

# 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  $f : A \rightarrow B$  to  $f : A^* \rightarrow B^*$
- $\sigma(xs) \Rightarrow$  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  $\Sigma^\vee$ )
- The set of all the possible traces:  
 $TRACE = \{tr : \sigma(tr) \subseteq \Sigma^\vee, \#tr \in \mathbb{N}, \checkmark \notin \sigma(\text{init}(tr))\}$
- 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  $f : \Sigma^\vee \rightarrow \Sigma^\vee$  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
  - $P \xrightarrow{tr} P'$  means that there is a run of  $P$  with trace  $tr$  and ending in  $P'$
  - $P \xrightarrow{tr}$  is shorthand for  $\exists P' : P \xrightarrow{tr} P'$
  - $\text{traces}(P) = \{tr \in TRACE : P \xrightarrow{tr}\}$

## TRACE SEMANTICS, PREFIX



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



- $\text{traces}(\text{SKIP}) = \{\langle \rangle, \langle \checkmark \rangle\}$
- Some more processes (convenient for defining rules):
  - $\text{RUN} = (x : \Sigma \rightarrow \text{RUN}) \square \text{SKIP} \quad \text{traces}(\text{RUN}) = \text{TRACE}$
  - $\text{RUN}_A = x : A \rightarrow \text{RUN}_A \quad \text{traces}(\text{RUN}_A) = A^*$
  - $\text{RUN}_{A^\vee} = (x : A \rightarrow \text{RUN}_{A^\vee}) \square \text{SKIP} \quad \text{traces}(\text{RUN}_{A^\vee}) = \text{TRACE} \cap (A^\vee)^*$
- $\text{traces}(P \square Q) = \text{traces}(P \sqcap Q) = \text{traces}(P) \cup \text{traces}(Q)$ 
  - $\text{traces}(\square_{i \in I} P_i) = \text{traces}(\sqcap_{i \in I} P_i) = \{\langle \rangle\} \cup \bigcup_{i \in I} \text{traces}(P_i)$



$$\begin{aligned}
 P \square P &= P && (\square \text{-idem}) \\
 P \square (Q \square R) &= (P \square Q) \square R && (\square \text{-assoc}) \\
 P \square Q &= Q \square P && (\square \text{-sym}) \\
 P \square \text{STOP} &= P && (\square \text{-unit}) \\
 P \square \text{RUN} &=_{\mathcal{T}} \text{RUN} && (\square \text{-zero}_{\mathcal{T}}) \\
 x : A \rightarrow P_1(x) \square y : B \rightarrow P_2(y) &= z : A \cup B \rightarrow R(z) && (\square \text{-step}) \\
 \text{where } R(c) &= \begin{cases} P_1(c) & \text{if } c \in A \setminus B \\ P_2(c) & \text{if } c \in B \setminus A \\ P_1(c) \sqcap P_2(c) & \text{if } c \in A \cap B \end{cases} \\
 \square_{i \in \{ \}} P_i &= \text{STOP} && (\square \text{-unit})
 \end{aligned}$$

$$\begin{aligned}
 P \sqcap P &= P && (\sqcap \text{-idem}) \\
 P \sqcap (Q \sqcap R) &= (P \sqcap Q) \sqcap R && (\sqcap \text{-assoc}) \\
 P \sqcap Q &= Q \sqcap P && (\sqcap \text{-sym}) \\
 P \sqcap Q &=_{\mathcal{T}} P \square Q && (\text{choice-equiv}_{\mathcal{T}})
 \end{aligned}$$



$$\text{traces}(P_1 \parallel_B P_2) = \{tr \in \text{TRACE} : \begin{aligned} &tr \upharpoonright A^\vee \in \text{traces}(P_1), \\ &tr \upharpoonright B^\vee \in \text{traces}(P_2), \\ &\sigma(tr) \subseteq (A \cup B)^\vee \}
 \end{aligned}$$

$$\begin{aligned}
 P \parallel_A P &= P \text{ if } \alpha(P) \in A && (\parallel \text{-idem}_{\mathcal{T}}) \\
 P \parallel_B Q &= Q \parallel_A P && (\parallel \text{-sym}) \\
 P \parallel_{B \cup C} (Q \parallel_C R) &= (P \parallel_B Q) \parallel_{A \cup B} C R && (\parallel \text{-assoc}) \\
 P \parallel_B \text{RUN}_{(A \cap B)^\vee} &= P \text{ if } \alpha(P) \subseteq A && (\parallel \text{-unit}) \\
 C \subseteq A \wedge D \subseteq B &\implies && (\parallel \text{-step}) \\
 (x : C \rightarrow P_1(x)) \parallel_B (y : D \rightarrow P_2(y)) &= \\
 z : ((C \setminus B) \cup (D \setminus A) \cup (C \cap D)) &\rightarrow R(z) \\
 \text{where } R(c) &= \begin{cases} P_1(c) \parallel_B (y : D \rightarrow P_2(y)) & \text{if } c \in C \setminus B \\ (x : C \rightarrow P_1(x)) \parallel_B P_2(c) & \text{if } c \in D \setminus A \\ P_1(c) \parallel_B P_2(c) & \text{if } c \in C \cap D \end{cases} \\
 \text{SKIP} \parallel_B \text{SKIP} &= \text{SKIP} && (\parallel \text{-term 1}) \\
 (x : C \rightarrow P(x)) \parallel_B \text{SKIP} &= x : C \cap (A \setminus B) \rightarrow (P(x) \parallel_B \text{SKIP}) && (\parallel \text{-term 2}) \\
 P \parallel_{\Sigma} \parallel_{\Sigma} \text{RUN} &=_{\mathcal{T}} P && (\parallel \text{-unit}) \\
 P \parallel_{\Sigma} \text{STOP} &= \text{STOP} && (\parallel \text{-zero})
 \end{aligned}$$



