{l} \mathit{II\ and} \\ \mathit{props\text{-}} \\ \mathit{itional} \\ \mathit{calculus} \end{array} \right]
$$

$$
= \left(\begin{array}{l}
\left(\begin{array}{l} (\neg \mathit{ok} \wedge \mathit{tr} \leq \mathit{tr}') \vee \\ (\mathit{wait} \wedge \mathit{ok}' \wedge \mathit{tr}' = \mathit{tr} \wedge \mathit{wait}' = \mathit{wait} \wedge \mathit{ref}' = \mathit{ref}) \vee \\ (\neg \mathit{wait} \wedge P) \end{array} \right) [0.\mathit{obs}/\mathit{obs}'])\\
\wedge \\
\left(\begin{array}{l} (\neg \mathit{ok} \wedge \mathit{tr} \leq \mathit{tr}') \vee \\ (\mathit{wait} \wedge \mathit{ok}' \wedge \mathit{tr}' = \mathit{tr} \wedge \mathit{wait}' = \mathit{wait} \wedge \mathit{ref}' = \mathit{ref}) \vee \\ (\neg \mathit{wait} \wedge \mathit{ok}' \wedge \mathit{tr}' = \mathit{tr} \wedge \mathit{wait}' = \mathit{wait}) \end{array} \right) [1.\mathit{obs}/\mathit{obs}']
\end{array} \right) \wedge M(X_1, X_2) \qquad \left[\begin{array}{l} \mathit{predicate} \\ \mathit{calculus} \end{array} \right]
$$

$$
= \left(\begin{array}{l}
\left(\begin{array}{l} (\neg \mathit{ok} \wedge \mathit{tr} \leq 0.\mathit{tr}) \vee \\ (\mathit{wait} \wedge 0.\mathit{ok} \wedge 0.\mathit{tr} = \mathit{tr} \wedge 0.\mathit{wait} = \mathit{wait} \wedge 0.\mathit{ref} = \mathit{ref}) \vee \\ (\neg \mathit{wait} \wedge P[0.\mathit{obs}/\mathit{obs}']) \end{array} \right)\\
\wedge \\
\left(\begin{array}{l} (\neg \mathit{ok} \wedge \mathit{tr} \leq 1.\mathit{tr}) \vee \\ (\mathit{wait} \wedge 1.\mathit{ok} \wedge 1.\mathit{tr} = \mathit{tr} \wedge 1.\mathit{wait} = \mathit{wait} \wedge 1.\mathit{ref} = \mathit{ref}) \vee \\ (\neg \mathit{wait} \wedge 1.\mathit{ok} \wedge 1.\mathit{tr} = \mathit{tr} \wedge 1.\mathit{wait} = \mathit{wait}) \end{array} \right)
\end{array} \right) \wedge M(X_1, X_2) \qquad \left[\begin{array}{l} \mathit{propos\text{-}} \\ \mathit{itional} \\ \mathit{calculus} \end{array} \right]
$$

$$
= \left(\begin{array}{l}
(\neg \mathit{ok} \wedge \mathit{tr} \leq 0.\mathit{tr} \wedge \mathit{tr} \leq 1.\mathit{tr}) \vee \\
(\neg \mathit{ok} \wedge \mathit{tr} \leq 0.\mathit{tr} \wedge \mathit{wait} \wedge 1.\mathit{ok} \\
\quad \wedge 1.\mathit{tr} = \mathit{tr} \wedge 1.\mathit{wait} = \mathit{wait} \wedge 1.\mathit{ref} = \mathit{ref}) \vee \\
(\neg \mathit{ok} \wedge \mathit{tr} \leq 0.\mathit{tr} \wedge \neg \mathit{wait} \wedge 1.\mathit{ok} \\
\quad \wedge 1.\mathit{tr} = \mathit{tr} \wedge 1.\mathit{wait} = \mathit{wait}) \vee \\
(\neg \mathit{ok} \wedge \mathit{tr} \leq 1.\mathit{tr} \wedge \mathit{wait} \wedge 0.\mathit{ok} \\
\quad \wedge 0.\mathit{tr} = \mathit{tr} \wedge 0.\mathit{wait} = \mathit{wait} \wedge 0.\mathit{ref} = \mathit{ref}) \vee \\
(\mathit{wait} \wedge 0.\mathit{ok} \wedge 1.\mathit{ok} \wedge 0.\mathit{tr} = 1.\mathit{tr} = \mathit{tr} \\
\quad \wedge 0.\mathit{wait} = 1.\mathit{wait} = \mathit{wait} \wedge 0.\mathit{ref} = 1.\mathit{ref} = \mathit{ref}) \vee \\
(\neg \mathit{ok} \wedge \mathit{tr} \leq 1.\mathit{tr} \wedge \neg \mathit{wait} \wedge (P[0.\mathit{obs}/\mathit{obs}'])) \vee \\
(\neg \mathit{wait} \wedge 1.\mathit{ok} \wedge 1.\mathit{tr} = \mathit{tr} \wedge 1.\mathit{wait} = \mathit{wait} \wedge P[0.\mathit{obs}/\mathit{obs}'])
\end{array} \right) \wedge M(X_1, X_2) \qquad \left[\begin{array}{l} 3.3.8 \\ M(X_1, X_2) \end{array} \right]
$$

