

Process Algebra CSP – A Technique to Model Concurrent Programs

Hui Shi

January 23, 2002



Contents

- CSP-Processes
- Operational Semantics
- Denotational Semantics I
 - The traces model
 - Specification
 - Refinement
- Denotational Semantics II

Denotational Semantics I

The traces model (\mathcal{T})

- Definitions

A **trace** of process P is a sequence of communications between the environment and P .

For any process P , **traces**(P) is defined to be the set of all its finite traces, i.e., members of Σ^* .

The set of all non-empty, prefix-closed subsets of Σ^* is called **traces model** (\mathcal{T}).

- Properties of **traces**(P)

- **traces**(P) is non-empty: $\langle \rangle \in \text{traces}(P)$

- **traces**(P) is prefix closed: if $s \hat{ } t \in \text{traces}(P)$ then $s \in \text{traces}(P)$

- Traces related operations

$$\mathit{initials}(P) = \{a \mid \langle a \rangle \in \mathit{traces}(P)\}$$

$$\mathit{traces}(P/s) = \{t \mid s \hat{\ } t \in \mathit{traces}(P)\} \quad (P \text{ after } s)$$

$\#S$: length of s

$$s \leq t \equiv \exists u. s \hat{\ } u = t \quad (s \text{ is a prefix of } t)$$

$$\langle \rangle \upharpoonright A = \langle \rangle$$

$$s \hat{\ } \langle a \rangle \upharpoonright A = \mathbf{if } a \in A \mathbf{ then } (s \upharpoonright A) \hat{\ } \langle a \rangle \mathbf{ else } (s \upharpoonright A)$$

$$s \setminus X = s \upharpoonright (\Sigma \setminus X) \quad (\text{hiding})$$

$s \downarrow c$: if c is an event then it results $\#(s \upharpoonright \{c\})$
 if c is a channel name it means the sequence of values communicated along c in s

$$\langle \rangle \parallel s = \{s\}$$

$$s \parallel \langle \rangle = \{s\}$$

$$\begin{aligned} \langle a \rangle^s \parallel \langle b \rangle^t &= \{ \langle a \rangle^u \mid u \in s \parallel \langle b \rangle^t \} \\ &\cup \{ \langle b \rangle^u \mid u \in \langle a \rangle^s \parallel t \} \end{aligned}$$

$$\begin{aligned}
s \parallel_X t &= t \parallel_X s \\
\langle \rangle \parallel_X t &= \mathbf{if} (t \upharpoonright X \neq \{\}) \mathbf{then} \{\} \mathbf{else} \{t\} \\
\langle x \rangle^{\wedge} s \parallel_X \langle y \rangle^{\wedge} t &= \{\}, \mathbf{if} x, y \in X \wedge x \neq y \\
&= \{\langle x \rangle^{\wedge} u \mid u \in s \parallel_X t\}, \mathbf{if} x, y \in X \wedge x = y \\
&= \{\langle y \rangle^{\wedge} u \mid u \in \langle x \rangle^{\wedge} s \parallel_X t\}, \mathbf{if} x \in X \wedge y \notin X \\
&= \{\langle y \rangle^{\wedge} u \mid u \in \langle x \rangle^{\wedge} s \parallel_X t\} \\
&\cup \{\langle x \rangle^{\wedge} u \mid u \in s \parallel_X \langle y \rangle^{\wedge} t\}, \mathbf{if} x, y \notin X
\end{aligned}$$

- Rules for working out $traces(P)$

- Fundamental operators

$$traces(Stop) = \{\langle \rangle\}$$

$$traces(e \rightarrow P) = \{\langle \rangle\} \cup \{\langle a \rangle^s \mid a \in comms(e) \wedge s \in traces(subs(a, e, P))\}$$

$$traces(P \square Q) = traces(P) \cup traces(Q)$$

$$traces(P \sqcap Q) = traces(P) \cup traces(Q)$$

$$traces(P \leftarrow b \rightarrow Q) = \mathbf{if} \ b \ \mathbf{then} \ traces(P) \ \mathbf{else} \ traces(Q)$$