- 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  $tr_1, \text{tail}(tr_2)$
- $\text{traces}(P_1 \parallel \parallel P_2) = \{tr \in \text{TRACE} : \exists tr_1, tr_2 : tr_1 \in \text{traces}(P_1), tr_2 \in \text{traces}(P_2), tr \text{ interleaves } tr_1, tr_2\}$
- $\text{traces}(P_1 \parallel_A P_2) + \text{laws} \Rightarrow$  [see textbook](#)



$$\begin{array}{ll}
 P \parallel Q = Q \parallel P & (\parallel\text{-sym}) \\
 P \parallel (Q \parallel R) = (P \parallel Q) \parallel R & (\parallel\text{-assoc}) \\
 P \parallel \text{SKIP} = P & (\parallel\text{-unit}) \\
 P \parallel \text{RUN}_{\Sigma} = \text{RUN}_{\Sigma} & (\parallel\text{-zero}) \\
 (x : C \rightarrow P_1(x)) \parallel (y : D \rightarrow P_2(y)) = z : (C \cup D) \rightarrow R(z) & (\parallel\text{-step}) \\
 \text{where } R(c) = \begin{array}{ll} P_1(c) \parallel (y : D \rightarrow P_2(y)) & \text{if } c \in C \setminus D \\ (x : C \rightarrow P_1(x)) \parallel P_2(c) & \text{if } c \in D \setminus C \\ P_1(c) \parallel (y : D \rightarrow P_2(y)) & \\ \sqcap (x : C \rightarrow P_1(x)) \parallel P_2(c) & \text{if } c \in C \cap D \end{array} \\
 \text{SKIP} \parallel \text{SKIP} = \text{SKIP} & (\parallel\text{-term 1}) \\
 (x : C \rightarrow P(x)) \parallel \text{SKIP} = x : C \rightarrow (P(x) \parallel \text{SKIP}) & (\parallel\text{-term 2})
 \end{array}$$



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

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



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

$$\begin{array}{ll}
 f(x : C \rightarrow P(x)) = y : f(C) \rightarrow f(P(f^{-1}(y))) \text{ if } f \text{ is one-to-one} & (f\text{-step 1}) \\
 f(x : C \rightarrow P(x)) = y : f(C) \rightarrow \sqcap_{x: f(x)=y} f(P(x)) & (f\text{-step 2}) \\
 f(\text{SKIP}) = \text{SKIP} & (f\text{-term}) \\
 f^{-1}(x : C \rightarrow P(x)) = y : f^{-1}(C) \rightarrow f^{-1}(P(f(y))) & (f^{-1}\text{-step}) \\
 f^{-1}(\text{SKIP}) = \text{SKIP} & (f^{-1}\text{-term})
 \end{array}$$



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

$$\begin{array}{ll}
 (P_1; P_2); P_3 = P_1; (P_2; P_3) & (;\text{-assoc}) \\
 \text{SKIP}; P = P & (;\text{-unit-l}) \\
 P; \text{SKIP} =_T P & (;\text{-unit-r}) \\
 \text{STOP}; P = \text{STOP} & (;\text{-zero-l}) \\
 (x : C \rightarrow P(x)); P_1 = x : C \rightarrow (P(x); P_1) & (;\text{-step})
 \end{array}$$

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

$$\begin{array}{ll}
 (P_1 \triangle P_2) \triangle P_3 = P_1 \triangle (P_2 \triangle P_3) & (\triangle\text{-assoc}) \\
 \text{STOP} \triangle P = P & (\triangle\text{-unit-l}) \\
 P \triangle \text{STOP} = P & (\triangle\text{-unit-r}) \\
 \text{SKIP} \triangle P = \text{SKIP} \sqcap P & (\triangle\text{-term}) \\
 (x : C \rightarrow P(x)) \triangle P_1 = P_1 \sqcap x : C \rightarrow (P(x) \triangle P_1) & (\triangle\text{-step})
 \end{array}$$

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



- 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  $\implies \text{traces}(F^n(STOP)) \subseteq \text{traces}(N)$
- 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  $N$

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



- 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
  - $N$  does not appear in  $P$ ; or
  - $P$  does not use the hiding operator; and
    - Every occurrence of  $N$  in  $P$  is either
      - within the scope of a prefix operator; or
      - contained in the second argument of a sequential composition whose first argument does not terminate immediately

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

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



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

$$\overline{\text{SUCCESS} \xrightarrow{\omega} \text{STOP}}$$

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

$$(P \parallel_{\Sigma} T) \setminus \Sigma$$

- The only event that may appear in the trace of this process is  $\omega$
- 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)



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

$$P \text{ may } T = (P \parallel_{\Sigma} T) \setminus \Sigma \xrightarrow{\langle \omega \rangle}$$

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

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

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

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

Theorem

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