

Process Algebra CSP – A Technique to Model Concurrent Programs

Hui Shi

January 14, 2002



Bibliographic Remarks

- C. A. R. Hoare: *Communicating Sequential Processes*, Prentice Hall, 1985.
- A. W. Roscoe: *The Theory and Practice of Concurrency*, Prentice Hall, 1998.
- *Failures-Divergence Refinement: FDR2 User Manual*, Formal Systems (Europe) Ltd.
- Web site:
`pttp://www.comlab.ox.ac.uk/oucl/publications/books/concurrency/`

Contents

- CSP-Processes
 - Fundamental operators
 - Parallel operators
 - Hiding and renaming
 - Termination and sequential composition
- Operational Semantics
- Denotational Semantics
- Specification and Refinement

CSP-Processes

Fundamental operators

- Prefixing $a \rightarrow P$

$up \rightarrow down \rightarrow up \rightarrow down \rightarrow \dots$

- Recursion $P = F(P)$ oder $\mu p.F(p)$

$P = up \rightarrow down \rightarrow P$

$\mu p.up \rightarrow down \rightarrow p$

- Guarded alternative $P \mid Q$

$a_1 \rightarrow P_1 \mid a_2 \rightarrow P_2$

$?x : A \rightarrow P(x)$

• Choice operators

- External choice $P \square Q$

$$a \rightarrow ((a \rightarrow P) \square (b \rightarrow Q))$$

$$(a \rightarrow a \rightarrow P) \square (a \rightarrow b \rightarrow Q)$$

- Internal choice $P \sqcap Q$

$$a \rightarrow ((a \rightarrow P) \sqcap (b \rightarrow Q))$$

- Conditional choice $P \leftarrow b \rightarrow Q$

$$P = \text{left?}x \rightarrow \text{right!}((-x) \leftarrow x < 0 \rightarrow x) \rightarrow P$$

$$P = \text{left?}x \rightarrow ((\text{right!}(-x) \rightarrow P) \leftarrow x < 0 \rightarrow (\text{right!}x) \rightarrow P))$$

Definition 1 (Determinism and Nondeterminism)

A *deterministic* process is one where the range of events offered to the environment depends only on things it has seen, i.e., the sequence of communications so far.

A process is *nondeterministic* when some internal decision can lead to uncertainty about what will happen.

Parallel operators

- Synchronous parallel $P \parallel Q$

$$(a \rightarrow \text{REPEAT}) \parallel \text{REPEAT}$$

$$\text{REPEAT} = ?x : \Sigma \rightarrow x \rightarrow \text{REPEAT} \quad (a \in \Sigma)$$

- Alphabetized parallel $P \mathop{A \parallel}_B Q$

$$\text{COPY}(in, out) = in?x : T \rightarrow out.x \rightarrow \text{COPY}(in, out)$$

$$\text{COPYS} =$$

$$\text{COPY}(c_0, c_1) \{c_0, c_1\} \parallel \{c_1, \dots, c_n\} (\text{COPY}(c_1, c_2) \{c_1, c_2\} \parallel \{c_2, \dots, c_n\})$$

$$(\dots (\text{COPY}(c_{n-2}, c_{n-1}) \{c_{n-2}, c_{n-1}\} \parallel \{c_{n-1}, c_n\})$$

$$\text{COPY}(c_{n-1}, c_n) \dots))$$

- Interleaving $P \parallel Q$

$\text{FORK}(i) = (\text{picksup}.i.i \rightarrow \text{putsdown}.i.i \rightarrow \text{FORK}(i))$

□ $(\text{picksup}.i\ominus 1.i \rightarrow \text{putsdown}.i\ominus 1.i \rightarrow \text{FORK}(i))$

$\text{PHIL}(i) = \text{thinks}.i \rightarrow \text{sits}.i \rightarrow \text{picksup}.i.i \rightarrow \text{picksup}.i.i\oplus 1$
 $\rightarrow \text{eats}.i \rightarrow \text{putsdown}.i.i\oplus 1 \rightarrow \text{putsdown}.i.i$
 $\rightarrow \text{getsup}.i \rightarrow \text{PHIL}(i)$

$\text{FORK} = \text{FORK}(0) \parallel \text{FORK}(1) \parallel \dots \parallel \text{FORK}(4)$

$\text{PHIL} = \text{PHIL}(0) \parallel \text{PHIL}(1) \parallel \dots \parallel \text{PHIL}(4)$

