« Collecting Semantics »

Patrick Cousot
Jerome C. Hunsaker Visiting Professor
Massachusetts Institute of Technology
Department of Aeronautics and Astronautics

cousot@mit.edu
www.mit.edu/~cousot

Course 16.399: "Abstract interpretation"

http://web.mit.edu/afs/athena.mit.edu/course/16/16.399/www/

Collecting Semantics

Collecting semantics

- A program static analysis determines a property of the program executions as defined by a (so-called standard) semantics
- The so-called collecting [1] semantics defines the strongest static property of interest
- A collecting semantics therefore defines a whole class of static analyzes, all the ones that abstract/approximate it

[1] Used to be called *static semantics* in [1], it collects information about programs.

- There is no "universal" collecting semantics, since the information collected about program runtime executions can always be refined
- Examples of collecting semantics are computation traces, transitive closure of the program transition relation, set of states/predicate transformers, forward/backward reachable states, etc.

Reference

[1] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, California, 1977. ACM Press, New York, NY, USA.

Collecting Semantics of Arithmetic Expressions

Definition of the forward collecting semantics of arithmetic expressions

The *forward/bottom-up collecting semantics* of an arithmetic expression defines the possible values that the arithmetic expression can evaluate to in a given set of environments[2]

$$\text{Faexp} \in \text{Aexp} \mapsto \wp(\mathbb{R}) \xrightarrow{\text{cjm}} \wp(\mathbb{I}_\Omega),$$

$$\text{Faexp}[\![A]\!]R \stackrel{\text{def}}{=} \{v \mid \exists \rho \in R : \rho \vdash A \Rightarrow v\} . \qquad (1)$$

Reference

[2] E.W. Dijkstra and C.S. Scholten. *Predicate Calculus and Program Semantics.* Springer, 1990.

2 The forward collecting semantics $\text{Faexp}[\![A]\!]R$ specifies the strongest postcondition that values of the arithmetic expression A do satisfy when this expression is evaluated in an environment satisfying the precondition R. The forward collecting semantics can therefore be understood as a predicate transformer [2].

Properties of the forward collecting semantics of arithmetic expressions

The *forward/bottom-up collecting semantics* is a complete join morphism (denoted with $\xrightarrow{\text{cjm}}$), that is ($\mathcal{S}$ is an arbitrary set)

$$\text{Faexp}[\![A]\!]\Big(\bigcup_{k\in\mathcal{S}} R_k\Big) = \bigcup_{k\in\mathcal{S}} (\text{Faexp}[\![A]\!]R_k) ,$$

which implies monotony (when $\mathcal{S} = \{1,2\}$ and $R_1 \subseteq R_2$) and $\emptyset$-strictness (when $\mathcal{S} = \emptyset$)

$$\text{Faexp}[\![A]\!]\emptyset = \emptyset .$$

Structural specification of the forward collecting semantics of arithmetic expressions

$$\text{Faexp}[\![\mathtt{n}]\!]R = \{\underline{\mathtt{n}}\}^{3}$$

$$\text{Faexp}[\![\mathtt{X}]\!]R = R(\mathtt{X})$$

$$\text{where } R(\mathtt{X}) \stackrel{\text{def}}{=} \{\rho(\mathtt{X}) \mid \rho \in R\}$$

$$\text{Faexp}[\![?]\!] = \mathbb{I}$$

$$\text{Faexp}[\![\mathtt{u}\ A']\!]R = \underline{\mathtt{u}}^{\mathcal{C}}(\text{Faexp}[\![A']\!]R)$$

$$\text{where } \underline{\mathtt{u}}^{\mathcal{C}}(V) \stackrel{\text{def}}{=} \{\underline{\mathtt{u}}(v) \mid v \in V\}$$

$$\text{Faexp}[\![A_1\ \mathtt{b}\ A_2]\!]R = \underline{\mathtt{b}}^{\mathcal{C}}(\text{Faexp}[\![A_1]\!], \text{Faexp}[\![A_2]\!])R$$

$$\text{where } \underline{\mathtt{b}}^{\mathcal{C}}(F_1, F_2)R \stackrel{\text{def}}{=} \{v_1\ \underline{\mathtt{b}}\ v_2 \mid \exists \rho \in R : v_1 \in F_1(\{\rho\}) \wedge v_2 \in F_2(\{\rho\})\}$$

3 For short, the case $\text{Faexp}[\![A]\!]\emptyset = \emptyset$ is not recalled.

PROOF.

$$\begin{aligned}
& \mathrm{Faexp}[\![\mathtt{n}]\!]R \\
\stackrel{\text{def}}{=}\ & \{v \mid \exists \rho \in R : \rho \vdash \mathtt{n} \Rrightarrow v\} \\
=\ & \{\underline{\mathtt{n}}\}
\end{aligned}$$

$$\begin{aligned}
& \mathrm{Faexp}[\![\mathtt{X}]\!]R \\
\stackrel{\text{def}}{=}\ & \{v \mid \exists \rho \in R : \rho \vdash \mathtt{X} \Rrightarrow v\} \\
=\ & \{\rho(\mathtt{X}) \mid \rho \in R\} \\
\stackrel{\text{def}}{=}\ & R(\mathtt{X})
\end{aligned}$$

$$\begin{aligned}
& \mathrm{Faexp}[\![\mathtt{?}]\!] \\
\stackrel{\text{def}}{=}\ & \{v \mid \exists \rho \in R : \rho \vdash \mathtt{?} \Rrightarrow v\} \\
=\ & \{v \mid v \in \mathbb{I}\} \\
=\ & \mathbb{I}
\end{aligned}$$

$$\begin{aligned}
& \mathrm{Faexp}[\![\mathtt{u}\ A']\!]R \\
\stackrel{\text{def}}{=}\ & \{v \mid \exists \rho \in R : \rho \vdash \mathtt{u}\ A' \Rrightarrow v\} \\
=\ & \{\underline{\mathtt{u}}\, v \mid \exists \rho \in R : \rho \vdash A' \Rrightarrow v\} \\
=\ & \{\underline{\mathtt{u}}\, v \mid v \in \{v' \mid \exists \rho \in R : \rho \vdash A' \Rrightarrow v'\}\} \\
=\ & \{\underline{\mathtt{u}}\, v \mid v \in \mathrm{Faexp}[\![A']\!]R\} \\
=\ & \underline{\mathtt{u}}^{C}(\mathrm{Faexp}[\![A']\!]R)
\end{aligned}$$

$$\begin{aligned}
& \mathrm{Faexp}[\![A_1\ \mathtt{b}\ A_2]\!]R \\
\stackrel{\text{def}}{=}\ & \{v \mid \exists \rho \in R : \rho \vdash A_1\ \mathtt{b}\ A_2 \Rrightarrow v\} \\
=\ & \{v_1\ \underline{\mathtt{b}}\ v_2 \mid \exists \rho \in R : \rho \vdash A_1 \Rrightarrow v_1 \wedge \rho \vdash A_2 \Rrightarrow v_2\} \\
=\ & \{v_1\ \underline{\mathtt{b}}\ v_2 \mid \exists \rho \in R : v_1 \in \{v_1' \mid \exists \rho' \in \{\rho\} : \rho' \vdash A_1 \Rrightarrow v_1'\} \wedge v_2 \in \{v_2' \mid \exists \rho' \in \\
& \{\rho\} : \rho' \vdash A_2 \Rrightarrow v_2'\}\} \\
=\ & \{v_1\ \underline{\mathtt{b}}\ v_2 \mid \exists \rho \in R : v_1 \in \mathrm{Faexp}[\![A_1]\!](\{\rho\}) \wedge v_2 \in \mathrm{Faexp}[\![A_2]\!](\{\rho\})\} \\
\stackrel{\text{def}}{=}\ & \underline{\mathtt{b}}^{C}(\mathrm{Faexp}[\![A_1]\!], \mathrm{Faexp}[\![A_2]\!])R
\end{aligned}$$

