CS 515: Trace Semantics

Stefan D. Bruda

Winter 2019

Sequence Notations

- Sequences surrounded by $\langle$ and $\rangle$, values separated by commas (example: $\langle a, b, a, c\rangle \in \{a, b, c\}^*$)
  - $A^* \Rightarrow$ the set of exactly all the sequences over A
- Concatenation operator: $\frown$
- Pattern matching for concatenation:
  - $\langle a\rangle \frown xs$ to identify the head and the tail of a sequence: $\text{head}(\langle a\rangle \frown xs) = a$, $\text{tail}(\langle a\rangle \frown xs) = xs$
  - $xs \frown \langle a\rangle$ to identify the last element of a sequence: $\text{foot}(xs \frown \langle a\rangle) = a$, $\text{init}(xs \frown \langle a\rangle) = xs$
- Length of a sequence xs: $\#xs$
- Prefix: $\leq$ (not strict), $<$ (strict)
- Subsequence: $\preceq$ (not necessarily contiguous)
- Projection: $xs \upharpoonright A$ (only symbols from A), $xs \setminus A$ (only symbols outside A)
- Natural (homomorphic) extension from $f : A \to B$ to $f : A^* \to B^*$
- $\sigma(xs) \Rightarrow$ the set of exactly all the values appearing in xs
- Further notations: $xs \downarrow A = \#(xs \upharpoonright A)$, $a \in xs$ shorthand for $a \in \sigma(xs)$
  - Often omit braces for singletons $\{a\}$ ($xs \downarrow a$, $xs \upharpoonright a$, etc.)

Traces

- If we observe a process and note all the visible actions being performed we obtain traces $\Rightarrow$ sequences of events (over $\Sigma^{\checkmark}$)
- The set of all the possible traces:
  $TRACE = \{tr : \sigma(tr) \subseteq \Sigma^{\checkmark} . \#tr \in \mathbb{N}, \checkmark \notin \sigma(\text{init}(tr))\}$
- Concatenation $tr_1 ^\frown tr_2$ of two traces tr_1 and tr_2 defined iff $\checkmark \notin \sigma(tr_1)$
- Mapping a trace into another using a function $f : \Sigma^{\checkmark} \to \Sigma^{\checkmark}$ is possible iff $f(\checkmark) = \checkmark$ and $f(x) \neq \checkmark$ for any $x \in \Sigma$
- Notations for channel communication:
  - $\text{channel}(c.v) = c$
  - $\text{value}(c.v) = v$
  - $\text{channels}(tr) = \{\text{channel}(c) : x \in tr\}$
  - $tr \Downarrow c = \langle \text{value}(x) : x \in tr, \text{channel}(x) = c \rangle$
  - $tr \Downarrow C = \langle \text{value}(x) : x \in tr, \text{channel}(x) \in C \rangle$
- Traces and runs: Every run of a CSP process results in a trace, which is the sequence of visible actions that the process performs during that run
  - $P \stackrel{tr}{\Longrightarrow} P'$ means that there is a run of P with trace tr and ending in P'
  - $P \stackrel{tr}{\Longrightarrow}$ is shorthand for $\exists P' : P \stackrel{tr}{\Longrightarrow} P'$
  - $\text{traces}(P) = \{tr \in TRACE : P \stackrel{tr}{\Longrightarrow}\}$

Trace Semantics, Prefix

- Operational semantics is very low level, usually a more abstract semantics is preferable
- The trace model associates a set of traces traces(P) with each process P
  - Introduces an equivalence relation between processes: $P_1 =_T P_2$ iff traces(P_1) = traces(P_2)
  - Compositional, denotational model

Trace Semantics, Prefix

- Operational semantics is very low level, usually a more abstract semantics is preferable
- The trace model associates a set of traces traces(P) with each process P
  - Introduces an equivalence relation between processes: $P_1 =_T P_2$ iff traces(P_1) = traces(P_2)
  - Compositional, denotational model
- General properties of the trace model: For any process P,
  - $\langle\rangle \in \text{traces}(P)$
  - $\forall\, tr_1, tr_2 \in TRACE : tr_1 \leq tr_2 \wedge tr_2 \in \text{traces}(P) \implies tr_1 \in \text{traces}(P)$

