Introduction to Communicating Sequential Process (CSP) (Lecture 8)

Mannheim, September 2007
Contents

• Sequential Composition
• Semantics
Termination

Forms of unsuccessful termination resulting from design flaws are

- *Stop*, representing deadlock
- *Div* representing livelock.

By comparison process *Skip* represents deliberate successful termination on completion of a task.

A terminating trace of process *P* is a trace *t* after which *P* may terminate

\[ P \text{ after } t \sqsubseteq \text{Skip}. \]
Sequential Composition

If $P$ and $Q$ are processes over then

$$P ; Q$$

denotes their *sequential composition* which first behaves like $P$; if $P$ terminates it then behaves like $Q$; if $P$ doesn’t terminate neither does $P ; Q$. The *iteration*, $P^*$, of $P$ is defined $P^* = P ; P^*$. 
Sequential Composition: Example

A vending machine which serves one customer is

\[ V_1 = \text{coin} \rightarrow (\text{choc} \rightarrow \text{Skip} \mid \text{toffee} \rightarrow \text{Skip}) . \]

One which serves two is

\[ V_1 ; V_1. \]

And one which serves customers forever is

\[ V = V_1^*. \]
Sequential Composition: Example

Recall the infinite mutual recursion

$$R = R_0 = (\text{around} \rightarrow R \mid \text{up} \rightarrow R_1)$$
$$R_{n+1} = (\text{up} \rightarrow R_{n+2} \mid \text{down} \rightarrow R_n).$$

That process is expressed in finite form using sequential composition

$$Z = (\text{around} \rightarrow Z \mid \text{up} \rightarrow P ; Z)$$
$$P = (\text{up} \rightarrow P ; P \mid \text{down} \rightarrow \text{Skip}).$$
Sequential Composition: Example

The language consisting of strings having any number of $a$’s, followed by a $b$, followed by the same number of $c$’s as $a$’s is

$$\{<a>^n \wedge <b>^n \wedge <c>^n \mid n \in \mathbb{N}\}.$$ 

A process for that language is

$$L = \mu X \cdot (b \rightarrow \text{Skip} \quad | \quad a \rightarrow (X ; c \rightarrow \text{Skip})).$$
Sequential Composition: Example

The language whose strings start as above and are then followed by a $d$ and then the same number of $e$’s as $a$’s is

\[ \{ <a>^n \land <b>^n \land <c>^n \land <d>^n \land <e>^n \mid n \in \mathbb{N} \}. \]

A process for that language is

\[ M = (L; d \rightarrow \text{Skip}) \|\{c,d\}\| f L, \]

where the injective relabelling $f$ is defined

\[ f a = c, f b = d, f c = e. \]
Sequential Composition: Laws

Sequential composition is associative and distributive in each argument, with unit Skip

- \((P ; Q) ; R = P ; (Q ; R)\)
- \((P \sqcap Q) ; R = (P ; R) \sqcap (Q ; R)\)
- \(P ; (Q \sqcap R) = (P ; Q) \sqcap (P ; R)\)
- \(\text{Skip} ; P = P = P ; \text{Skip}\)

Stop is a left zero, as is any divergent process

- \(\text{Stop} ; P = \text{Stop}\)
- \(\text{Div} ; P = \text{Div} \ldots\)
Sequential Composition: Laws

- Processes do not share their local variables. Thus in $P; Q$ the final state of $P$ is independent of the initial state of $Q$.

For example in the sequential composition

$$(\ldots \rightarrow \text{out}!x \rightarrow \text{Skip}) ; (\text{in}?x \rightarrow \ldots)$$

the value of $x$ in the first process has no relationship to the value of $x$ in the second.
Sequential Composition: Laws

For example

\[ in?x \rightarrow out!x \rightarrow Skip \]
\[ \neq \]
\[ (in?x \rightarrow Skip) ; (out!x \rightarrow Skip). \]

Indeed the latter process may output any value of the appropriate type on channel out whilst the former can output only the value it has input on in.
Sequential Composition: Laws