$$
= \begin{array}{l}
\left(\begin{array}{l}
(\neg \mathit{ok} \wedge \mathit{tr} \leq 0.\mathit{tr} \wedge \mathit{tr} \leq 1.\mathit{tr}) \vee \\
(\neg \mathit{ok} \wedge \mathit{tr} \leq 0.\mathit{tr} \wedge \mathit{wait} \wedge 1.\mathit{ok} \\
\quad \wedge 1.\mathit{tr} = \mathit{tr} \wedge 1.\mathit{wait} = \mathit{wait} \wedge 1.\mathit{ref} = \mathit{ref}) \vee \\
(\neg \mathit{ok} \wedge \mathit{tr} \leq 0.\mathit{tr} \wedge \neg \mathit{wait} \wedge 1.\mathit{ok} \\
\quad \wedge 1.\mathit{tr} = \mathit{tr} \wedge 1.\mathit{wait} = \mathit{wait}) \vee \\
(\neg \mathit{ok} \wedge \mathit{tr} \leq 1.\mathit{tr} \wedge \mathit{wait} \wedge 0.\mathit{ok} \\
\quad \wedge 0.\mathit{tr} = \mathit{tr} \wedge 0.\mathit{wait} = \mathit{wait} \wedge 0.\mathit{ref} = \mathit{ref}) \vee \\
(\mathit{wait} \wedge 0.\mathit{ok} \wedge 1.\mathit{ok} \wedge 0.\mathit{tr} = 1.\mathit{tr} = \mathit{tr} \\
\quad \wedge 0.\mathit{wait} = 1.\mathit{wait} = \mathit{wait} \wedge 0.\mathit{ref} = 1.\mathit{ref} = \mathit{ref}) \vee \\
(\neg \mathit{ok} \wedge \mathit{tr} \leq 1.\mathit{tr} \wedge \neg \mathit{wait} \wedge (P[0.\mathit{obs}/\mathit{obs}'])) \vee \\
(\neg \mathit{wait} \wedge 1.\mathit{ok} \wedge 1.\mathit{tr} = \mathit{tr} \wedge 1.\mathit{wait} = \mathit{wait} \wedge (P[0.\mathit{obs}/\mathit{obs}']))
\end{array} \right) \wedge \\
\left(\begin{array}{l}
(\mathit{ok}' = 0.\mathit{ok} \wedge 1.\mathit{ok}) \wedge \\
(\mathit{wait}' = 0.\mathit{wait} \vee 1.\mathit{wait}) \wedge \\
(\mathit{ref}' = ((0.\mathit{ref} \cup 1.\mathit{ref}) \cap X_1) \cup ((0.\mathit{ref} \cap 1.\mathit{ref}) - X_1) \\
\qquad\qquad \cup (0.\mathit{ref} \cap 1.\mathit{ref} \cap X_2)) \wedge \\
(\mathit{tr}' - \mathit{tr} \in (0.\mathit{tr} - \mathit{tr} \parallel_X 1.\mathit{tr} - \mathit{tr}))
\end{array} \right)
\end{array} \qquad \left[\begin{array}{l} \mathit{predicate} \\ \mathit{calculus} \end{array} \right]
$$

