CS 515: Specification and Verification with Stable Failures

Stefan D. Bruda

Winter 2019

Property-Oriented Specifications

- The specification is a pair of predicates $(S_T(tr), S_{SF}(tr, X))$, one on traces and the other on stable failures

$$P \textbf{ sat } S(tr) = \forall\, tr \in \text{traces}(P) : S_T(tr) \wedge \forall\, (tr, X) \in \mathcal{SF}[\![P]\!] : S_{SF}(tr, X)$$

- Safety specifications ("nothing bad will happen") still captured by the predicate on traces $S_T(tr)$
- Liveness specifications ("something good will happen") can now be captured by the other predicate $S_{SF}(tr, X)$
  - Liveness is expressed in terms of the ability of a process to engage in actions
  - That is, the process is guaranteed to offer (that is, not refuse after stabilizing) certain action at a certain point of the execution
  - The "certain point in the execution" is given by the trace, the "guarantee to offer certain action" by the refusal set
- We will not concern ourselves here with the trace properties $S_T(tr)$ (we already saw those), but only with the stable failure properties $S_{SF}(tr, X)$

Verification Rules for Basic Processes

- *STOP*: $\dfrac{}{STOP\ \mathbf{sat}\ tr = \langle\rangle}$
- *SKIP*: $\dfrac{}{SKIP\ \mathbf{sat}\ tr = \langle\rangle \wedge \checkmark \in X \vee tr = \langle\checkmark\rangle}$
- *DIV* satisfies everything, including *false*! $\dfrac{}{DIV\ \mathbf{sat}\ S_{SF}(tr, X)}$
- CHAOS only satisfies *true*: $\dfrac{}{CHAOS\ \mathbf{sat}\ true(tr, X)}$
- *RUN*: $\dfrac{}{RUN\ \mathbf{sat}\ \checkmark \notin \sigma(tr) \implies X = \{\}}$

Prefix and Choice

- Prefix:
$$\frac{P \textbf{ sat } S_{SF}(tr, X)}{\begin{array}{l} a \to P \textbf{ sat } tr = \langle\rangle \wedge a \notin X \vee \text{head}(tr) = a \wedge S_{SF}(\text{tail}(tr), X) \\ \forall\, a \in A : P(a) \textbf{ sat } S_a(tr, X) \end{array}}$$
$$\frac{}{\begin{array}{l} x : A \to P(x) \textbf{ sat } tr = \langle\rangle \wedge A \cap X = \{\} \vee \\ \exists\, a \in A : \text{head}(tr) = a \wedge S_a(\text{tail}(tr), X) \end{array}}$$
- Input and output:
$$\frac{P \textbf{ sat } S_{SF}(tr, X)}{\begin{array}{l} c!v \to P \textbf{ sat } tr = \langle\rangle \wedge c.v \notin X \vee \text{head}(tr) = c.v \wedge S_{SF}(\text{tail}(tr), X) \\ \forall\, v \in T : P(v) \textbf{ sat } S_v(tr, X) \end{array}}$$
$$\frac{}{\begin{array}{l} c?x : T \to P(x) \textbf{ sat } tr = \langle\rangle \wedge in.T \cap X = \{\} \vee \\ \exists\, v \in T : \text{head}(tr) = c.v \wedge S_v(\text{tail}(tr), X) \end{array}}$$
- External choice:
$$\frac{\begin{array}{c} P_1 \textbf{ sat } S_1(tr, X) \\ P_2 \textbf{ sat } S_2(tr, X) \end{array}}{\begin{array}{l} P_1 \,\Box\, P_2 \textbf{ sat } (tr = \langle\rangle \implies S_1(tr, X) \wedge S_2(tr, X)) \wedge \\ \qquad\qquad (tr \neq \langle\rangle \implies (S_1(tr, X) \vee S_2(tr, X))) \end{array}}$$
- Internal choice:
$$\frac{\begin{array}{c} P_1 \textbf{ sat } S_1(tr, X) \\ P_2 \textbf{ sat } S_2(tr, X) \end{array}}{P_1 \sqcap P_2 \textbf{ sat } S_1(tr, X) \vee S_2(tr, X)}$$

Parallel, Interleaving, Hiding

- Alphabetized parallel:

$$\frac{P_1 \textbf{ sat } S_1(tr, X) \qquad P_2 \textbf{ sat } S_2(tr, X)}{P_1 \ {}_{A_1}\|_{A_2} \ P_2 \textbf{ sat } \exists X_1, X_2 : S_1(tr \upharpoonright A_1^{\checkmark}, X_1) \wedge S_2(tr \upharpoonright A_2^{\checkmark}, X_2) \wedge \sigma(tr) \subseteq (A_1 \cup A_2)^{\checkmark} \wedge X \cap (A_1 \cup A_2)^{\checkmark} = (X \cap A_1^{\checkmark}) \cup (X_2 \cap A_2^{\checkmark})}$$