□

Collecting Semantics of Boolean Expressions

Definition of the forward collecting semantics of boolean expressions

The *collecting semantics* $\mathrm{Cbexp}[\![B]\!]R$ of a boolean expression B defines the subset of possible environments $\rho \in R$ for which the boolean expression may evaluate to true (hence without producing a runtime error)

$$\mathrm{Cbexp} \in \mathrm{Bexp} \mapsto \wp(\mathbb{R}) \xrightarrow{\text{cjm}} \wp(\mathbb{R}),$$

$$\mathrm{Cbexp}[\![B]\!]R \stackrel{\text{def}}{=} \{\rho \in R \mid \rho \vdash B \Rrightarrow \mathtt{tt}\}\ . \qquad (2)$$

Structural specification of the forward collecting semantics of boolean expressions

$$\mathrm{Cbexp}\llbracket \mathtt{true} \rrbracket R = R$$
$$\mathrm{Cbexp}\llbracket \mathtt{false} \rrbracket R = \emptyset$$
$$\mathrm{Cbexp}\llbracket A_1 \mathrel{\mathsf{c}} A_2 \rrbracket = \underline{\mathsf{c}}^{\mathcal{C}}\ (\mathrm{Faexp}\llbracket A_1 \rrbracket, \mathrm{Faexp}\llbracket A_2 \rrbracket) R$$

where $\underline{\mathsf{c}}^{\mathcal{C}}\ (F, G) R \stackrel{\text{def}}{=} \{\rho \in R \mid \exists v_1 \in F(\{\rho\}) \cap \mathbb{I} : \exists v_2 \in G(\{\rho\}) \cap \mathbb{I} : v_1 \mathrel{\underline{\mathsf{c}}} v_2 = \mathit{tt}\}$

$$\mathrm{Cbexp}\llbracket B_1 \mathbin{\&} B_2 \rrbracket R = \mathrm{Cbexp}\llbracket B_1 \rrbracket R \cap \mathrm{Cbexp}\llbracket B_2 \rrbracket R$$
$$\mathrm{Cbexp}\llbracket B_1 \mid B_2 \rrbracket R = (\mathrm{Cbexp}\llbracket B_1 \rrbracket R \cap (\mathrm{Cbexp}\llbracket B_2 \rrbracket R \cup \mathrm{Cbexp}\llbracket T(\neg B_2) \rrbracket R)) \cup (\mathrm{Cbexp}\llbracket B_2 \rrbracket R \cap (\mathrm{Cbexp}\llbracket B_1 \rrbracket R \cup \mathrm{Cbexp}\llbracket T(\neg B_1) \rrbracket R))$$

Proof.

$\mathrm{Cbexp}\llbracket \mathtt{true} \rrbracket R$

$\stackrel{\text{def}}{=} \{\rho \in R \mid \rho \vdash \mathtt{true} \Rrightarrow \mathit{tt}\}$

$= \{\rho \mid \rho \in R\}$

$= R$

$\mathrm{Cbexp}\llbracket \mathtt{false} \rrbracket R$

$\stackrel{\text{def}}{=} \{\rho \in R \mid \rho \vdash \mathtt{false} \Rrightarrow \mathit{tt}\}$

$= \{\rho \mid \mathit{ff}\}$

$= \emptyset$

$\mathrm{Cbexp}\llbracket A_1 \mathrel{\mathsf{c}} A_2 \rrbracket$

$\stackrel{\text{def}}{=} \{\rho \in R \mid \rho \vdash A_1 \mathrel{\mathsf{c}} A_2 \Rrightarrow \mathit{tt}\}$

$= \{\rho \in R \mid \exists v_1, v_2 \in \mathbb{I}_\Omega : \rho \vdash A_1 \Rrightarrow v_1 \wedge \rho \vdash A_2 \Rrightarrow v_2 \wedge v_1 \mathrel{\underline{\mathsf{c}}} v_2 = \mathit{tt}\}$

$= \{\rho \in R \mid \exists v_1, v_2 \in \mathbb{I}_\Omega : v_1 \in \{v' \mid \exists \rho' \in \{\rho\} : \rho' \vdash A_1 \Rrightarrow v'\} \wedge v_2 \in \{v' \mid \exists \rho' \in \{\rho\} : \rho' \vdash A_2 \Rrightarrow v'\} \wedge v_1 \mathrel{\underline{\mathsf{c}}} v_2 = \mathit{tt}\}$

$= \{\rho \in R \mid \exists v_1, v_2 \in \mathbb{I}_\Omega : v_1 \in \mathrm{Faexp}\llbracket A_1 \rrbracket\{\rho\} \wedge v_2 \in \mathrm{Faexp}\llbracket A_2 \rrbracket\{\rho\} \wedge v_1 \mathrel{\underline{\mathsf{c}}} v_2 = \mathit{tt}\}$

$= \quad (v_1 \mathrel{\underline{\mathsf{c}}} \Omega = \Omega \mathrel{\underline{\mathsf{c}}} v_2 = \Omega \neq \mathit{tt})$

$\{\rho \in R \mid \exists v_1 \in \mathrm{Faexp}\llbracket A_1 \rrbracket\{\rho\} \cap \mathbb{I} : \exists v_2 \in \mathrm{Faexp}\llbracket A_2 \rrbracket\{\rho\} \cap \mathbb{I} : v_1 \mathrel{\underline{\mathsf{c}}} v_2 = \mathit{tt}\}$

$= \underline{\mathsf{c}}^{\mathcal{C}}\ (\mathrm{Faexp}\llbracket A_1 \rrbracket, \mathrm{Faexp}\llbracket A_2 \rrbracket) R$

$\mathrm{Cbexp}\llbracket B_1 \mathbin{\&} B_2 \rrbracket R$

$= \{\rho \in R \mid \rho \vdash B_1 \Rrightarrow w_1 \wedge \rho \vdash B_2 \Rrightarrow w_2 \wedge w_1 \mathbin{\underline{\&}} w_2 = \mathit{tt}\}$

$= (\{\rho \in R \mid \rho \vdash B_1 \Rrightarrow \mathit{tt} \wedge \rho \vdash B_2 \Rrightarrow \mathit{tt}\})$

$= \{\rho \in R \mid \rho \vdash B_1 \Rrightarrow \mathit{tt}\} \cap \{\rho \in R \rho \vdash B_2 \Rrightarrow \mathit{tt}\}$

$= \mathrm{Cbexp}\llbracket B_1 \rrbracket R \cap \mathrm{Cbexp}\llbracket B_2 \rrbracket R$

$\mathrm{Cbexp}\llbracket B_1 \mid B_2 \rrbracket R$

$= \{\rho \in R \mid \rho \vdash B_1 \Rrightarrow w_1 \wedge \rho \vdash B_2 \Rrightarrow w_2 \wedge w_1 \mathbin{\underline{\mid}} w_2 = \mathit{tt}\}$

$= \{\rho \in R \mid \rho \vdash B_1 \Rrightarrow \mathit{tt} \vee \rho \vdash B_2 \Rrightarrow \mathit{tt}\}$

$=$ (Avoiding the case when B_1 holds but B_2 yields to a runtime error, or inversely)

$\{\rho \in R \mid \rho \vdash B_1 \Rrightarrow \mathit{tt} \wedge (\rho \vdash B_2 \Rrightarrow \mathit{tt} \vee \rho \vdash B_2 \Rrightarrow \mathit{ff})\} \cup \{\rho \in R \mid \rho \vdash B_2 \Rrightarrow \mathit{tt} \wedge (\rho \vdash B_2 \Rrightarrow \mathit{tt} \vee \rho \vdash B_2 \Rrightarrow \mathit{ff}) \wedge (\rho \vdash B_1 \Rrightarrow \mathit{tt} \vee \rho \vdash B_1 \Rrightarrow \mathit{ff})\}$