- Generalized parallel $P \underset{A}{\parallel} Q$

$$\text{PHIL} \underset{\{picksup, puttdown\}}{\parallel} \text{FORK}$$

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

$$P \underset{A}{\parallel} \underset{B}{\parallel} Q = P \underset{A \cap B}{\parallel} Q$$

Hiding and renaming

- Hiding $P \setminus A$

COPYS $\setminus \{c_1, \dots, c_{n-1}\}$

$(a \rightarrow P \sqcap b \rightarrow Q) \setminus \{b\} =$

$(a \rightarrow (P \setminus \{b\}) \sqcap Stop) \sqcap (Q \setminus \{b\})$

$P \triangleright Q = (P \sqcap Stop) \sqcap Q$ (*time-out*)

- Renaming and alphabet transformations $P \parallel f \parallel$

- Injective functions $f : \Sigma \rightarrow \Sigma$

$g(in.x) = left.x$ $g(out.x) = right.x$

$COPY(in, out) \parallel g \parallel = COPY(left, right)$

- Non-injective functions

$$\text{SPLIT} = in?x : T \rightarrow$$

$$((out_1.x \rightarrow \text{SPLIT}) \leftarrow x \in S \rightarrow (out_2.x \rightarrow \text{SPLIT}))$$

$$\text{SPLIT} \parallel \text{forget} \parallel = in \rightarrow (out_1 \rightarrow \text{SPLIT} \sqcap out_2 \rightarrow \text{SPLIT})$$

$$\text{forget}(in.x) = in$$

$$\text{forget}(out_1.x) = out_1$$

$$\text{forget}(out_2.x) = out_2$$

Termination and sequential composition

- What is termination? *Stop* vs. *Skip*

$$\textit{Skip} = \checkmark \rightarrow \textit{Stop}$$

- Sequential operator $P; Q$

$$P^* = P; P^*$$

$$\text{COPY}(in, out) = (in?x : T \rightarrow out.x \rightarrow \textit{Skip})^*$$

$$(\text{FOR } n = a, b \text{ DO } P) =$$

$$\textit{Skip} \not\leftarrow a > b \not\rightarrow (P[a/n]; (\text{FOR } n = a + 1, b \text{ DO } P))$$

Examples

Example 1 (Piping)

$$P \gg Q = (P \parallel f \parallel \parallel Q \parallel g \parallel) \setminus \{mid\}$$

$\{mid\}$

$$f(right) = mid \quad g(left) = mid$$

$$ITER = left?(data, x) \rightarrow right.(data, F(data, x)) \rightarrow ITER$$

$$ITER \gg ITER \gg \dots \gg ITER$$

Exercise: Develop a process which produces the fibonacci numbers less than 100.

Example 2 (Buffers)

$$BUFF(\langle \rangle) = left?x \rightarrow BUFF(\langle x \rangle)$$

$$BUFF(s \hat{ } \langle a \rangle) = ((left?x \rightarrow BUFF(\langle x \rangle \hat{ } s \hat{ } \langle a \rangle)) \sqcap Stop)$$

$$\square right.a \rightarrow BUFF(s)$$

$$PROD = produce?x \rightarrow left.x \rightarrow PROD$$

$$CONS = right?x \rightarrow consume.x \rightarrow CONS$$

$$PC_SYS = ((PROD \parallel CONS)$$

$$\parallel BUFF(\langle \rangle)) \setminus \{left, right\}$$

$$\{left, right\}$$

Example 3 (Shared variables)

$$SVAR(x) = (read.x \rightarrow SVAR(x))$$

$$\square (write.y \rightarrow SVAR(y))$$

$$USER(i) = \dots read.x \dots write.v \dots USER(i)$$

$$SVAR_SYS = (SVAR(init) \quad \parallel \quad \{read, write\} \\ (USER(1) \parallel \dots \parallel USER(n))) \setminus \{read, write\}$$

Summary: CSP operators vs. FDR operators

	CSP operator	FDR operator
prefix	\rightarrow	\dashrightarrow
external choice	\square	$[]$
internal choice	\sqcap	$ \sim $
parallel operator	\parallel	
	\parallel A	$[A]$
	$A \parallel B$	$[A \parallel B]$
		$[c \dashleftarrow \dashrightarrow c']$

	CSP operator	FDR operator
interleaving	$ $	$ $
hiding	\backslash	\backslash
sequential comp.	$;$	$;$
condition	$\langle b \rangle$	if-then-else
interrupt	Δ_e	\wedge
time-out	\triangleright	$[\triangleright$
stop-Process	Stop	STOP
skip-Process	Skip	SKIP