• However, provided a variable $x$ is not free in process $Q$

$$(?x:A \rightarrow P(x));Q = ?x:A \rightarrow (P(x);Q)$$
Sequential Composition: Traces

The event of successful termination is represented by $\checkmark$, an event not in any $\Sigma$. It occurs only as the last event of a terminating process and is not available like other elements of for synchronisation, nor can it be hidden or renamed.

$$traces\ Skip = \{<> , < \checkmark > \}.$$ 

Write

$$\Sigma \checkmark = \Sigma \{ \checkmark \}$$

$$\Sigma \checkmark^* = \Sigma^* \{ t^< \checkmark > | t \Sigma^* \}$$
Sequential Composition: Traces

• The traces of $P; Q$ consist of those of $P$ or those terminating traces of $P$ with $\sqrt{}$ removed and concatenated with a trace of $Q$

$$\text{traces}(P ; Q) = \text{traces } P \quad \{s ^ t | (s ^ {< \sqrt{}}) \quad \text{traces } P \text{ and } t \quad \text{traces } Q\}.$$
Sequential Composition: Traces

- The traces of \( P; Q \) consist of those of \( P \) or those terminating traces of \( P \) with \( \sqrt{\ } \) removed and catenated with a trace of \( Q \)

\[
\text{traces}(P; Q) = \text{traces } P \ {s \upharpoonright t \mid (s \upharpoonright < \sqrt{\ } \quad \text{traces } P \quad \text{and} \quad t \quad \text{traces } Q}).
\]
Sequential Composition: Traces

- The traces of $P; Q$ consist of those of $P$ or those terminating traces of $P$ with $\sqrt{}$ removed and catenated with a trace of $Q$

$$\text{traces}(P; Q) = \text{traces } P \cup \{s^\ast t | (s^\ast <\sqrt{} >) \text{ traces } P \text{ and } t \text{ traces } Q\}.$$
Sequential Composition: Traces

- The traces of $P; Q$ consist of those of $P$ or those terminating traces of $P$ with $\sqrt{}$ removed and catenated with a trace of $Q$

\[
\text{traces}(P; Q) = \text{traces } P \cup \{s^\uparrow t | (s^\uparrow < \sqrt{}) \text{ traces } P \text{ and } t \text{ traces } Q\}.
\]
Assignment

• If \( x \) is a program variable and \( e \) is an expression and \( P \) a process

\[
(x := e; P)
\]

is a process that behaves like \( P \), except that the initial value of \( x \) is defined to be the initial value of the expression \( e \). Initial values of all other variables are unchanged.
Assignment: Examples

• A process that behaves like Rocket

\[ X1 = \mu X. (\text{around} \rightarrow X \mid \text{up} \rightarrow (n:=1;X)) \]
\[ <n=0> \]
\[ (\text{up} \rightarrow (n:=n+1;X) \mid \text{down} \rightarrow (n:=n-1;X)) \]
Assignment: Examples

• A process which divides a natural number $x$ by a positive number $y$, assigning the quotient to $q$ and the remainder to $r$

$$QUOT = (q := x + y; r := x - q \cdot y)$$
Assignment: Laws

- \((x:=x) = \text{SKIP}\)
- \((x:=e; x:=f(x)) = (x:=f(e))\)
- If \(x, y\) is a list of distinct variables \((x:=e) = (x,y := e,y)\)
- If \(x,y,z\) are of the same length as \(e,f,g\) respectively
  \((x,y,z := e,f,g) = (x,z,y := e,g,f)\)
- \(x:=e ; (P \triangleleft b(x)\triangleright Q) = (x:=e;P) \triangleleft b(x)\triangleright (x:=e;Q)\)
- \(((x:=e;P)||Q) = (x:=e ; (P||Q))\) provided that \(P\) and \(Q\) are data independent...
Assignment: Laws

• \((x:=x) = \text{SKIP}\)
• \((x:=e; x:=f(x)) = (x:=f(e))\)
• If \(x, y\) is a list of distinct variables \((x:=e) = (x,y := e,y)\)
• If \(x,y,z\) are of the same length as \(e,f,g\) respectively \((x,y,z := e,f,g) = (x,z,y := e,g,f)\)
• \(x:=e ; (P \langle b(x) \rangle Q) = (x:=e;P) \langle b(x) \rangle (x:=e;Q)\)
• \(((x:=e;P)||Q) = (x:=e ; (P||Q))\) provided that \(P\) and \(Q\) are data independent...
Semantics

Traces do not distinguish internal and external choice

\[ \text{traces}(P \sqcap Q) = \text{traces}(P[\sqcup Q]). \]

How do those processes differ?