$$= \begin{pmatrix} (\neg ok \wedge tr \leq tr') \vee \\ (\neg ok \wedge wait \wedge tr \leq tr' \wedge ok' = 0.ok \wedge \Psi_1) \vee \\ (\neg ok \wedge \neg wait \wedge tr \leq tr' \wedge wait' = 0.wait) \vee \\ (\neg ok \wedge wait \wedge tr \leq tr' \wedge ok' = 1.ok \wedge \Psi_2) \vee \\ (wait \wedge ok' \wedge tr' = tr \wedge wait' = wait \wedge ref' = ref) \vee \\ (\neg ok \wedge \neg wait \wedge tr \leq tr' \wedge \Psi_3) \vee \\ (\neg wait \wedge ok' = 0.ok \wedge tr' = 0.tr \wedge wait' = 0.wait \\ \quad \wedge ref' = 0.ref \wedge (P[0.obs/obs'])) \end{pmatrix}^5 \qquad \begin{bmatrix} predicate \\ calculus \end{bmatrix}$$

$$= \begin{pmatrix} (\neg ok \wedge tr \leq tr') \vee \\ (wait \wedge ok' \wedge tr' = tr \wedge wait' = wait \wedge ref' = ref) \vee \\ (\neg wait \wedge P) \end{pmatrix} \qquad [II]$$

$$= \begin{pmatrix} (wait \wedge II) \vee \\ (\neg wait \wedge P) \vee \\ (\neg ok \wedge tr \leq tr') \end{pmatrix} \qquad [\mathbf{R3}]$$

$$= \begin{pmatrix} \mathbf{R3}(P) \vee \\ (\neg ok \wedge tr \leq tr') \end{pmatrix} \qquad [P \ is \ \mathbf{R3}]$$

$$= (P \ \vee \ \neg ok \wedge tr \leq tr') \qquad [\mathbf{CSP1}]$$

$$= \mathbf{CSP1}(P) \qquad [P \ is \ \mathbf{CSP1}]$$

$$= P \qquad \square$$

3. Sequential composition is H healthy

CSP# sequential composition $P;\ Q$ is **H** healthy given that processes P and Q are **H** healthy.

Law seq_H

$\mathbf{H}(P;\ Q) = P;\ Q),$ *provided that* $\mathbf{H}(P) = P$ *and* $\mathbf{H}(Q) = Q$.

Proof:

$$\begin{array}{lll} & \mathbf{H}(P;\ Q) & [H] \\ = & \mathbf{CSP2}(\mathbf{CSP1}(\mathbf{R3}(\mathbf{R2}(\mathbf{R1}(P;\ Q))))) & [\mathbf{seq\_R1}] \\ = & \mathbf{CSP2}(\mathbf{CSP1}(\mathbf{R3}(\mathbf{R2}(P;\ Q)))) & [\mathbf{seq\_R2}] \\ = & \mathbf{CSP2}(\mathbf{CSP1}(\mathbf{R3}(P;\ Q))) & [\mathbf{seq\_R3}] \\ = & \mathbf{CSP2}(\mathbf{CSP1}(P;\ Q)) & [\mathbf{seq\_CSP1}] \\ = & \mathbf{CSP2}(P;\ Q) & [\mathbf{seq\_CSP2}] \\ = & P;\ Q & \square \end{array}$$

Law seq_R1

$\mathbf{R1}(P;\ Q) = P;\ Q),$ *provided that* $\mathbf{R1}(P) = P$ *and* $\mathbf{R1}(Q) = Q$.

Proof:

$$\begin{array}{lll} & R1(P;\ Q) & [assumption] \\ = & R1(R1(P);\ R1(Q)) & [\mathbf{R1}] \\ = & ((P \wedge tr \leq tr');\ (Q \wedge tr \leq tr')) \wedge tr \leq tr' & [3.3.2] \\ = & (\exists\, tr_0, v_0 \bullet P[tr_0, v_0/tr', v'] \wedge tr \leq tr_0 \wedge Q[tr_0, v_0/tr, v] \wedge tr_0 \leq tr') \wedge tr \leq tr' & [property\ of\ \leq] \\ = & \exists\, tr_0, v_0 \bullet P[tr_0, v_0/tr', v'] \wedge tr \leq tr_0 \wedge Q[tr_0, v_0/tr, v] \wedge tr_0 \leq tr' & \begin{bmatrix} predicate \\ calculus \end{bmatrix} \\ = & \exists\, tr_0, v_0 \bullet (P \wedge tr \leq tr')[tr_0, v_0/tr', v'] \wedge (Q \wedge tr \leq tr')[tr_0, v_0/tr, v] & [\mathbf{R1}] \end{array}$$

[5] Ψ_1, Ψ_2 and Ψ_3 are logic formulae in terms of ref', $0.ref$ and $1.ref$.