Trace Semantics, Prefix

- Operational semantics is very low level, usually a more abstract semantics is preferable
- The trace model associates a set of traces traces(P) with each process P
  - Introduces an equivalence relation between processes: $P_1 =_T P_2$ iff traces(P_1) = traces(P_2)
  - Compositional, denotational model
- General properties of the trace model: For any process P,
  - $\langle\rangle \in \text{traces}(P)$
  - $\forall\, tr_1, tr_2 \in TRACE : tr_1 \leq tr_2 \wedge tr_2 \in \text{traces}(P) \implies tr_1 \in \text{traces}(P)$
- $\text{traces}(STOP) = \{\langle\rangle\}$
- $\text{traces}(a \to P) = \{\langle\rangle\} \cup \{\langle a\rangle^\frown tr : tr \in \text{traces}(P)\}$
- $\text{traces}(x : A \to P(x)) = \{\langle\rangle\} \cup \{\langle a\rangle^\frown tr : a \in A, tr \in \text{traces}(P(a))\}$

$$x : \{\} \to P(x) = STOP \qquad (STOP\text{-step})$$
$$x : \{b\} \to P(x) = b \to P(b) \qquad (\text{prefix})$$

- $\text{traces}(c!v \to P) = \{\langle\rangle\} \cup \{\langle c.v\rangle^\frown tr : tr \in \text{traces}(P)\}$
- $\text{traces}(c?v : T \to P) = \{\langle\rangle\} \cup \{\langle c.v\rangle^\frown tr : v \in T, tr \in \text{traces}(P)\}$

SKIP, RUN, CHOICE

- $\text{traces}(SKIP) = \{\langle\rangle, \langle\checkmark\rangle\}$
- Some more processes (convenient for defining rules):

  $RUN = (x : \Sigma \to RUN) \,\Box\, SKIP$ $\quad\quad$ $\text{traces}(RUN) = TRACE$

  $RUN_A = x : A \to RUN_A$ $\quad\quad$ $\text{traces}(RUN_A) = A^*$

  $RUN_{A^\checkmark} = (x : A \to RUN_{A^\checkmark}) \,\Box\, SKIP$ $\quad\quad$ $\text{traces}(RUN_{A^\checkmark}) = TRACE \cap (A^\checkmark)^*$
- $\text{traces}(P \,\Box\, Q) = \text{traces}(P \sqcap Q) = \text{traces}(P) \cup \text{traces}(Q)$
  - $\text{traces}(\Box_{i \in I} P_i) = \text{traces}(\sqcap_{i \in I} P_i) = \{\langle\rangle\} \cup \bigcup_{i \in I} \text{traces}(P_i)$

Choice Laws

$P \,\Box\, P = P$ ($\Box$ -idem)

$P \,\Box\, (Q \,\Box\, R) = (P \,\Box\, Q) \,\Box\, R$ ($\Box$ -assoc)

$P \,\Box\, Q = Q \,\Box\, P$ ($\Box$ -sym)

$P \,\Box\, STOP = P$ ($\Box$ -unit)

$P \,\Box\, RUN =_T RUN$ ($\Box$ -zero$_T$)

$x : A \to P_1(x) \,\Box\, y : B \to P_2(y) = z : A \cup B \to R(z)$ ($\Box$ -step)

where $R(c) = \begin{cases} P_1(c) & \text{if } c \in A \setminus B \\ P_2(c) & \text{if } c \in B \setminus A \\ P_1(c) \sqcap P_2(c) & \text{if } c \in A \cap B \end{cases}$

$\Box_{i \in \{\}} P_i = STOP$ ($\Box$ -unit)

$P \sqcap P = P$ ($\sqcap$ -idem)