- Since \( a \rightarrow A[\sqcup]b \rightarrow B \) offers its environment the choice between \( a \) and \( b \) the environment cannot refuse either; whichever of them is offered by the environment must be performed.
- Since \( a \rightarrow A \sqcap b \rightarrow B \) permits its environment no say in which of the two processes occurs, it may refuse either \( a \) or \( b \) but not both; whichever of them is offered by the environment, deadlock may occur.
Semantics: Refusals

If $P$ is a (nonsequential) process its *refusals*, $\text{refusals } P$, are those subsets $E$ of the universe which it may (initially) refuse to perform; if the environment offers a general choice from $E$, deadlock may occur.

For example over universe $\{a, b\}$,

$$\text{refusals}(a \rightarrow A[], b \rightarrow B) = \{\{\}\}$$

$$\text{refusals}(a \rightarrow A \cap b \rightarrow B) = \{\{\}, \{b\}, \{a\}\}.$$  

Refusals thus distinguish internal and external choice.
Semantics: Refusals

Observe

\[
\text{refusals}(a \rightarrow A) = \{\{\}, \{b\}\}
\]
\[
\text{refusals}(b \rightarrow B) = \{\{\}, \{a\}\}.
\]

Thus from that example, and in general,

\[
\text{refusals}(P \mid\mid Q) = \text{refusals}(P) \cap \text{refusals}(Q)
\]
\[
\text{refusals}(P \cap Q) = \text{refusals}(P) \cup \text{refusals}(Q).
\]
Semantics: Failures

• If $P$ is a (nonsequential) process its failures, $\text{failures } P$, consists of those pairs $\langle t, E \rangle$ for which $t$ is a trace of $P$ and $E$ is a refusal of $P$ after $t$. Thus after it has engaged in trace $t$ the process may refuse $E$.

• For example over universe $\Sigma = \{a, b\}$:
  
  $\text{failures } \text{Stop} = \{(>,\{\}), (>,\{a\}), (>,\{b\}), (>,\{a,b\})\}$
  
  $= \{\langle >, E \rangle \mid E \subseteq \Sigma \}$

• The traces of a process can be reclaimed from its failures
  
  $\text{traces } P = \{t : \Sigma^* \mid (t, \{\}) \text{ failures } P\}$. 
Semantics: Failures

- \( \text{failures}(a \rightarrow \text{Stop}) = \{(<>), \{ \}\}, (<>), \{b\}), (<>), \{a\}), (<a>, \{ \\}), (<a>, \{a\}), (<a>, \{b\}), (<a>, \{a, b\}) = \{(<>), E) | a \not\in E \Sigma \} \{(<a>, E) | E \Sigma \} \)

- \( \text{failures}(b \rightarrow \text{Stop}) = \{(<>), \{ \}\}, (<>), \{a\}), (<>), \{b\}), (<>), \{a, b\}) = \{(<>), E) | b \not\in E \Sigma \} \{(<b>, E) | E \Sigma \} \)
Semantics: Failures

\[ \text{failures}(a \rightarrow \text{Stop}[\emptyset]b \rightarrow \text{Stop}) = \]
\[ \{ (\langle \rangle, \{ \} ), (\langle a \rangle, \{ \} ), (\langle a \rangle, \{a\} ), (\langle a \rangle, \{b\} ), (\langle a \rangle, \{a, b\} ), (\langle b \rangle, \{ \} ), (\langle b \rangle, \{a\} ), (\langle b \rangle, \{b\} ), (\langle b \rangle, \{a, b\} ) \} = \]
\[ \{ (\langle \rangle, \{ \} ) \} \]