$= (\mathrm{Cbexp}\llbracket B_1 \rrbracket R \cap (\mathrm{Cbexp}\llbracket B_2 \rrbracket R \cup \mathrm{Cbexp}\llbracket T(\neg B_2) \rrbracket R)) \cup (\mathrm{Cbexp}\llbracket B_2 \rrbracket R \cap (\mathrm{Cbexp}\llbracket B_1 \rrbracket R \cup \mathrm{Cbexp}\llbracket T(\neg B_1) \rrbracket R))$

$\square$

Small-step operation semantics of commands

Recall that in lecture 5, we have defined the transition system of a program $P = S\ ;;$ as

$$\langle \Sigma[\![P]\!],\ \tau[\![P]\!] \rangle \tag{3}$$

where $\Sigma[\![P]\!]$ is the set of program states and $\tau[\![C]\!]$, $C \in \mathrm{Cmp}[\![P]\!]$ is the transition relation for component C of program P, defined by

$$\tau[\![C]\!] \stackrel{\mathrm{def}}{=} \{\langle\langle \ell,\ \rho\rangle,\ \langle \ell',\ \rho'\rangle\rangle \mid \langle \ell,\ \rho\rangle \models\!\!=[\![C]\!]\!\!\Longrightarrow \langle \ell',\ \rho'\rangle\} \tag{4}$$

– Execution starts at the program entry point with all variables uninitialized:

$$\mathrm{Entry}[\![P]\!] \stackrel{\mathrm{def}}{=} \{\langle \mathrm{at}_P[\![P]\!],\ \lambda \mathtt{X} \in \mathrm{Var}[\![P]\!] \cdot \Omega_{\mathrm{i}}\rangle\}\ . \tag{5}$$

– Execution ends without error when control reaches the program exit point

$$\mathrm{Exit}[\![P]\!] \stackrel{\mathrm{def}}{=} \{\mathrm{after}_P[\![P]\!]\} \times \mathrm{Env}[\![P]\!]\ .$$

When the evaluation of an arithmetic or boolean expression fails with a runtime error, the program execution is blocked so that no further transition is possible.

– A basic result on the program transition relation is that it is not possible to jump into or out of program components ($C \in \mathrm{Cmp}[\![P]\!]$)

$$\langle\langle \ell,\ \rho\rangle,\ \langle \ell',\ \rho'\rangle\rangle \in \tau[\![C]\!] \Longrightarrow \{\ell, \ell'\} \subseteq \mathrm{in}_P[\![C]\!]\ . \tag{6}$$

Collecting Semantics of Commands

Big-step operational semantics of commands

- The reflexive transitive closure of the transition relation $\tau[\![C]\!]$ of a program component $C \in \mathrm{Cmp}[\![P]\!]$ is $\tau^\star[\![C]\!] \stackrel{\mathrm{def}}{=} (\tau[\![C]\!])^\star$.
- This is called the big-step operational semantics of commands
- $\tau^\star[\![P]\!]$ can be expressed compositionally (in the sense of denotational semantics, by structural induction on the program components $C \in \mathrm{Cmp}[\![P]\!]$ of program P)

- Observe that contrary to the classical big step operational or natural semantics [3], the effect of execution is described not only from entry to exit states but also from any (intermediate) state to any subsequently reachable state. This is better adapted to our later reachability analyses.

Reference

[3] G.D. Plotkin. A structural approach to operational semantics. Tech. rep. DAIMI FN-19, Aarhus University, Denmark, Sep. 1981.

Structural big-step operational semantics: skip and assignment

$$\tau^\star[\![\mathtt{skip}]\!] = 1_{\Sigma[\![P]\!]} \cup \tau[\![\mathtt{skip}]\!] \tag{7}$$

$$\tau^\star[\![\mathtt{X} := A]\!] = 1_{\Sigma[\![P]\!]} \cup \tau[\![\mathtt{X} := A]\!]$$

Proof. For the identity $C = \mathtt{skip}$ and the assignment $C = \mathtt{X} := A$

$\tau^\star[\![C]\!]$

$=$ ⟅def. of $(\tau[\![C]\!])^\star$ and $\tau[\![C]\!]$ so that $\mathrm{at}_P[\![C]\!] \neq \mathrm{after}_P[\![C]\!]$ implies $(\tau[\![C]\!])^2 = \emptyset$, whence by recurrence $(\tau[\![C]\!])^n = \emptyset$ for all $n \geq 2$, 1_S was defined as the identity on the set S⟆

$1_{\Sigma[\![P]\!]} \cup \tau[\![C]\!]$.

∎

Structural big-step operational semantics: conditional command

$$\begin{aligned}\tau^\star[\![\mathtt{if}\ B\ \mathtt{then}\ S_t\ \mathtt{else}\ S_f\ \mathtt{fi}]\!] = &\\ &(1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t) \ \cup \\ &(1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^f)\end{aligned} \tag{8}$$

where:

$\tau^B \stackrel{\mathrm{def}}{=} \{\langle\langle \mathrm{at}_P[\![\mathtt{if}\ B\ \mathtt{then}\ S_t\ \mathtt{else}\ S_f\ \mathtt{fi}]\!],\ \rho\rangle,\ \langle \mathrm{at}_P[\![S_t]\!],\ \rho\rangle\rangle \mid \rho \vdash B \Rightarrow \mathtt{tt}\}$

$\tau^{\bar{B}} \stackrel{\mathrm{def}}{=} \{\langle\langle \mathrm{at}_P[\![\mathtt{if}\ B\ \mathtt{then}\ S_t\ \mathtt{else}\ S_f\ \mathtt{fi}]\!],\ \rho\rangle,\ \langle \mathrm{at}_P[\![S_f]\!],\ \rho\rangle\rangle \mid \rho \vdash T(\neg B) \Rightarrow \mathtt{tt}\}$

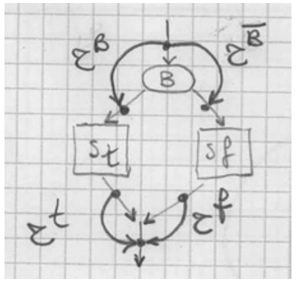
$\tau^t \stackrel{\mathrm{def}}{=} \{\langle\langle \mathrm{after}_P[\![S_t]\!],\ \rho\rangle,\ \langle \mathrm{after}_P[\![\mathtt{if}\ B\ \mathtt{then}\ S_t\ \mathtt{else}\ S_f\ \mathtt{fi}]\!],\ \rho\rangle\rangle \mid \rho \in \mathrm{Env}[\![P]\!]\}$

$\tau^f \stackrel{\mathrm{def}}{=} \{\langle\langle \mathrm{after}_P[\![S_f]\!],\ \rho\rangle,\ \langle \mathrm{after}_P[\![\mathtt{if}\ B\ \mathtt{then}\ S_t\ \mathtt{else}\ S_f\ \mathtt{fi}]\!],\ \rho\rangle\rangle \mid \rho \in \mathrm{Env}[\![P]\!]\}$

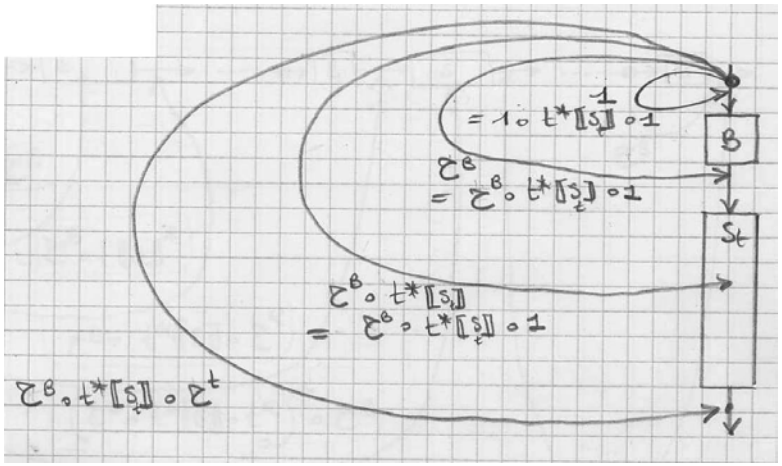
Auxiliary definitions

For the conditional $C = \texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}$, we define