$P \sqcap (Q \sqcap R) = (P \sqcap Q) \sqcap R$ ($\sqcap$ -assoc)

$P \sqcap Q = Q \sqcap P$ ($\sqcap$ -sym)

$P \sqcap Q =_T P \,\Box\, Q$ (choice-equiv$_T$)

Alphabetized Parallel

$$\text{traces}(P_1 \;{}_A\|_B\; P_2) = \{tr \in TRACE : tr \upharpoonright A^{\checkmark} \in \text{traces}(P_1), tr \upharpoonright B^{\checkmark} \in \text{traces}(P_2), \sigma(tr) \subseteq (A \cup B)^{\checkmark}\}$$

$$P \;{}_A\|_A\; P = P \text{ if } \alpha(P) \in A \qquad (\|\text{-idem}_T)$$

$$P \;{}_A\|_B\; Q = Q \;{}_B\|_A\; P \qquad (\|\text{-sym})$$

$$P \;{}_A\|_{B \cup C}\; (Q \;{}_B\|_C\; R) = (P \;{}_A\|_B\; Q) \;{}_{A \cup B}\|_C\; R \qquad (\|\text{-assoc})$$

$$P \;{}_A\|_B\; RUN_{(A \cap B)^{\checkmark}} = P \text{ if } \alpha(P) \subseteq A \qquad (\|\text{-unit})$$

$$C \subseteq A \wedge D \subseteq B \implies \qquad (\|\text{-step})$$

$$(x : C \to P_1(x)) \;{}_A\|_B\; (y : D \to P_2(y)) =$$
$$z : ((C \setminus B) \cup (D \setminus A) \cup (C \cap D)) \to R(z)$$
$$\text{where } R(c) = \begin{cases} P_1(c) \;{}_A\|_B\; (y : D \to P_2(y)) & \text{if } c \in C \setminus B \\ (x : C \to P_1(x)) \;{}_A\|_B\; P_2(c) & \text{if } c \in D \setminus A \\ P_1(c) \;{}_A\|_B\; P_2(c) & \text{if } c \in C \cap D \end{cases}$$

$$SKIP \;{}_A\|_B\; SKIP = SKIP \qquad (\|\text{-term 1})$$

$$(x : C \to P(x)) \;{}_A\|_B\; SKIP = x : C \cap (A \setminus B) \to (P(x) \;{}_A\|_B\; SKIP) \qquad (\|\text{-term 2})$$

$$P \;{}_\Sigma\|_\Sigma\; RUN =_T P \qquad (\|\text{-unit})$$

$$P \;{}_A\|_\Sigma\; STOP = STOP \qquad (\|\text{-zero})$$

INTERLEAVING

- A trace tr is an interleaving of traces tr_1 and tr_2 whenever each occurrence of each event from tr_1 and tr_2 appears exactly once in tr, and all the events from tr_1 and tr_2 occur in the same order in tr:
  - $\langle\rangle$ interleaves tr_1, tr_2 iff $tr_1 = tr_2 = \langle\rangle$
  - $\langle\checkmark\rangle$ interleaves tr_1, tr_2 iff $tr_1 = tr_2 = \langle\checkmark\rangle$
  - if $a \neq \checkmark$ then $\langle a\rangle^\frown tr$ interleaves tr_1, tr_2 iff
    $\text{head}(tr_1) = a \wedge tr$ interleaves $\text{tail}(tr_1), tr_2$
    $\vee \text{head}(tr_2) = a \wedge tr$ interleaves $tr_1, \text{tail}(tr_2)$
- $\text{traces}(P_1 \,|||\, P_2) = \{tr \in TRACE : \exists\, tr_1, tr_2 : tr_1 \in traces(P1), tr_2 \in \text{traces}(P_2), tr$ interleaves $tr_1, tr_2\}$