$$traces(\mu . p . F(p)) = \bigcup_{i=0}^{\infty} traces(F^i(Stop))$$

- Parallel operators

$$\text{traces}(P \parallel Q) = \text{traces}(P) \cap \text{traces}(Q)$$

$$\text{traces}(P \underset{X}{\parallel} \underset{Y}{\parallel} Q) = \{s \in (X \cup Y)^* \mid s \upharpoonright X \in \text{traces}(P) \wedge s \upharpoonright Y \in \text{traces}(Q)\}$$

$$\text{traces}(P \parallel\!\!\parallel Q) = \bigcup \{s \parallel\!\!\parallel t \mid s \in \text{traces}(P) \wedge t \in \text{traces}(Q)\}$$

$$\text{traces}(P \underset{X}{\parallel} Q) = \bigcup \{s \underset{X}{\parallel} t \mid s \in \text{traces}(P) \wedge t \in \text{traces}(Q)\}$$

- Hiding and renaming

$$\text{traces}(P \setminus A) = \{s \upharpoonright \Sigma \setminus A \mid s \in \text{traces}(P)\}$$

$$\text{traces}(f \parallel P \parallel) = \{f(s) \mid s \in \text{traces}(P)\}$$

- Sequential operator

$$\text{traces}(\text{Skip}) = \{\langle \rangle, \langle \checkmark \rangle\}$$

$$\begin{aligned} \text{traces}(P; Q) = & (\text{traces}(P) \cap \Sigma^*) \\ & \cup \{s \hat{ } t \mid s \hat{ } \langle \checkmark \rangle \in \text{traces}(P) \wedge t \in \text{traces}(Q)\} \end{aligned}$$

- Time-out

$$\text{traces}(P \triangleright Q) = \text{traces}(P) \cup \text{traces}(Q)$$

Example 1 (Proving the following equivalence)

$$P' \triangleright Q' =_{\mathcal{T}} (P' \sqcap Stop) \sqcap Q'$$

$$P' = a \rightarrow P \qquad Q' = b \rightarrow Q$$

Example 2 (Proving in \mathcal{T} the equivalences)

$$P \parallel Q = P \underset{\Sigma}{\parallel} Q$$

$$P \underset{X}{\parallel} \underset{Y}{\parallel} Q = P \underset{X \cap Y}{\parallel} Q$$

$$P \parallel\parallel Q = P \underset{\{\}}{\parallel} Q$$

Trace specifications

- Definition

A **specification** is some condition that we wish a given process to satisfy.

$$P \text{ sat } R(tr) \Leftrightarrow \forall tr \in \text{traces}(P). R(tr)$$

- Some rules of trace specifications

Stop sat ($tr = \langle \rangle$)

$$\frac{\forall a \in \text{comms}(e) \wedge \text{subs}(a, e, P) \text{ sat } \text{subs}(a, e, R(tr))}{e \rightarrow P \text{ sat } (tr = \langle \rangle \vee \exists a, tr'. tr = \langle a \rangle \wedge tr' \wedge \text{subs}(a, e, R(tr')))}$$

$$\frac{P \text{ sat } R(tr) \wedge Q \text{ sat } R(tr)}{P \sqcap Q \text{ sat } R(tr)}$$

$$\frac{P \text{ sat } R(tr) \wedge Q \text{ sat } R(tr)}{P \sqcap Q \text{ sat } R(tr)}$$

$$\frac{P \text{ sat } R(tr) \wedge \forall tr. R(tr) \Rightarrow R'(tr)}{P \text{ sat } R'(tr)}$$

$$\frac{P \text{ sat } R(tr) \wedge P \text{ sat } R'(tr)}{P \text{ sat } R(tr) \wedge R'(tr)}$$

Example 3 (Some trace specifications)

$$P = up \rightarrow down \rightarrow P$$

$$P \text{ sat } (tr \downarrow down \leq tr \downarrow up \leq tr \downarrow down + 1)$$

$$COPY = left?.x \rightarrow right.x \rightarrow COPY$$

$$COPY \text{ sat } (tr \downarrow right \leq tr \downarrow left)$$

$$COUNT(0) = up \rightarrow COUNT(1)$$

$$COUNT(n) = (up \rightarrow COUNT(n + 1))$$

$$\square (down \rightarrow COUNT(n - 1)) \quad (n > 0)$$

$$COUNT(n) \text{ sat } (tr \downarrow down \leq tr \downarrow up + n)$$

Traces refinement

- Definition

Process Q **refines** process P (written $P \sqsubseteq_T Q$) if $traces(Q) \subseteq traces(P)$.

$$P \sqsubseteq_T Q \equiv P =_T Q \sqcap P$$

$$Q \text{ sat } R(tr) \Leftrightarrow P_R \sqsubseteq_T Q$$

where P_R is the most nondeterministic process satisfying R for any satisfiable behavioural trace specification R .

