CS 515: Stable Failures Semantics

Stefan D. Bruda

Winter 2019

Stable Failures

- While safety properties ("something bad will never happen") can be expressed with traces, traces cannot express liveness properties
  - A liveness property guarantees a certain behaviour ("something good will happen")
  - Specifies what a process is prepared to do
- A stable process is a process P that can make no internal progress: $P \downarrow = \neg(P \xrightarrow{\tau})$ (we also say that P converges)
  - Contrast in passing with a divergent process: $P \uparrow = \exists (P_i)_{i \in \mathbb{N}} : P = P_0 \wedge \forall i \in \mathbb{N} : P_i \xrightarrow{\tau} P_{i+1}$
- A process refuses a set of actions X whenever no action from X is available from P: $P \text{ ref } X$ iff $\exists P' : P \stackrel{\langle\rangle}{\Longrightarrow} P' \wedge P' \downarrow \wedge \forall a \in X : \neg(P' \xrightarrow{a})$
- The fact that a process does not refuse an event is recorded in a trace information (as before)
- In all a stable failure of process P is an observation (tr, X) such that $P \stackrel{tr}{\Longrightarrow} P' \wedge P' \downarrow \wedge P' \text{ ref } X$

Semantics of Stable Failures

- The semantic model of stable failures is less abstract than traces
  - The approach to specification and verification is however the same
- Six consistency conditions for some T and SF sets of traces and stable failures corresponding to the same process, respectively:

1. $\langle\rangle \in T$
2. $tr_1 \leq tr_2 \wedge tr_2 \in T \implies tr_1 \in T$
3. $(tr, X) \in SF \implies tr \in T$
4. $(tr, X) \in SF \wedge X' \subseteq X \implies (tr, X') \in SF$
5. $(tr, X) \in SF \wedge \forall a \in X' : tr^\frown\langle a\rangle \notin T \implies (tr, X \cup X') \in SF$
6. $\forall X \subseteq \Sigma : tr^\frown\langle\checkmark\rangle \in T \implies (tr^\frown\langle\checkmark\rangle, X) \in SF$

Semantics of Stable Failures

- The semantic model of stable failures is less abstract than traces
  - The approach to specification and verification is however the same
- Six consistency conditions for some T and SF sets of traces and stable failures corresponding to the same process, respectively:

1. $\langle\rangle \in T \Rightarrow$ from trace semantics
2. $tr_1 \leq tr_2 \wedge tr_2 \in T \implies tr_1 \in T \Rightarrow$ closure under prefix, from trace semantics
3. $(tr, X) \in SF \implies tr \in T$
4. $(tr, X) \in SF \wedge X' \subseteq X \implies (tr, X') \in SF$
5. $(tr, X) \in SF \wedge \forall a \in X' : tr^\frown\langle a\rangle \notin T \implies (tr, X \cup X') \in SF$
6. $\forall X \subseteq \Sigma : tr^\frown\langle\checkmark\rangle \in T \implies (tr^\frown\langle\checkmark\rangle, X) \in SF$

Semantics of Stable Failures

- The semantic model of stable failures is less abstract than traces
  - The approach to specification and verification is however the same
- Six consistency conditions for some T and SF sets of traces and stable failures corresponding to the same process, respectively:

1. $\langle\rangle \in T \Rightarrow$ from trace semantics
2. $tr_1 \leq tr_2 \wedge tr_2 \in T \implies tr_1 \in T \Rightarrow$ closure under prefix, from trace semantics
3. $(tr, X) \in SF \implies tr \in T \Rightarrow$ first component of a stable failure is a trace (of the same process)
4. $(tr, X) \in SF \wedge X' \subseteq X \implies (tr, X') \in SF \Rightarrow$ any subset of a refused set is also refused
5. $(tr, X) \in SF \wedge \forall a \in X' : tr^\frown\langle a\rangle \notin T \implies (tr, X \cup X') \in SF \Rightarrow$ a process can either perform an action of refuse it (never both)
6. $\forall X \subseteq \Sigma : tr^\frown\langle\checkmark\rangle \in T \implies (tr^\frown\langle\checkmark\rangle, X) \in SF \Rightarrow$ a terminated process refuses anything