INTERLEAVING

- A trace tr is an interleaving of traces tr_1 and tr_2 whenever each occurrence of each event from tr_1 and tr_2 appears exactly once in tr, and all the events from tr_1 and tr_2 occur in the same order in tr:
  - $\langle\rangle$ interleaves tr_1, tr_2 iff $tr_1 = tr_2 = \langle\rangle$
  - $\langle\checkmark\rangle$ interleaves tr_1, tr_2 iff $tr_1 = tr_2 = \langle\checkmark\rangle$
  - if $a \neq \checkmark$ then $\langle a\rangle^\frown tr$ interleaves tr_1, tr_2 iff
    $\text{head}(tr_1) = a \wedge tr$ interleaves $\text{tail}(tr_1), tr_2$
    $\vee \text{head}(tr_2) = a \wedge tr$ interleaves $tr_1, \text{tail}(tr_2)$
- $\text{traces}(P_1 \,|||\, P_2) = \{tr \in TRACE : \exists\, tr_1, tr_2 : tr_1 \in traces(P1), tr_2 \in \text{traces}(P_2), tr$ interleaves $tr_1, tr_2\}$

- $\text{traces}(P_1 \underset{A}{\|} P_2)$ + laws $\Rightarrow$ see textbook

Interleaving Laws

$$P \;|||\; Q = Q \;|||\; P \qquad (|||\text{-sym})$$

$$P \;|||\; (Q \;|||\; R) = (P \;|||\; Q) \;|||\; R \qquad (|||\text{-assoc})$$

$$P \;|||\; SKIP = P \qquad (|||\text{-unit})$$

$$P \;|||\; RUN_\Sigma = RUN_\Sigma \qquad (|||\text{-zero})$$

$$(x : C \to P_1(x)) \;|||\; (y : D \to P_2(y)) = z : (C \cup D) \to R(z) \qquad (|||\text{-step})$$

$$\begin{array}{lll} \text{where } R(c) = & P_1(c) \;|||\; (y : D \to P_2(y)) & \text{if } c \in C \setminus D \\ & (x : C \to P_1(x)) \;|||\; P_2(c) & \text{if } c \in D \setminus C \\ & P_1(c) \;|||\; (y : D \to P_2(y)) & \\ & \quad \sqcap\; (x : C \to P_1(x)) \;|||\; P_2(c) & \text{if } c \in C \cap D \end{array}$$

$$SKIP \;|||\; SKIP = SKIP \qquad (|||\text{-term 1})$$

$$(x : C \to P(x)) \;|||\; SKIP = x : C \to (P(x) \;|||\; SKIP) \qquad (|||\text{-term 2})$$

Hiding

- $\text{traces}(P \setminus A) = \{tr \setminus A : tr \in \text{traces}(P)\}$

$$
\begin{array}{ll}
(P \setminus A) \setminus B = P \setminus (A \cup B) & \text{(hide-combine)} \\
(\sqcap_{i \in I} P_i) \setminus A = \sqcap_{i \in I} (P_i \setminus A) & (\sqcap\text{-dist}) \\
STOP \setminus A = STOP & \text{(hide-}STOP\text{)} \\
SKIP \setminus A = SKIP & \text{(hide-term)} \\
(a \to P) \setminus A = \begin{cases} a \to (P \setminus A) & \text{if } a \notin A \\ P \setminus A & \text{if } a \in A \end{cases} & \text{(hide-step 1)} \\
(x : C \to P(x)) \setminus A = x : C \to (P(x) \setminus A) \text{ if } A \cap C = \{\} & \text{(hide-step 2)} \\
(x : C \to P(x)) \setminus A = \sqcap_{x \in C} (P(x) \setminus A) \text{ if } C \subseteq A & \text{(hide-step 3)}
\end{array}
$$

Renaming

- $\text{traces}(f(P)) = \{f(tr) : tr \in \text{traces}(P)\}$
- $\text{traces}(f^{-1}(P)) = \{tr : f(tr) = \text{traces}(P)\}$