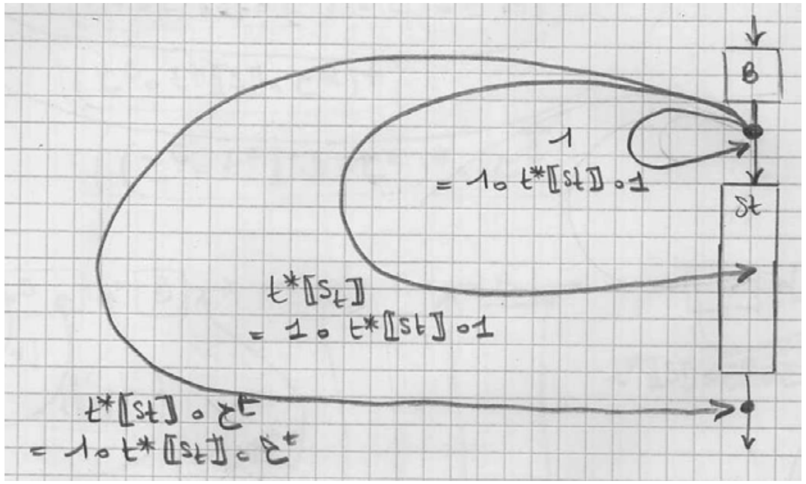
$$
\begin{aligned}
\tau^B &\stackrel{\text{def}}{=} \{\langle\langle \text{at}_P[\![C]\!],\ \rho\rangle,\ \langle \text{at}_P[\![S_t]\!],\ \rho\rangle\rangle \mid \rho \vdash B \Rightarrow \text{tt}\},\\
\tau^{\bar{B}} &\stackrel{\text{def}}{=} \{\langle\langle \text{at}_P[\![C]\!],\ \rho\rangle,\ \langle \text{at}_P[\![S_f]\!],\ \rho\rangle\rangle \mid \rho \vdash T(\neg B) \Rightarrow \text{tt}\},\\
\tau^t &\stackrel{\text{def}}{=} \{\langle\langle \text{after}_P[\![S_t]\!],\ \rho\rangle,\ \langle \text{after}_P[\![C]\!],\ \rho\rangle\rangle \mid \rho \in \text{Env}[\![P]\!]\},\\
\tau^f &\stackrel{\text{def}}{=} \{\langle\langle \text{after}_P[\![S_f]\!],\ \rho\rangle,\ \langle \text{after}_P[\![C]\!],\ \rho\rangle\rangle \mid \rho \in \text{Env}[\![P]\!]\}\ .
\end{aligned}
$$



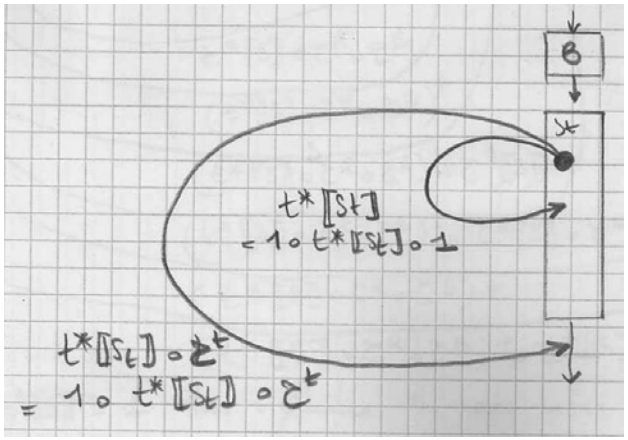
Starting at the test entry...



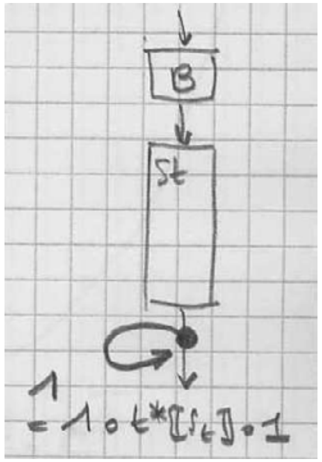
Starting before the "then" part entry...



Starting within the "then" branch



Starting after the "then" branch



Proof. Recall from Lecture 5, that for the $C = \texttt{if}\ B\ \texttt{then}\ S_t\ \texttt{else}\ S_f\ \texttt{fi}$ (where $\mathrm{at}_P[\![C]\!] = \ell$ and $\mathrm{after}_P[\![C]\!] = \ell'$), we have:

$$\frac{\rho \vdash B \Mapsto \mathtt{tt}}{\langle \ell,\ \rho\rangle \models\!\!=\!\![\![\texttt{if}\ B\ \texttt{then}\ S_t\ \texttt{else}\ S_f\ \texttt{fi}]\!]\!\!\Longrightarrow \langle \mathrm{at}_P[\![S_t]\!],\ \rho\rangle}\ , \tag{9}$$

$$\frac{\rho \vdash T(\neg B) \Mapsto \mathtt{tt}}{\langle \ell,\ \rho\rangle \models\!\!=\!\![\![\texttt{if}\ B\ \texttt{then}\ S_t\ \texttt{else}\ S_f\ \texttt{fi}]\!]\!\!\Longrightarrow \langle \mathrm{at}_P[\![S_f]\!],\ \rho\rangle}\ . \tag{10}$$

$$\frac{\langle \ell_1,\ \rho_1\rangle \models\!\!=\!\![\![S_t]\!]\!\!\Longrightarrow \langle \ell_2,\ \rho_2\rangle}{\langle \ell_1,\ \rho_1\rangle \models\!\!=\!\![\![\texttt{if}\ B\ \texttt{then}\ S_t\ \texttt{else}\ S_f\ \texttt{fi}]\!]\!\!\Longrightarrow \langle \ell_2,\ \rho_2\rangle}\ , \tag{11}$$

$$\frac{\langle \ell_1,\ \rho_1\rangle \models\!\!=\!\![\![S_f]\!]\!\!\Longrightarrow \langle \ell_2,\ \rho_2\rangle}{\langle \ell_1,\ \rho_1\rangle \models\!\!=\!\![\![\texttt{if}\ B\ \texttt{then}\ S_t\ \texttt{else}\ S_f\ \texttt{fi}]\!]\!\!\Longrightarrow \langle \ell_2,\ \rho_2\rangle}\ . \tag{12}$$

$$\langle \mathrm{after}_P[\![S_t]\!],\ \rho\rangle \models\!\!=\!\![\![\texttt{if}\ B\ \texttt{then}\ S_t\ \texttt{else}\ S_f\ \texttt{fi}]\!]\!\!\Longrightarrow \langle \ell',\ \rho\rangle\ , \tag{13}$$

$$\langle \mathrm{after}_P[\![S_f]\!],\ \rho\rangle \models\!\!=\!\![\![\texttt{if}\ B\ \texttt{then}\ S_t\ \texttt{else}\ S_f\ \texttt{fi}]\!]\!\!\Longrightarrow \langle \ell',\ \rho\rangle\ . \tag{14}$$

Recall also from Lecture 5, that the labelling scheme of a conditional command $C = \texttt{if}\ B\ \texttt{then}\ S_t\ \texttt{else}\ S_f\ \texttt{fi} \in \mathrm{Cmp}[\![P]\!]$ satisfies

$$\begin{aligned}
\mathrm{in}_P[\![C]\!] &= \{\mathrm{at}_P[\![C]\!], \mathrm{after}_P[\![C]\!]\} \cup \mathrm{in}_P[\![S_t]\!] \cup \mathrm{in}_P[\![S_f]\!],\\
\{\mathrm{at}_P[\![C]\!], \mathrm{after}_P[\![C]\!]\} \cap (\mathrm{in}_P[\![S_t]\!] \cup \mathrm{in}_P[\![S_f]\!]) &= \emptyset,\\
\mathrm{in}_P[\![S_t]\!] \cap \mathrm{in}_P[\![S_f]\!] &= \emptyset\ .
\end{aligned} \tag{15}$$