Example 4 (A change-giving machine)

A change-give machine which takes in 1 Euro coins and gives changes in 10, 20, 50 cent coins. It should have the following events:

$$\{in, out.10, out.20, out.50\}$$

Specification I *A machine has never given out more money than it has received.*

$$\Sigma (tr \downarrow out) \leq (tr \downarrow in) * 100$$

Specification II *A machine never receives another 1 Euro until it has given out all changes after receiving one.*

$$(tr \downarrow in) * 100 \leq \Sigma (tr \downarrow out) + 100$$

The following processes satisfy the two specifications:

CHANGE1 = in → out.50 → out.50 → CHANGE1

*CHANGE2 = in → out.50 → out.20 →
out.20 → out.10 → CHANGE2*

*CHANGE3 = in → out.50 → out.20 →
out.10 → out.10 →
out.10 → CHANGE3*

...

A more general specification:

$$CHANGE = in \rightarrow CH_OUT(100)$$

$$CH_OUT(x) = \mathbf{if} (x == 0) \mathbf{then} CHANGE \\ \mathbf{if} (x \geq 50) \mathbf{then} CH_OUT1(x) \\ \mathbf{if} (50 > x \geq 20) \mathbf{then} CH_OUT2(x) \\ \mathbf{else} out.10 \rightarrow CH_OUT(x - 10)$$

$$CH_OUT1(x) = out.10 \rightarrow CH_OUT(x - 10)$$

$$\sqcap out.20 \rightarrow CH_OUT(x - 20)$$

$$\sqcap out.50 \rightarrow CH_OUT(x - 50)$$

$$CH_OUT2(x) = out.10 \rightarrow CH_OUT(x - 10)$$

$$\sqcap out.20 \rightarrow CH_OUT(x - 20)$$

- Some advantages in identifying P_R for specification R
 - Mechanical verification with tools like FDR.
 - Refinement has many properties that can be exploited, for example