\[ \{ (\langle a \rangle, E) \mid E \in \Sigma \} \]

\[ \{ (\langle b \rangle, E) \mid E \in \Sigma \} \]
Semantics: Failures

\[ \text{failures}(a \rightarrow \text{Stop} \sqcap b \rightarrow \text{Stop}) \]
\[ = \]
\[ \{(<>), (<>\{a\}), (<>\{b\}), (<a>, \{\}), (<a>, \{a\}), (<a>, \{b\}), (<a>, \{a, b\}), (<b>, \{\}), (<b>, \{a\}), (<b>, \{b\}), (<b>, \{a, b\})\} \]
\[ = \]
\[ \{(<>), (<>\{a\}), (<>\{b\})\} \]

\[ \{(<a>,E) \mid E \in \Sigma\} \]

\[ \{(<b>,E) \mid E \in \Sigma\} \]

With failures we can distinguish internal from external choice.
Semantics: Failures

Failures refinement ordering

\[ F \preceq_F G \equiv F \quad G. \]

Informally, every trace of \( G \) is a trace of \( F \) and if \( G \) deadlocks then \( F \) deadlocks; thus both the trace behaviour and the deadlock behaviour of \( G \) conform to that of \( F \).

Note: restricted to traces, \( \preceq_F \) yields refinement \( \preceq_T \) in the traces model:

\[ F \preceq_F G \text{ implies } F \preceq_T G. \]
Semantics: Failures

The failures model is finer than the traces model (it distinguishes $\square$ from $[[]]$) but is still not fully abstract for CSP (it doesn’t distinguish $\text{Div}$ from $\text{Stop}$).
Semantics: Divergences

Failures do not distinguish deadlock and divergence
\[ \text{failures } \text{Stop} = \text{failures } \text{Div} = \{(<> , E) \mid E \in \Sigma \} \].

How do those two processes differ?
- Stop performs no events, deadlocks immediately and does not diverge
- Div performs no events but diverges immediately.
Semantics: Divergences

For process $P$ the divergences of $P$ are the traces after which it diverges

$$\text{divergences } P = \{ t : \text{traces } P \mid P \text{ after } t = \text{Div} \}.$$  

For example over universe $\{a, b\}$,

$$\text{divergences Stop} = \{ \}$$

$$\text{divergences Div} = ?$$

Recall that $\text{Div}$ is minimal since $\text{Div} \cap P = \text{Div}$. Similarly from any point in the evolution of a process, divergent behaviour is indistinguishable from arbitrary behaviour.
Semantics: Divergences

Thus after diverging a process behaves like the least element: any trace is a divergence and any subset a refusal. Hence, because $<> \ divergences \ Div$,

$divergences \ Div = \Sigma^*$

$failures \ Div = \Sigma^* \times |P \Sigma|.$

Divergences thus distinguish $Stop$ and $Div$.
The healthiness conditions for divergences are

• if $t \ divergences \ P$ and $u \Sigma^*$ then $t \uparrow u \ divergences \ P$

• if $t \ divergences \ P$ then for all $E \Sigma$, $(t,E) \ failures \ P.$
Semantics: Failures & Divergences

• For finite universe the failures & divergences model of processes over $\Sigma$ consists of the set $N$ of pairs $(F,D) : F \times \Sigma^*$

• satisfying
  • $t \ D \ u \ \Sigma^* \ t \ ^u \ D$
  • $t \ D \ (t,E) \ F$. 
Semantics: Failures & Divergences

The space is partially ordered by the failures & divergences refinement ordering

\[(F,D) \sqsubseteq_N (G,E) \equiv F \quad G \quad D \quad E.\]

Thus both the failures behaviour and the divergences behaviour of \((G,E)\) conform to that of \((F,D)\).
Semantics: Failures & Divergences

Home exercise: Study the failures and divergence semantics of the constructs of CSP.