It follows that by (9) to (14), we have

$$\tau[\![C]\!] = \tau_{\mathtt{tt}}[\![C]\!] \cup \tau_{\mathtt{ff}}[\![C]\!]$$

where

$$\begin{aligned}
\tau_{\mathtt{tt}}[\![C]\!] &\stackrel{\text{def}}{=} \tau^B \cup \tau[\![S_t]\!] \cup \tau^t,\\
\tau_{\mathtt{ff}}[\![C]\!] &\stackrel{\text{def}}{=} \tau^{\bar{B}} \cup \tau[\![S_f]\!] \cup \tau^f\ .
\end{aligned}$$

By the conditions (15) and (6) on labelling of the conditional command C, we have $\tau_{\mathtt{tt}}[\![C]\!] \circ \tau_{\mathtt{ff}}[\![C]\!] = \tau_{\mathtt{ff}}[\![C]\!] \circ \tau_{\mathtt{tt}}[\![C]\!] = \emptyset$ so that

$$\tau^\star[\![C]\!] = (\tau_{\mathtt{tt}}[\![C]\!])^\star \cup (\tau_{\mathtt{ff}}[\![C]\!])^\star\ . \tag{16}$$

Intuitively the steps which are repeated in the conditional must all take place in one branch or the other since it is impossible to jump from one branch into the other.

Assume by induction hypothesis that

$$(\tau_{\mathtt{tt}}[\![C]\!])^n = \tau^B \circ \tau[\![S_t]\!]^{n-2} \circ \tau^t \cup \tau^B \circ \tau[\![S_t]\!]^{n-1} \cup \tau[\![S_t]\!]^{n-1} \circ \tau^t \cup \tau[\![S_t]\!]^n \tag{17}$$

This holds for the basis $n = 1$ since $\tau[\![S_t]\!]^{-1} = \emptyset$ and $\tau[\![S_t]\!]^0 = 1_{\Sigma[\![P]\!]}$ is the identity. For $n \geq 1$, we have

$$\begin{aligned}
& (\tau_{\mathtt{tt}}[\![C]\!])^{n+1}\\
= \quad & \wr \text{def. } t^{n+1} = t^n \circ t \wr\\
& (\tau_{\mathtt{tt}}[\![C]\!])^n \circ \tau_{\mathtt{tt}}[\![C]\!]\\
= \quad & \wr \text{induction hypothesis} \wr\\
& (\tau^B \circ \tau[\![S_t]\!]^{n-2} \circ \tau^t \cup \tau^B \circ \tau[\![S_t]\!]^{n-1} \cup \tau[\![S_t]\!]^{n-1} \circ \tau^t \cup \tau[\![S_t]\!]^n) \circ \tau_{\mathtt{tt}}[\![C]\!]\\
= \quad & \wr \circ \text{ distributes over } \cup \text{ (and } \circ \text{ has priority over } \cup) \wr\\
& \tau^B \circ \tau[\![S_t]\!]^{n-2} \circ \tau^t \circ \tau_{\mathtt{tt}}[\![C]\!] \cup \tau^B \circ \tau[\![S_t]\!]^{n-1} \circ \tau_{\mathtt{tt}}[\![C]\!] \cup \tau[\![S_t]\!]^{n-1} \circ \tau^t \circ \tau_{\mathtt{tt}}[\![C]\!] \cup\\
& \tau[\![S_t]\!]^n \circ \tau_{\mathtt{tt}}[\![C]\!]\\
= \quad & \wr \text{by the labelling scheme (15), (6) and the def. (9) to (14) of the possible transitions so that } \tau^t \circ \tau_{\mathtt{tt}}[\![C]\!] = \emptyset, \text{ etc.} \wr
\end{aligned}$$

$$
\begin{aligned}
& \tau^B \circ \tau\llbracket S_t\rrbracket^{n-1} \circ \tau_{\mathtt{tt}}\llbracket C\rrbracket \cup \tau\llbracket S_t\rrbracket^{n} \circ \tau_{\mathtt{tt}}\llbracket C\rrbracket \\
= \quad & \wr\text{def. of } \tau_{\mathtt{tt}}\llbracket C\rrbracket \text{ and } \circ \text{ distributes over } \cup\wr \\
& \tau^B \circ \tau\llbracket S_t\rrbracket^{n-1} \circ \tau^B \cup \tau^B \circ \tau\llbracket S_t\rrbracket^{n-1} \circ \tau\llbracket S_t\rrbracket \cup \tau^B \circ \tau\llbracket S_t\rrbracket^{n-1} \circ \tau^t \cup \tau\llbracket S_t\rrbracket^{n} \circ \tau^B \cup \\
& \tau\llbracket S_t\rrbracket^{n} \circ \tau\llbracket S_t\rrbracket \cup \tau\llbracket S_t\rrbracket^{n} \circ \tau^t \\
= \quad & \wr\text{by the labelling scheme (15), (6) and the def. (9) to (14) of the} \\
& \quad\text{possible transitions so that } \tau^B \circ \tau^B = \emptyset,\ \tau\llbracket S_t\rrbracket^{n} \circ \tau^B, \text{ etc.}\wr \\
& \tau^B \circ \tau\llbracket S_t\rrbracket^{n} \cup \tau^B \circ \tau\llbracket S_t\rrbracket^{n-1} \circ \tau^t \cup \tau\llbracket S_t\rrbracket^{n+1} \cup \tau\llbracket S_t\rrbracket^{n} \circ \tau^t \\
= \quad & \wr\cup \text{ is associative and commutative and def. (17) of } (\tau_{\mathtt{tt}}\llbracket C\rrbracket)^{n+1}\wr \\
& (\tau_{\mathtt{tt}}\llbracket C\rrbracket)^{n+1} .
\end{aligned}
$$

By recurrence, (17) holds for all $n \geq 1$ so that

$$
\begin{aligned}
& (\tau_{\mathtt{tt}}\llbracket C\rrbracket)^{\star} \\
= \quad & \wr\text{def. } t^{\star}\wr \\
& (\tau_{\mathtt{tt}}\llbracket C\rrbracket)^{0} \cup \bigcup_{n\geq 1} (\tau_{\mathtt{tt}}\llbracket C\rrbracket)^{n} \\
= \quad & \wr\text{def. } t^{0} \text{ and (17)}\wr
\end{aligned}
$$

$$
\begin{aligned}
& 1_{\Sigma\llbracket P\rrbracket} \cup \bigcup_{n\geq 1} (\tau^B \circ \tau\llbracket S_t\rrbracket^{n-2} \circ \tau^t \cup \tau^B \circ \tau\llbracket S_t\rrbracket^{n-1} \cup \tau\llbracket S_t\rrbracket^{n-1} \circ \tau^t \cup \tau\llbracket S_t\rrbracket^{n}) \\
= \quad & \wr\circ \text{ distributes over } \cup\wr \\
& 1_{\Sigma\llbracket P\rrbracket} \cup \tau^B \circ \left(\bigcup_{n\geq 1} \tau\llbracket S_t\rrbracket^{n-2}\right) \circ \tau^t \cup \tau^B \circ \left(\bigcup_{n\geq 1} \tau\llbracket S_t\rrbracket^{n-1}\right) \cup \left(\bigcup_{n\geq 1} \tau\llbracket S_t\rrbracket^{n-1}\right) \circ \\
& \tau^t \cup \bigcup_{n\geq 1} \tau\llbracket S_t\rrbracket^{n} \\
= \quad & \wr\text{changing variables } k = n-2 \text{ and } j = n-1,\ \tau\llbracket S_t\rrbracket^{-1} = \emptyset,\ \tau\llbracket S_t\rrbracket^{0} = \\
& \quad 1_{\Sigma\llbracket P\rrbracket} \text{ and by the labelling scheme (15), (6) and the def. (9) to (14)} \\
& \quad\text{of the possible transitions, } \tau^B \circ \tau^t = \emptyset, \text{ etc.}\wr \\
& \tau^B \circ \left(\bigcup_{k\geq 1} \tau\llbracket S_t\rrbracket^{k}\right) \circ \tau^t \cup \tau^B \circ \left(\bigcup_{j\geq 0} \tau\llbracket S_t\rrbracket^{j}\right) \cup \left(\bigcup_{j\geq 0} \tau\llbracket S_t\rrbracket^{j}\right) \circ \tau^t \cup \bigcup_{n\geq 0} \tau\llbracket S_t\rrbracket^{n} \\
= \quad & \wr\tau^B \circ \tau^t = \emptyset \text{ and def. of } t^{\star}\wr \\
& \tau^B \circ (\tau\llbracket S_t\rrbracket)^{\star} \circ \tau^t \cup \tau^B \circ (\tau\llbracket S_t\rrbracket)^{\star} \cup (\tau\llbracket S_t\rrbracket)^{\star} \circ \tau^t \cup (\tau\llbracket S_t\rrbracket)^{\star}
\end{aligned}
$$