- Interleaving:

$$\frac{P_1 \textbf{ sat } S_1(tr, X) \qquad P_2 \textbf{ sat } S_2(tr, X)}{P_1 \ ||| \ P_2 \textbf{ sat } \exists tr_1, tr_2, X_1, X_2 : S_1(tr_1, X_1) \wedge S_2(tr_2, X_2) \wedge tr \text{ interleaves } tr_1, tr_2 \wedge X_1 \cup X_2 = X \wedge X_1 \setminus \{\checkmark\} = X_2 \setminus \{\checkmark\}}$$

- Hiding:

$$\frac{P \textbf{ sat } S(tr, X)}{P \setminus A \textbf{ sat } \exists tr_1 : S(tr_1, X \cup A) \wedge tr = tr_1 \setminus A}$$

$$\frac{P \textbf{ sat } S(tr, X)}{P \setminus A \textbf{ sat } S(tr, X)} \ \left[\forall tr, X : S(tr, X) \text{ iff } S(tr \setminus A, X \setminus A)\right]$$

Renaming, Sequential, Interrupt

- Renaming: $\dfrac{P \textbf{ sat } S(tr, X)}{\exists\, tr_1 : S(tr_1, f^{-1}(X)) \wedge f(tr_1) = tr}$

$$\frac{P \textbf{ sat } S(tr, X)}{f^{-1}(P) \textbf{ sat } S(f(tr), f(X))}$$

- Sequential composition:

$$\frac{\begin{array}{c} P_1 \textbf{ sat } (S_t(tr), S_1(tr, X)) \\ P_2 \textbf{ sat } S_2(tr, X) \end{array}}{\begin{array}{l} P_1; P_2 \textbf{ sat } \checkmark \notin \sigma(tr) \wedge S_1(tr, X \cup \{\checkmark\}) \vee \\ \qquad \exists\, tr_1, tr_2 : tr = tr_1 \frown tr_2 \wedge S_T(tr_1 \frown \langle \checkmark \rangle) \wedge S_2(tr_2, X) \end{array}}$$

- Interrupt:

$$\frac{\begin{array}{c} P_1 \textbf{ sat } (S_t(tr), S_1(tr, X)) \\ P_2 \textbf{ sat } S_2(tr, X) \end{array}}{\begin{array}{l} P_1 \triangle P_2 \textbf{ sat } S_1(tr, X) \wedge (S_2(\langle\rangle, X) \vee \checkmark \in \sigma(tr)) \vee \\ \qquad \exists\, tr_1, tr_2 : tr = tr_1 \frown tr_2 \wedge \checkmark \notin \sigma(tr_1) \wedge S_T(tr_1) \wedge S_2(tr_2, X) \end{array}}$$

Recursion

$$\frac{\forall\, Y : Y \textbf{ sat } S_{SF}(tr, X) \implies F(Y) \textbf{ sat } S_{SF}(tr, X)}{N \textbf{ sat } S_{SF}(tr, X)} \; [N = F(N)]$$

- No basis needed, because all specifications are satisfied by *DIV*

Process-Oriented Specification

- Refinement relation based on set inclusion: $(T_1, SF_1) \sqsubseteq_{SF} (T_2, SF_2)$ iff $T_2 \subseteq T_1 \wedge SF_2 \subseteq SF_1 \Rightarrow$ obvious definition for $P_1 \sqsubseteq_{SF} P_2$
  - The stable failures and traces of P_2 do not introduce any new behaviour to P_1
  - Alternative definition: $P_1 \sqsubseteq_{SF} P_2$ iff $P_1 = P_1 \sqcap P_2$
- Same as trace refinement: An implementation I meets a specification S ($S \sqsubseteq_{SF} I$) whenever all the possible behaviours of I are allowed by S
  - The difference being that now behaviours are defined by stable failures

Process-Oriented Specification

- Refinement relation based on set inclusion: $(T_1, SF_1) \sqsubseteq_{SF} (T_2, SF_2)$ iff $T_2 \subseteq T_1 \wedge SF_2 \subseteq SF_1 \Rightarrow$ obvious definition for $P_1 \sqsubseteq_{SF} P_2$
  - The stable failures and traces of P_2 do not introduce any new behaviour to P_1
  - Alternative definition: $P_1 \sqsubseteq_{SF} P_2$ iff $P_1 = P_1 \sqcap P_2$
- Same as trace refinement: An implementation I meets a specification S ($S \sqsubseteq_{SF} I$) whenever all the possible behaviours of I are allowed by S
  - The difference being that now behaviours are defined by stable failures
- Stable failure refinement also captured by failure trace testing
  - Tests as before, plus a special action θ (deadlock) with the lowest priority:

$$\frac{P \xrightarrow{\tau} P'}{P \,||\, T \xrightarrow{\tau} P' \,||\, T} \qquad \frac{T \xrightarrow{\tau} t'}{P \,||\, T \xrightarrow{\tau} P' \,||\, T} \qquad \frac{\begin{array}{c} P \xrightarrow{a} P' \\ T \xrightarrow{a} T' \end{array}}{P \,||\, T \xrightarrow{a} P' \,||\, T'}\, a \in \Sigma$$

$$\frac{}{P \,||\, SUCCESS \xrightarrow{\omega} STOP}$$

$$\frac{T \xrightarrow{\theta} T'}{P \,||\, T \xrightarrow{\theta} P \,||\, T'}\, \neg(\exists x \in \Sigma \cup \{\tau, \omega\} : P \,||\, T \xrightarrow{x})$$

Failure Traces

- Let $\text{init}(P) = \{a \in \Sigma : P \stackrel{a}{\Longrightarrow}\}$
- Let $P \stackrel{\varepsilon}{\Longrightarrow} P_0 \stackrel{a_1}{\Longrightarrow} P_1 \stackrel{a_2}{\Longrightarrow} \cdots \stackrel{a_n}{\Longrightarrow} P_n$
- Then $f = A_0 a_1 A_1 a_2 A_2 \ldots a_n A_n$, $n \geq 0$, $a_i \in \Sigma^*$, $A_i \subseteq \Sigma$ is a failure trace of P if:
  - If $\neg(p_i \stackrel{\tau}{\longrightarrow})$, then $A_i \subseteq (A \setminus \text{init}(p_i))$, and
  - If $p_i \stackrel{\tau}{\longrightarrow}$ then $A_i = \{\}$
- We obtain a failure trace of P by taking a trace of P and inserting refusal sets after stable states

Lemma

Failure traces and stable failures describe the same behaviour

- Every stable failure is a failure trace
- Every failure trace $f = A_0 a_1 A_1 a_2 A_2 \ldots a_n A_n$ is equivalent with the set of stable failures $\{(\langle\rangle, A_0)\} \cup \{(\langle a_1, \ldots, a_i\rangle, A_i) : i \geq 0\}$

Sequential Failure Trace Tests

- The set of sequential failure trace tests $\mathcal{ST}$:
  - $SUCCESS \in \mathcal{ST}$
  - If $T \in \mathcal{ST}$ then $a \rightarrow T \in \mathcal{ST}$ for any $a \in \Sigma$
  - If $t \in \mathcal{ST}$ then $(\ \Box_{a \in A'} a \rightarrow STOP) \ \Box\ \theta \rightarrow T \in \mathcal{ST}$ for any $A' \subseteq A$

Lemma

There exists a bijection between failure traces and sequential tests

- For a sequential test t the failure trace $\text{ftr}(t)$ is:
  $\text{ftr}(SUCCESS) = \{\}$
  $\text{ftr}(a \rightarrow T') = a\ \text{ftr}(T')$
  $\text{ftr}((\ \Box_{a \in A'} a \rightarrow STOP) \ \Box\ \theta \rightarrow T') = A'\ \text{ftr}(T')$
- Conversely, for a failure trace f the sequential test $\text{seqt}(f)$ is:
  $\text{seqt}(\{\}) = SUCCESS$
  $\text{seqt}(af) = a \rightarrow \text{seqt}(f)$
  $\text{seqt}(Af) = (\ \Box_{a \in A} a \rightarrow STOP) \ \Box\ \theta \rightarrow \text{seqt}(f)$
- Note that $\text{ftr}(\text{seqt}(f)) = f$ and $\text{seqt}(\text{ftr}(t)) = t$

Failure Trace Testing vs Stable Failures

Lemma

*For any failure trace test T there exists a set $T_s \subseteq \mathcal{ST}$ such that P **may** T if and only if $\exists T' \in T_s : P$ **may** T'*

- $T_s = \{\text{seqt}(f) : f \text{ a failure trace of } T\}$

Failure Trace Testing vs Stable Failures

Lemma

*For any failure trace test T there exists a set $T_s \subseteq \mathcal{ST}$ such that P **may** T if and only if $\exists T' \in T_s : P$ **may** T'*

- $T_s = \{\text{seqt}(f) : f \text{ a failure trace of } T\}$

Theorem

*$P \sqsubseteq_{SF} Q$ iff P **may** $T \implies Q$ **may** T for all failure trace tests T*

- We go from stable failures to failure trace tests (and the other way around) via failure traces and sequential tests