$f(x : C \to P(x)) = y : f(C) \to f(P(f^{-1}(y)))$ if f is one-to-one	(f-step 1)
$f(x : C \to P(x)) = y : f(C) \to \sqcap_{x:f(x)=y} f(P(x))$	(f-step 2)
$f(SKIP) = SKIP$	(f-term)
$f^{-1}(x : C \to P(x)) = y : f^{-1}(C) \to f^{-1}(P(f(y))$	(f^{-1}-step)
$f^{-1}(SKIP) = SKIP$	(f^{-1}-term)

Sequential Composition, Interrupt

- $\text{traces}(P_1; P_2) = \{tr \in \text{traces}(P_1), \checkmark \notin \sigma(tr)\}$
  $\cup \{tr_1 \frown tr_2 : tr_1 \frown \langle\checkmark\rangle \in \text{traces}(P_1), tr_2 \in \text{traces}(P_2)\}$

$(P_1; P_2); P_3 = P_1; (P_2; P_3)$	(;-assoc)
$SKIP; P = P$	(;-unit-l)
$P; SKIP =_T P$	(;-unit-r)
$STOP; P = STOP$	(;-zero-l)
$(x : C \to P(x)); P_1 = x : C \to (P(x); P_1)$	(;-step)

- $\text{traces}(P_1 \triangle P_2) =$
  $\text{traces}(P_1) \cup$
  $\{tr_1 \frown tr_2 : tr_1 \in \text{traces}(P_1), \checkmark \notin \sigma(tr_1), tr_2 \in \text{traces}(P_2)$

$(P_1 \triangle P_2) \triangle P_3 = P_1 \triangle (P_2 \triangle P_3)$	($\triangle$-assoc)
$STOP \triangle P = P$	($\triangle$-unit-l)
$P \triangle STOP = P$	($\triangle$-unit-r)
$SKIP \triangle P = SKIP \Box P$	($\triangle$-term)
$(x : C \to P(x)) \triangle P_1 = P_1 \Box x : C \to (P(x) \triangle P_1)$	($\triangle$-step)

Sequential Composition, Interrupt

- $\text{traces}(P_1; P_2) = \{tr \in \text{traces}(P_1), \checkmark \notin \sigma(tr)\}$
  $\cup \{tr_1 \frown tr_2 : tr_1 \frown \langle\checkmark\rangle \in \text{traces}(P_1), tr_2 \in \text{traces}(P_2)\}$

$(P_1; P_2); P_3 = P_1; (P_2; P_3)$	(;-assoc)
$SKIP; P = P$	(;-unit-l)
$P; SKIP =_T P$	(;-unit-r)
$STOP; P = STOP$	(;-zero-l)
$(x : C \to P(x)); P_1 = x : C \to (P(x); P_1)$	(;-step)

- $\text{traces}(P_1 \triangle P_2) =$
  $\text{traces}(P_1) \cup$
  $\{tr_1 \frown tr_2 : tr_1 \in \text{traces}(P_1), \checkmark \notin \sigma(tr_1), tr_2 \in \text{traces}(P_2)$

$(P_1 \triangle P_2) \triangle P_3 = P_1 \triangle (P_2 \triangle P_3)$	($\triangle$-assoc)
$STOP \triangle P = P$	($\triangle$-unit-l)
$P \triangle STOP = P$	($\triangle$-unit-r)
$SKIP \triangle P = SKIP \Box P$	($\triangle$-term)
$(x : C \to P(x)) \triangle P_1 = P_1 \Box x : C \to (P(x) \triangle P_1)$	($\triangle$-step)

- Distributive laws: All CSP operators except recursion distribute over internal choice (and so over external choice in the trace model)