$$
\begin{aligned}
= \quad & \wr\circ \text{ distributes over } \cup \text{ (and } \star \text{ has priority over } \circ \text{ which has priority over } \cup)\wr \\
& (1_{\Sigma\llbracket P\rrbracket} \cup \tau^B) \circ (\tau\llbracket S_t\rrbracket)^{\star} \circ (1_{\Sigma\llbracket P\rrbracket} \cup \tau^t) .
\end{aligned}
$$

A similar result is easily established for $(\tau_{\mathtt{ff}}\llbracket C\rrbracket)^{\star}$ whence by (16), we get

$$
\begin{aligned}
\tau^{\star}\llbracket \mathtt{if}\ B\ \mathtt{then}\ S_t\ \mathtt{else}\ S_f\ \mathtt{fi}\rrbracket = \ & (1_{\Sigma\llbracket P\rrbracket} \cup \tau^B) \circ (\tau\llbracket S_t\rrbracket)^{\star} \circ (1_{\Sigma\llbracket P\rrbracket} \cup \tau^t) \cup \\
& (1_{\Sigma\llbracket P\rrbracket} \cup \tau^{\bar{B}}) \circ (\tau\llbracket S_f\rrbracket)^{\star} \circ (1_{\Sigma\llbracket P\rrbracket} \cup \tau^f) .
\end{aligned}
$$

$\square$

Structural big-step operational semantics: iteration command

$$
\begin{aligned}
& \tau^{\star}\llbracket \mathtt{while}\ B\ \mathtt{do}\ S\ \mathtt{od}\rrbracket = \\
& \quad ((1_{\Sigma\llbracket P\rrbracket} \cup \tau^{\star}\llbracket S\rrbracket \circ \tau^{R}) \circ (\tau^{B} \circ \tau^{\star}\llbracket S\rrbracket \circ \tau^{R})^{\star} \circ \\
& \qquad (1_{\Sigma\llbracket P\rrbracket} \cup \tau^{B} \circ \tau^{\star}\llbracket S\rrbracket \cup \tau^{\bar{B}})) \cup \tau\llbracket S\rrbracket^{\star}
\end{aligned} \tag{18}
$$

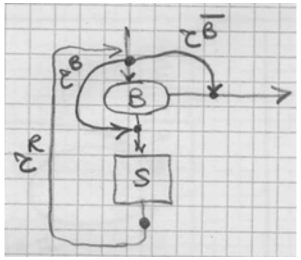
where:

$$
\begin{aligned}
\tau^{B} & \stackrel{\text{def}}{=} \{\langle\langle \mathrm{at}_P\llbracket \mathtt{while}\ B\ \mathtt{do}\ S\ \mathtt{od}\rrbracket,\ \rho\rangle,\ \langle \mathrm{at}_P\llbracket S\rrbracket,\ \rho\rangle\rangle \mid \rho \vdash B \Rightarrow \mathtt{tt}\} \\
\tau^{\bar{B}} & \stackrel{\text{def}}{=} \{\langle\langle \mathrm{at}_P\llbracket \mathtt{while}\ B\ \mathtt{do}\ S\ \mathtt{od}\rrbracket,\ \rho\rangle,\ \langle \mathrm{after}_P\llbracket \mathtt{while}\ B\ \mathtt{do}\ S\ \mathtt{od}\rrbracket,\ \rho\rangle\rangle \mid \\
& \qquad \rho \vdash T(\neg B) \Rightarrow \mathtt{tt}\} \\
\tau^{R} & \stackrel{\text{def}}{=} \{\langle\langle \mathrm{after}_P\llbracket S\rrbracket,\ \rho\rangle,\ \langle \mathrm{at}_P\llbracket \mathtt{while}\ B\ \mathtt{do}\ S\ \mathtt{od}\rrbracket,\ \rho\rangle\rangle \mid \rho \in \mathrm{Env}\llbracket P\rrbracket\}
\end{aligned}
$$

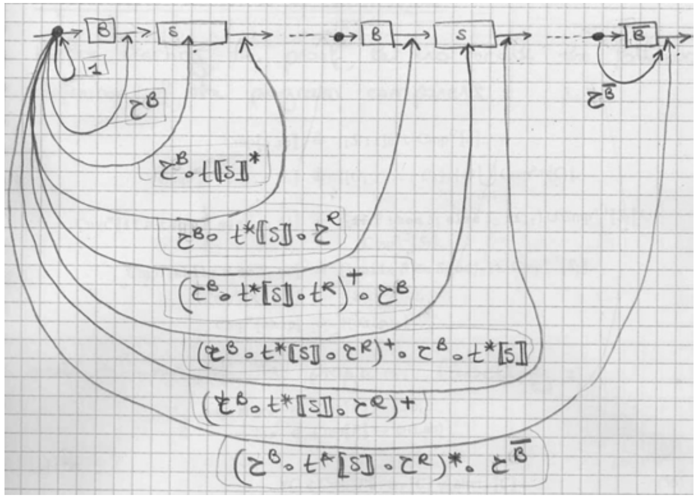
Auxiliary definitions

For the iteration $C = \texttt{while}\ B\ \texttt{do}\ S\ \texttt{od}$, we define

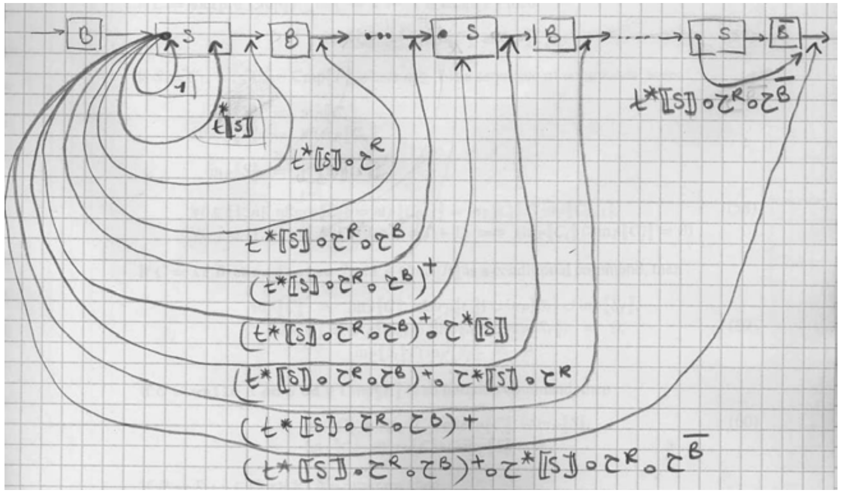
$$\tau^B \stackrel{\text{def}}{=} \{\langle\langle \text{at}_P[\![C]\!],\ \rho\rangle,\ \langle \text{at}_P[\![S]\!],\ \rho\rangle\rangle \mid \rho \vdash B \Rightarrow \text{tt}\},$$
$$\tau^{\bar{B}} \stackrel{\text{def}}{=} \{\langle\langle \text{at}_P[\![C]\!],\ \rho\rangle,\ \langle \text{after}_P[\![C]\!],\ \rho\rangle\rangle \mid \rho \vdash T(\neg B) \Rightarrow \text{tt}\},$$
$$\tau^R \stackrel{\text{def}}{=} \{\langle\langle \text{after}_P[\![S]\!],\ \rho\rangle,\ \langle \text{at}_P[\![C]\!],\ \rho\rangle\rangle \mid \rho \in \text{Env}[\![P]\!]\}\ .$$



Starting at the loop entry...