$$\begin{array}{lll}
= & \exists\, tr_0, v_0 \bullet (\mathbf{R1}(P)[tr_0, v_0/tr', v'] \wedge \mathbf{R1}(Q)[tr_0, v_0/tr, v]) & [3.3.2] \\
= & \mathbf{R1}(P);\ \mathbf{R1}(Q) & [assumption] \\
= & P;\ Q & \square
\end{array}$$

Law seq_R2

$\mathbf{R2}(P;\ Q) = P;\ Q),$ *provided that* $\mathbf{R2}(P) = P$ *and* $\mathbf{R2}(Q) = Q.$

Proof:

$$\begin{array}{lll}
 & \mathbf{R2}(P;\ Q) & [assumption] \\
= & \mathbf{R2}(\mathbf{R2}(P);\ \mathbf{R2}(Q)) & [3.3.2] \\
= & \mathbf{R2}(\exists\, tr_0, V_0 \bullet \mathbf{R2}(P)[tr_0, v_0/tr', v'] \wedge \mathbf{R2}(Q)[tr_0, v_0/tr, v]) & [\mathbf{R2}] \\
= & \left(\begin{array}{l} p[<>, tr' - tr/tr, tr'][tr_0, v_0/tr', v'] \wedge \\ Q[<>, tr' - tr/tr, tr'][tr_0, v_0/tr, v] \end{array} \right) [<>, tr' - tr/tr, tr'] & [substitution] \\
= & \begin{array}{l} P[v_0/v'][<>, tr_0 - tr- <> /tr, tr'] \wedge \\ Q[v_0/v'][<>, tr' - tr_0- <> /tr, tr'] \end{array} & [-] \\
= & \begin{array}{l} P[v_0/v'][<>, tr_0 - tr/tr, tr'] \wedge \\ Q[v_0/v'][<>, tr' - tr_0/tr, tr'] \end{array} & [\mathbf{R2}] \\
= & \mathbf{R2}(P)[tr_0, v_0/tr', v'] \wedge \mathbf{R2}(Q)[tr_0, v_0/tr, v'] & [3.3.2] \\
= & \mathbf{R2}(P);\ \mathbf{R2}(Q) & [assumption] \\
= & P;\ Q & \square
\end{array}$$

Law seq_R3

$\mathbf{R3}(P;\ Q) = P;\ Q),$ *provided that* $\mathbf{R3}(P) = P$ *and* $\mathbf{R3}(Q) = Q.$

Proof:

$$\begin{array}{lll}
 & P;\ Q & [assumption] \\
= & \mathbf{R3}(P);\ \mathbf{R3}(Q) & [\mathbf{R3}] \\
= & (II \lhd wait \rhd P);\ (II \lhd wait \rhd Q) & [property\ of\ \lhd \rhd] \\
= & (II;\ (II \lhd wait \rhd Q)) \lhd wait \rhd (P;\ (II \lhd wait \rhd Q)) & [property\ of\ II] \\
= & (II \lhd wait \rhd Q) \lhd wait \rhd (P;\ (II \lhd wait \rhd Q)) & [\mathbf{R3}] \\
= & (II \lhd wait \rhd Q) \lhd wait \rhd (P;\ \mathbf{R3}(Q)) & [assumption] \\
= & II \lhd wait \rhd Q) \lhd wait \rhd (P;\ Q) & [conditional\ choice] \\
= & (wait \wedge II \vee \neg wait \wedge Q) \wedge wait \vee \neg wait \wedge (P;\ Q) & [positional\ calculus] \\
= & wait \wedge II \vee \neg wait \wedge (P;\ Q) & [conditional\ choice] \\
= & II \lhd wait \rhd P;\ Q & [\mathbf{R3}] \\
= & \mathbf{R3}(P;\ Q) & \square
\end{array}$$

Law seq_CSP1

$\mathbf{CSP1}(P;\ Q) = P;\ Q),$ *provided that* $\mathbf{CSP1}(P) = P$ *and* $\mathbf{CSP1}(Q) = Q.$

Proof:

$$\begin{array}{lll}
 & P;\ Q & [assumption] \\
= & \mathbf{CSP1}(P);\ \mathbf{CSP1}(Q) & [\mathbf{CSP1}] \\
= & (P \vee \neg ok \wedge tr \leq tr');\ (Q \vee \neg ok \wedge tr \leq tr') & \left[\begin{array}{c} sequebtial \\ composition \end{array} \right] \\
= & \exists\, obs_0 \bullet (P \vee \neg ok \wedge tr \leq tr')[obs_0/obs'] \wedge (Q \vee \neg ok \wedge tr \leq tr')[obs_0/obs] & [substitution] \\
= & \exists\, obs_0 \bullet (P[obs_0/obs'] \vee \neg ok \wedge tr \leq tr_0) \wedge (Q[obs_0/obs] \vee \neg ok_0 \wedge tr_0 \leq tr') & \left[\begin{array}{c} predicate \\ calculus \end{array} \right]
\end{array}$$