$$P \sqsubseteq Q \wedge Q \sqsubseteq R \Leftrightarrow P \sqsubseteq R$$

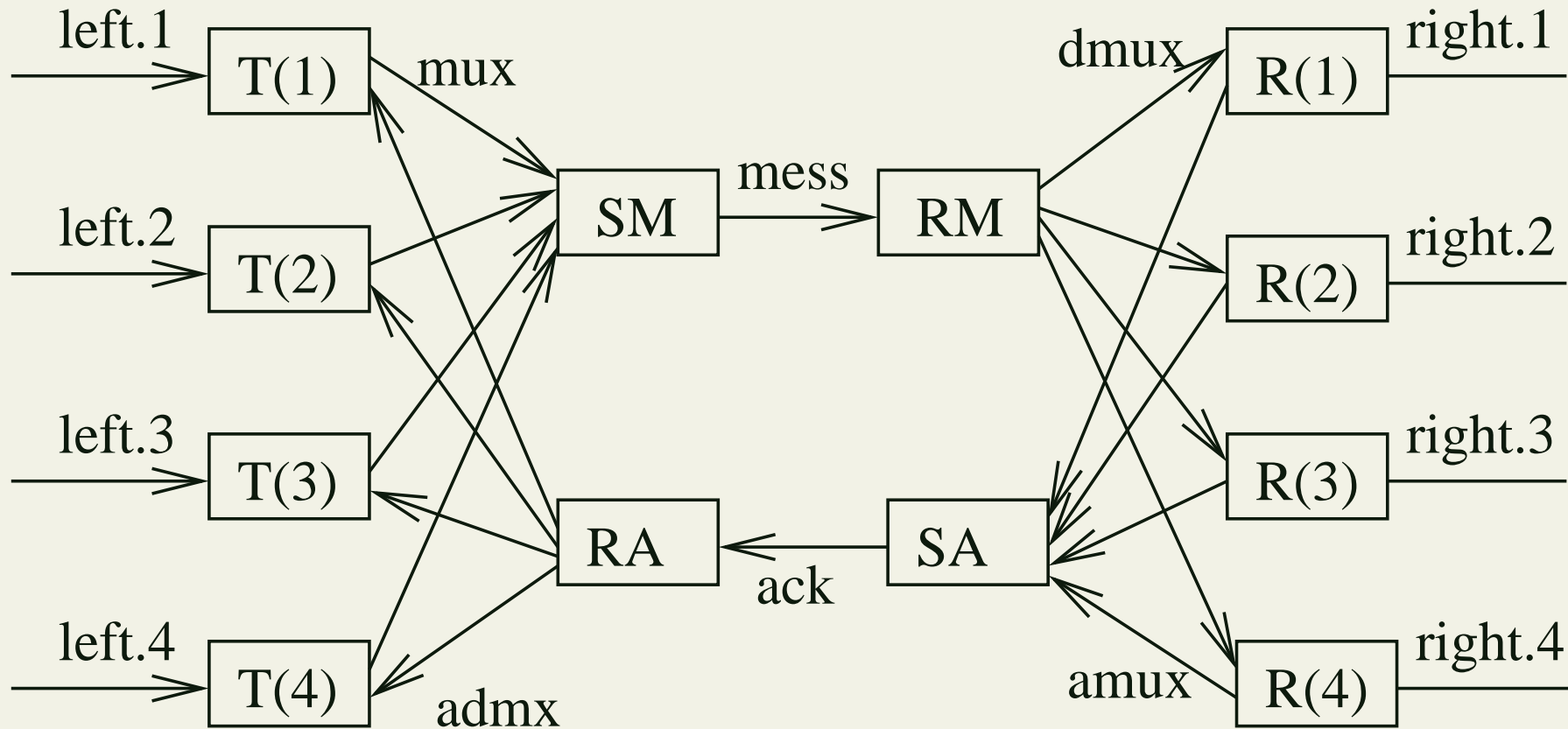
$$P \sqsubseteq Q \wedge Q \sqsubseteq P \Leftrightarrow P = Q$$

- **Compositionality**, that is

$$P \sqsubseteq Q \Rightarrow C[P] \sqsubseteq C[Q].$$

- **Stepwise refinement**

$$Spec \sqsubseteq P_1 \sqsubseteq \dots \sqsubseteq P_n \sqsubseteq Impl$$



Example 5 (Multiplexed buffer)

$$\begin{aligned}
\text{LHS} &= ((\parallel i : \{1, 2, 3, 4\} @ \mathbf{T}(i)) \parallel (\mathbf{SM} \parallel \mathbf{RA})) \setminus X \\
&\quad \parallel_X \\
\text{RHS} &= ((\parallel i : \{1, 2, 3, 4\} @ \mathbf{R}(i)) \parallel (\mathbf{RM} \parallel \mathbf{SA})) \setminus Y \\
&\quad \parallel_Y \\
\text{SYS} &= (\text{LHS} \parallel_Z \text{RHS}) \setminus Z
\end{aligned}$$

$$X = \{mux, admx\} \quad Y = \{amux, dmux\} \quad Z = \{mess, ack\}$$

$$\begin{aligned}T(i) &= \text{left}.i?x \rightarrow \text{mux}.i.x \rightarrow \text{adm}.i \rightarrow T(i) \\R(i) &= \text{dmux}.i?x \rightarrow \text{right}.i.x \rightarrow \text{amux}.i \rightarrow R(i) \\SM &= \text{mux}?i.x \rightarrow \text{mess}.i.x \rightarrow SM \\RM &= \text{mess}?i.x \rightarrow \text{dmux}.i.x \rightarrow RM \\SA &= \text{amux}?i \rightarrow \text{ack}.i \rightarrow SA \\RA &= \text{ack}?i \rightarrow \text{adm}.i \rightarrow RA\end{aligned}$$

Question: Does SYS satisfies the specification

$$\forall i : \{1, 2, 3, 4\}. tr \downarrow \text{right}.i \leq tr \downarrow \text{left}.i$$



$$\begin{aligned} \text{COPY}(i) &= \text{left}.i?x \rightarrow \text{right}.i.x \rightarrow \text{COPY}(i) \\ \text{SPEC} &= ||| i : \{1, 2, 3, 4\} @ \text{COPY}(i) \end{aligned}$$