Starting withing the loop body...



PROOF. Recall that for the iteration $C = \texttt{while}\ B\ \texttt{do}\ S\ \texttt{od}$ (where $\text{at}_P[\![C]\!] = \ell$, $\text{after}_P[\![C]\!] = \ell'$ and $\ell_1, \ell_2 \in \text{in}_P[\![S]\!]$), we have defined

$$\frac{\rho \vdash T(\neg B) \Rightarrow \text{tt}}{\langle \ell,\ \rho\rangle \models\!\![\texttt{while}\ B\ \texttt{do}\ S\ \texttt{od}]\!\!\Longrightarrow \langle \ell',\ \rho\rangle}\ , \tag{19}$$

$$\frac{\rho \vdash B \Rightarrow \text{tt}}{\langle \ell,\ \rho\rangle \models\!\![\texttt{while}\ B\ \texttt{do}\ S\ \texttt{od}]\!\!\Longrightarrow \langle \text{at}_P[\![S]\!],\ \rho\rangle}\ , \tag{20}$$

$$\frac{\langle \ell_1,\ \rho_1\rangle \models\!\![S]\!\!\Longrightarrow \langle \ell_2,\ \rho_2\rangle}{\langle \ell_1,\ \rho_1\rangle \models\!\![\texttt{while}\ B\ \texttt{do}\ S\ \texttt{od}]\!\!\Longrightarrow \langle \ell_2,\ \rho_2\rangle}\ , \tag{21}$$

$$\langle \text{after}_P[\![S]\!],\ \rho\rangle \models\!\![\texttt{while}\ B\ \texttt{do}\ S\ \texttt{od}]\!\!\Longrightarrow \langle \ell,\ \rho\rangle\ . \tag{22}$$

Recall also from Lecture 5 that the labelling scheme of an iteration command $C = \texttt{while}\ B\ \texttt{do}\ S\ \texttt{od} \in \text{Cmp}[\![P]\!]$ satisfies

$$\begin{aligned}&\text{in}_P[\![C]\!] = \{\text{at}_P[\![C]\!], \text{after}_P[\![C]\!]\} \cup \text{in}_P[\![S]\!],\\ &\{\text{at}_P[\![C]\!], \text{after}_P[\![C]\!]\} \cap \text{in}_P[\![S]\!] = \emptyset\ .\end{aligned} \tag{23}$$

and the basic results on the program transition relation which are

$$\forall C \in \mathrm{Cmp}\llbracket P \rrbracket : \mathrm{at}_P\llbracket C \rrbracket \neq \mathrm{after}_P\llbracket C \rrbracket \ . \tag{24}$$

and that it is not possible to jump into or out of program components ($C \in \mathrm{Cmp}\llbracket P \rrbracket$)

$$\langle\langle \ell,\ \rho\rangle,\ \langle \ell',\ \rho'\rangle\rangle \in \tau\llbracket C \rrbracket \implies \{\ell, \ell'\} \subseteq \mathrm{in}_P\llbracket C \rrbracket \ . \tag{25}$$

It follows that by (19) to (22), we have

$$\tau\llbracket C \rrbracket \ = \ \tau^B \cup \tau\llbracket S \rrbracket \cup \tau^R \cup \tau^{\bar{B}} \ . \tag{26}$$

We define the composition $\bigcirc_{i=1}^{n} t_i$ of relations $t_1, \ldots, t_n$[4]:

$$\bigcirc_{i=1}^{n} t_i \stackrel{\text{def}}{=} \emptyset, \qquad \text{when} \quad n < 0,$$

$$\bigcirc_{i=1}^{0} t_i \stackrel{\text{def}}{=} 1_{\Sigma\llbracket P \rrbracket}, \quad \text{when} \quad n = 0,$$

[4] Observe that $\circ$ is associative but not commutative so that the index set must be totally ordered for the notation to be meaningful

$$\bigcirc_{i=1}^{n} t_i \stackrel{\text{def}}{=} t_1 \circ \ldots \circ t_n, \quad \text{when} \quad n > 0 \ .$$

In order to compute $\tau^\star\llbracket C \rrbracket = \bigcup_{n \geq 0} \tau\llbracket C \rrbracket^n$ for the component $C = \mathtt{while}\ B\ \mathtt{do}\ S\ \mathtt{od}$ of program P, we first compute the n-th power $\tau\llbracket C \rrbracket^n$ for $n \geq 0$. By recurrence $\tau\llbracket C \rrbracket^0 = 1_{\Sigma\llbracket P \rrbracket}$, $\tau\llbracket C \rrbracket^1 = \tau\llbracket C \rrbracket = \tau^B \cup \tau\llbracket S \rrbracket \cup \tau^R \cup \tau^{\bar{B}}$. For $n > 1$, we have

$$\begin{array}{ll}
 & (\tau\llbracket C \rrbracket)^2 \\
= & \quad \wr \text{def. } t^2 = t \circ t \wr \\
 & \tau\llbracket C \rrbracket \circ \tau\llbracket C \rrbracket \\
= & \quad \wr \text{def. (26) of } \tau\llbracket C \rrbracket \wr \\
 & (\tau^B \cup \tau\llbracket S \rrbracket \cup \tau^R \cup \tau^{\bar{B}}) \circ (\tau^B \cup \tau\llbracket S \rrbracket \cup \tau^R \cup \tau^{\bar{B}}) \\
= & \quad \wr \circ \text{ distributes over } \cup \text{ (and } \circ \text{ has priority over } \cup) \wr
\end{array}$$

$$\begin{array}{ll}
 & \tau^B \circ \tau^B \cup \tau\llbracket S \rrbracket \circ \tau^B \cup \tau^R \circ \tau^B \cup \tau^{\bar{B}} \circ \tau^B \cup \tau^B \circ \tau\llbracket S \rrbracket \cup \tau\llbracket S \rrbracket \circ \tau\llbracket S \rrbracket \cup \tau^R \circ \tau\llbracket S \rrbracket \cup \\
 & \tau^{\bar{B}} \circ \tau\llbracket S \rrbracket \cup \tau^B \circ \tau^R \cup \tau\llbracket S \rrbracket \circ \tau^R \cup \tau^R \circ \tau^R \cup \tau^{\bar{B}} \circ \tau^R \cup \tau^B \circ \tau^{\bar{B}} \cup \tau\llbracket S \rrbracket \circ \tau^{\bar{B}} \cup \\
 & \tau^R \circ \tau^{\bar{B}} \cup \tau^{\bar{B}} \circ \tau^{\bar{B}} \\
= & \quad \wr \tau^B \circ \tau^B = \emptyset, \text{ by (20) and (24);} \\
 & \quad \ \tau\llbracket S \rrbracket \circ \tau^B = \emptyset, \text{ by (21), (20), (25) and (23);} \\
 & \quad \ \tau^{\bar{B}} \circ \tau^B = \emptyset, \text{ by (19), (20) and (24);} \\
 & \quad \ \tau^R \circ \tau\llbracket S \rrbracket = \emptyset, \text{ by (22), (21) and (23);} \\
 & \quad \ \tau^{\bar{B}} \circ \tau\llbracket S \rrbracket = \emptyset, \text{ by (19), (21), (25) and (23);} \\
 & \quad \ \tau^B \circ \tau^R = \emptyset, \text{ by (20), (22) and (24);} \\
 & \quad \ \tau^R \circ \tau^R = \emptyset, \text{ by (22), (23) and (25);} \\
 & \quad \ \tau^B \circ \tau^{\bar{B}} = \emptyset, \text{ by (20), (19), (23) and (25)} \\
 & \quad \ \tau\llbracket S \rrbracket \circ \tau^{\bar{B}} = \emptyset, \text{ by (21), (19), (23) and (25);} \\
 & \quad \ \tau^{\bar{B}} \circ \tau^{\bar{B}} = \emptyset, \text{ by (19) and (24)} \wr \\
 & \tau^R \circ \tau^B \cup \tau^B \circ \tau\llbracket S \rrbracket \cup \tau\llbracket S \rrbracket^2 \cup \tau\llbracket S \rrbracket \circ \tau^R \cup \tau^R \circ \tau^{\bar{B}} \ .
\end{array}$$