[5] The term v represents the set of remaining observational variables, as is the case of term v'. The term in the subsequent proofs represent the same meaning.

$$= \quad \exists\, obs_0 \bullet \left(\begin{array}{l} P[obs_0/obs'] \wedge Q[obs_0/obs] \vee P[obs_0/obs'] \wedge \neg ok_0 \wedge tr_0 \leq tr' \vee \\ Q[obs_0/obs] \wedge \neg ok \wedge tr \leq tr_0 \vee \neg ok \wedge tr \leq tr_0 \wedge \neg ok_0 \wedge tr_0 \leq tr' \end{array} \right) \quad \left[\begin{array}{l} \textit{predicate} \\ \textit{calculus} \end{array} \right]$$

$$\begin{array}{lll} = & \exists\, obs_0 \bullet (P[obs_0/obs'] \wedge Q[obs_0/obs] \vee \neg ok \wedge tr \leq tr') & [3.3.2] \\ = & (P;\ Q) \vee \neg ok \wedge tr \leq tr' & [\mathbf{CSP1}] \\ = & \mathbf{CSP1}(P;\ Q) & \square \end{array}$$

Law seq_CSP2

$\mathbf{CSP2}(P;\ Q) = P;\ Q),$ *provided that* $\mathbf{CSP2}(P) = P$ *and* $\mathbf{CSP2}(Q) = Q$.

Proof:

$$\begin{array}{lll} & \mathbf{CSP2}(P;\ Q) & [\mathbf{CSP2}] \\ = & (P;\ Q);\ (ok \Rightarrow ok' \wedge tr' = tr \wedge wait' = wait \wedge ref' = ref) & [\mathbf{seq-1}] \\ = & P;\ (Q;\ (ok \Rightarrow ok' \wedge tr' = tr \wedge wait' = wait \wedge ref' = ref)) & [\mathbf{CSP2}] \\ = & P;\ CSP2(Q) & [\textit{assumption}] \\ = & P;\ Q & \square \end{array}$$

4. The Theories of Arithmetic and Boolean Expressions

```
% syntax for arithmetic expressions
Aexp: Datatype
 BEGIN
  anum(n:int): anum?
  avar(x:Vars): avar?
  aplus(exp1,exp2:Aexp): apluse?
  aminus(exp1,exp2:Aexp): aminus?
  amult(exp1,exp2:Aexp): amult?
 END Aexp

% syntax for boolean expressions
Bexp: Datatype
 BEGIN
  bbool(b:bool): bbool?
  beq(exp1,exp2:Aexp): beq?
  blt(exp1,exp2:Aexp): blt?
  bnot(b:bool): bnot?
  band(b1,b2:Bexp): band?
  bor(b1,b2:Bexp):bor?
 END Bexp

% semantics for arithmetic expressions
aeval(a: Aexp): RECURSIVE S_int =
  (CASES a of
    anum(n): lambda (s:S): n,
    avar(x): lambda (s:S): s(x),
    aplus(exp1,exp2): lambda (s:S): (aeval(exp1)(s) + aeval(exp2)(s)),
    aminus(exp1,exp2): lambda (s:S): (aeval(exp1)(s) - aeval(exp2)(s)),
    amult(exp1, exp2): lambda (s:S): (aeval(exp1)(s) * aeval(exp2)(s))
   ENDCASES)
  MEASURE a by <<
% semantics for boolean expressions
beval(b: Bexp): RECURSIVE S_bool =
  (CASES b of
    bbool(b): lambda (s:S): b,
    beq(exp1,exp2): lambda (s:S):
      (IF aeval(exp1)(s) = aeval(exp2)(s) THEN TRUE ELSE FALSE ENDIF),
    blt(exp1,exp2): lambda (s:S):
      (IF aeval(exp1)(s) < aeval(exp2)(s) THEN TRUE ELSE FALSE ENDIF),
    bnot(b): lambda (s:S): (NOT b),
    band(b1,b2): lambda (s:S): (beval(b1)(s) AND beval(b2)(s)),
    bor(b1,b2): lambda (s:S): (beval(b1)(s) OR beval(b2)(s))
   ENDCASES)
  MEASURE b BY <<
```