Recursion

- Let $N = F(N)$ be a recursive definition; it follows that $\text{traces}(N) = \text{traces}(F(N))$
- All the CSP operators are monotonic with respect to $\subseteq$: $\text{traces}(P) \subseteq \text{traces}(Q) \implies \text{traces}(F(P)) \subseteq \text{traces}(F(Q))$
- Clearly $\{\langle\rangle\} = \text{traces}(STOP) \subseteq \text{traces}(N)$ and therefore $\text{traces}(F(STOP)) \subseteq \text{traces}(F(N)) = \text{traces}(N)$
- Induction $\Rightarrow \text{traces}(F^n(STOP)) \subseteq \text{traces}(N)$
- Furthermore, all of the processes $F^n(STOP)$ represent finite unwindings of $N = F(N)$ and therefore

$$\text{traces}(N = F(N)) = \bigcup_{n \in \mathbb{N}} \text{traces}(F^n(STOP))$$

  - $\text{traces}(N = F(N))$ is a fixed point for F, which in turn represent a fixed point for N

$$'N = F(N)' \implies N =_T F(N) \qquad \text{(recursion-unwinding}_T\text{)}$$

Unique fixed points

- For the fixed point to be unique the function F that defines the recursion $N = F(N)$ must specify some visible event before reaching the recursive invocation of N
  - The process N must be event guarded in $F(N)$
- A process name N is event guarded in expression P if
  1. N does not appear in P; or
  2.
     1. P does not use the hiding operator; and
     2. Every occurrence of N in P is either
        1. within the scope of a prefix operator; or
        2. contained in the second argument of a sequential composition whose first argument does not terminate immediately

$$F(Y) \text{ guarded} \wedge (F(P_1) =_T P_1) \wedge (F(P_2) =_T P_2) \implies P_1 =_T P_2 \qquad (\text{UFP}_T)$$

Unique fixed points

- For the fixed point to be unique the function F that defines the recursion $N = F(N)$ must specify some visible event before reaching the recursive invocation of N
  - The process N must be event guarded in $F(N)$
- A process name N is event guarded in expression P if
  1. N does not appear in P; or
  2.
     1. P does not use the hiding operator; and
     2. Every occurrence of N in P is either
        1. within the scope of a prefix operator; or
        2. contained in the second argument of a sequential composition whose first argument does not terminate immediately

$$F(Y) \text{ guarded} \wedge (F(P_1) =_T P_1) \wedge (F(P_2) =_T P_2) \implies P_1 =_T P_2 \qquad (\text{UFP}_T)$$

- Mutual recursion is considered in the same way, but this time with a system of equations (rather than a single one)

Model-Based Testing

- Testing analyzes processes based on how they behave in particular contexts
  - Contexts given by tests $\Rightarrow$ special CSP processes
  - Two processes are equivalent if no test can tell them apart
- For $\omega \notin \Sigma^{\checkmark}$ define the process *SUCCESS*:

$$\frac{}{SUCCESS \xrightarrow{\omega} STOP}$$

- The application of test T on process P is the process

$$(P \underset{\Sigma}{\parallel} T) \setminus \Sigma$$

  - The only event that may appear in the trace of this process is ω
  - An individual run is successful if its trace is $\langle \omega \rangle$ and unsuccessful if its trace is $\langle \rangle$
  - The application is deemed successful overall based on the success or failure of all the individual runs
  - There are multiple definitions of success or failure (of the application)

May Testing

- Traces are denotational (we know what a process does, but not how it does it)
- An equivalent operational definition is possible through may testing:

$$P \textbf{ may } T = (P \underset{\Sigma}{\|} T) \setminus \Sigma \stackrel{\langle\omega\rangle}{\Longrightarrow}$$

- If no tests can distinguish between two processes then they are equivalent:

$$P_1 \equiv_{may} P_2 = \forall\, T : P_1 \textbf{ may } T \text{ iff } P_2 \textbf{ may } T$$

- A refinement (or implementation) relation can also be defined:

$$P_1 \sqsubseteq_{may} P_2 = \forall\, T : P_2 \textbf{ may } T \implies P_1 \textbf{ may } T$$

Theorem

$P_1 \equiv_{may} P_2$ *iff* $P_1 =_T P_2$ *(that is,* $traces(P_1) = traces(P_2)$*)*
$P_1 \sqsubseteq_{may} P_2$ *iff* $traces(P_2) \subseteq traces(P_1)$