Process Semantics

$\mathcal{SF}[\![P]\!]$ denotes the stable failures of process P

- $\mathcal{SF}[\![STOP]\!] = \{(\langle\rangle, X) : X \subseteq \Sigma^{\checkmark}\}$
- $\mathcal{SF}[\![a \to P]\!] = \{(\langle\rangle, X) : a \notin X\} \cup \{(\langle a\rangle^\frown tr, X) : (tr, X) \in \mathcal{SF}[\![P]\!]\}$
- $\mathcal{SF}[\![x : A \to P]\!] = \{(\langle\rangle, X) : A \cap X \neq \{\}\} \cup \{(\langle a\rangle^\frown tr, X) : a \in A \wedge (tr, X) \in \mathcal{SF}[\![P]\!]\}$
- $\mathcal{SF}[\![SKIP]\!] = \{(\langle\rangle, X) : \checkmark \notin X\} \cup \{(\langle\checkmark\rangle, X) : X \subseteq \Sigma^{\checkmark}\}$
- traces(DIV) $= \{\langle\rangle\}$, $\mathcal{SF}[\![DIV]\!] = \{\}$
- traces($CHAOS$) $= TRACE$, $\mathcal{SF}[\![CHAOS]\!] = TRACE \times 2^{\Sigma^{\checkmark}}$
  - traces($CHAOS_A$) $= \{tr : \sigma(tr) \subseteq A\}$, $\mathcal{SF}[\![CHAOS_A]\!] = \{(tr, X) : \sigma(tr) \subseteq A\}$
- $\mathcal{SF}[\![RUN]\!] = \{(tr, X) : X = \{\} \vee \checkmark \in \sigma(tr)\}$
  - $\mathcal{SF}[\![RUN_A]\!] = \{(tr, X) : \sigma(tr) \subseteq A \wedge (X \cap A = \{\} \vee \checkmark \in \sigma(tr))\}$
- $\mathcal{SF}[\![P \,\Box\, Q]\!] = \{(\langle\rangle, X) : (\langle\rangle, X) \in \mathcal{SF}[\![P]\!] \cap \mathcal{SF}[\![Q]\!]\} \quad \cup$
  $\{(tr, X) : tr \neq \langle\rangle \wedge (tr, X) \in \mathcal{SF}[\![P]\!] \cup \mathcal{SF}[\![Q]\!]\}$
  - Idempotence, associativity, commutativity continue to hold
  - $STOP$ continues to be a unit, but the zero is now $RUN \,\Box\, DIV$:

$$P \,\Box\, (RUN \,\Box\, DIV) =_{SF} RUN \,\Box\, DIV \qquad (\,\Box\,_{SF}\text{-zero})$$

Process Semantics (cont'd)

- $\mathcal{SF}[\![P \sqcap Q]\!] = \mathcal{SF}[\![P]\!] \cup \mathcal{SF}[\![Q]\!]$

$$P \sqcap (Q \,\Box\, R) =_{SF} (P \sqcap Q) \,\Box\, (P \sqcap R) \qquad (\,\Box \text{ - } \sqcap \text{-dist})$$

- $\mathcal{SF}[\![P \;{}_A\|_B\; Q]\!] = \{(tr, X) : \exists\, X_P, X_Q \in 2^{\Sigma^\checkmark} :$
  $X \cap (A \cup B)^\checkmark = (X_P \cap A^\checkmark) \cup (X_Q \cap B^\checkmark) \wedge$
  $(tr \upharpoonright A, X_P) \in \mathcal{SF}[\![P]\!] \wedge (tr \upharpoonright B^\checkmark, X_Q) \in \mathcal{SF}[\![Q]\!] \wedge$
  $\sigma(tr) \subseteq (A \cup B)^\checkmark\}$
  - All laws from the trace semantics hold, except $\|$ -idem
- $\mathcal{SF}[\![P \;|||\; Q]\!] = \{(tr, X_P \cup X_Q) : \exists\, tr_P, tr_Q : tr$ interleaves $tr_P, tr_Q \wedge$
  $X_P \upharpoonright \Sigma = X_Q \upharpoonright \Sigma \wedge (tr_P, X_P) \in \mathcal{SF}[\![P]\!] \wedge (tr_Q, X_Q) \in \mathcal{SF}[\![Q]\!]\}$
  - All laws from the trace semantics continue to apply, except $|||$ -zero which becomes:

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

- $\mathcal{SF}[\![P \setminus A]\!] = \{(tr \setminus A, X) : (tr, X \cup A) \in \mathcal{SF}[\![P]\!]\}$
- $\mathcal{SF}[\![f(P)]\!] = \{(f(tr), X) : (tr, f^{-1}(X)) \in \mathcal{SF}[\![P]\!]\}$
- $\mathcal{SF}[\![f^{-1}(P)]\!] = \{(tr, X) : (f(tr), f(X)) \in \mathcal{SF}[\![P]\!]\}$

Process Semantics (cont'd)

- $\mathcal{SF}[\![P; Q]\!] = \{(tr, X) : (tr, X \cup \{\checkmark\}) \in \mathcal{SF}[\![P]\!]\} \quad \cup$
  $\{(tr_1 \frown tr_2, X) : tr_1 \frown \langle\checkmark\rangle \in \text{traces}(P) \wedge (tr_2, X) \in \mathcal{SF}[\![Q]\!]\}$
- $\mathcal{SF}[\![P \triangle Q]\!] = \{(tr, X) : (tr, X) \in \mathcal{SF}[\![P]\!] \wedge$
  $(\checkmark \in \sigma(tr) \vee (\langle\rangle, X) \in \mathcal{SF}[\![Q]\!])\} \quad \cup$
  $\{(tr_1 \frown tr_2, X) : tr_1 \in \text{traces} P \wedge \checkmark \notin \sigma(tr_1) \wedge$
  $(tr_2, X) \in \mathcal{SF}[\![Q]\!] \wedge tr_2 \neq \langle\rangle\}$
- Recursion needs a different SOS rule for the stable failure model to make unguarded expressions unstable:

$$\frac{}{N \xrightarrow{\tau} P} \; [N = P]$$

- The minimal process for stable failures is *DIV*, and so

$$\mathcal{SF}[\![N = F(N)]\!] = \bigcup_{n \in \mathbb{N}} \mathcal{SF}[\![F^n(DIV)]\!]$$

  - The laws of resursion unwinding and unique fixed points continue to hold

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