The generalization after computing the first few iterates $n = 1, \ldots, 4$ leads to the following *induction hypothesis* ($n \geq 1$)

$$(\tau\llbracket C \rrbracket)^n \stackrel{\text{def}}{=} A_n \cup B_n \cup C_n \cup D_n \cup E_n \cup F_n \cup G_n \ . \tag{27}$$

where

$$A_n \stackrel{\text{def}}{=} \bigcup_{n = \sum_{i=1}^{j}(k_i+2)} \bigcirc_{i=1}^{j} (\tau^B \circ \tau\llbracket S \rrbracket^{k_i} \circ \tau^R) \ ; \tag{28}$$

(This corresponds to j loops iterations from and to the loop entry $\mathrm{at}_P\llbracket C \rrbracket$ where the i-th execution of the loop body S exactly takes $k_i \geq 1$[5] steps. $A_n = \emptyset$, $n \leq 1$.)

$$B_n \stackrel{\text{def}}{=} \bigcup_{n = (\sum_{i=1}^{j}(k_i+2)) + 1 + \ell} \left(\left(\bigcirc_{i=1}^{j} (\tau^B \circ \tau\llbracket S \rrbracket^{k_i} \circ \tau^R) \right) \circ \tau^B \circ \tau\llbracket S \rrbracket^{\ell} \right) \ ; \tag{29}$$

[5] For short, the constraints $k_i > 0$, $i = 1, \ldots, j$ are not explicitly inserted in the formula.

(This corresponds to j loops iterations from and to the loop entry $\mathrm{at}_P[\![C]\!]$ where the i-th execution of the loop body S exactly takes $k_i \geq 1$ steps followed by a successful condition B and a partial execution of the loop body S for $\ell \geq 0$ [6] steps. $B_0 = \emptyset$, $B_1 = \tau^B$.)

$$C_n \stackrel{\text{def}}{=} \bigcup_{n=(\sum_{i=1}^{j}(k_i+2))+1} \left(\left(\bigcirc_{i=1}^{j}(\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R)\right) \circ \tau^{\bar{B}}\right) ; \qquad (30)$$

(This corresponds to j loops iterations where the i-th execution of the loop body S has $k_i \geq 1$ steps within S until termination with condition B false. $C_0 = \emptyset$, $C_1 = \tau^{\bar{B}}$.)

$$D_n \stackrel{\text{def}}{=} \bigcup_{n=\ell+1+(\sum_{i=1}^{j}(k_i+2))} \left(\tau[\![S]\!]^{\ell} \circ \tau^R \circ \left(\bigcirc_{i=1}^{j}(\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R)\right)\right) ; \qquad (31)$$

[6] Again, the constraint $\ell \geq 0$ is left implicit in the formula.

(This corresponds to an observation of the execution starting in the middle of the loop body S for ℓ steps followed by the jump back to the loop entry $\mathrm{at}_P[\![C]\!]$, followed by j complete loops iterations from and to the loop entry $\mathrm{at}_P[\![C]\!]$ where the i-th execution of the loop body S exactly takes $k_i \geq 1$ steps. $D_0 = \emptyset$, $D_1 = \tau^R$.)

$$E_n \stackrel{\text{def}}{=} \bigcup_{n=(\sum_{i=1}^{j}(k_i+2))+\ell+2+m} \left(\tau[\![S]\!]^{\ell} \circ \tau^R \circ \left(\bigcirc_{i=1}^{j}(\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R)\right) \circ \tau^B \circ \tau[\![S]\!]^m\right) ; \qquad (32)$$

(This corresponds to an observation of the execution starting in the middle of the loop body S for $\ell \geq 0$ steps followed by the jump back to the loop entry $\mathrm{at}_P[\![C]\!]$. Then there are j loops iterations from and to the loop entry $\mathrm{at}_P[\![C]\!]$ where the i-th execution of the loop body S exactly takes $k_i \geq 1$ steps. Finally the condition B holds and a partial execution of the loop body S for $m \geq 0$ steps is performed. $E_0 = E_1 = \emptyset$ and $E_2 = \tau^R \circ \tau^B$.)

$$F_n \stackrel{\text{def}}{=} \bigcup_{n=(\sum_{i=1}^{j}(k_i+2))+\ell+2} \left(\tau[\![S]\!]^{\ell} \circ \tau^R \circ \left(\bigcirc_{i=1}^{j}(\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R)\right) \circ \tau^{\bar{B}}\right) ; \qquad (33)$$

(This case is similar to E_n except that the execution of the loop terminates with condition B false. $F_0 = F_1 = \emptyset$ and $F_2 = \tau^R \circ \tau^{\bar{B}}$.)

$$G_n \stackrel{\text{def}}{=} (\tau[\![S]\!])^n ; \qquad (34)$$

(This case corresponds to the observation of $n \geq 1$ steps within the loop body S.).

We now proof (27) by recurrence on n. Given a formula $\mathcal{F}_n \in \{A_n, \ldots, F_n\}$ of the form $\mathcal{F}_n = \bigcup_{C(n,\ell,m,\ldots)} \mathcal{T}(n,\ell,m,\ldots)$, where $n,\ell,m,\ldots$ are free variables of the condition C and term $\mathcal{T}$, we write $\mathcal{F}_n \mid C'(n,\ell,m,\ldots)$ for the formula $\bigcup_{C(n,\ell,m,\ldots)\wedge C'(n,\ell,m,\ldots)} \mathcal{T}(n,\ell,m,\ldots)$.

— For the basis observe that for $n = 1$, $A_1 = \emptyset$, $B_1 = \tau^B$, $C_1 = \tau^{\bar{B}}$, $D_1 = \tau^R$, $E_1 = \emptyset$, $F_1 = \emptyset$ and $G_1 = (\tau[\![S]\!])^1 = \tau[\![S]\!]$ so that

$$\begin{aligned}
(\tau[\![C]\!])^1 &= \tau[\![C]\!] \\
&= \tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}} \\
&= B_1 \cup G_1 \cup D_1 \cup C_1 \\
&= A_1 \cup B_1 \cup C_1 \cup D_1 \cup E_1 \cup F_1 \cup G_1 .
\end{aligned}$$

— For $n = 2$, observe that $A_2 = \emptyset$, $B_2 = \tau^B \circ \tau[\![S]\!]$, $C_2 = \emptyset$, $D_2 = \tau[\![S]\!] \circ \tau^R$, $E_2 = \tau^R \circ \tau^B$, $F_2 = \tau^R \circ \tau^{\bar{B}}$ and $G_2 = (\tau[\![S]\!])^2$ so that

$$\begin{aligned}
(\tau[\![C]\!])^2 &= \tau^R \circ \tau^B \cup \tau^B \circ \tau[\![S]\!] \cup \tau[\![S]\!]^2 \cup \tau[\![S]\!] \circ \tau^R \cup \tau^R \circ \tau^{\bar{B}} \\
&= E_2 \cup B_2 \cup G_2 \cup D_2 \cup E_2 \cup F_2 \\
&= A_2 \cup B_2 \cup C_2 \cup D_2 \cup E_2 \cup F_2 \cup G_2 .
\end{aligned}$$

— For the induction step $n \geq 2$, we have to consider the compositions $A_n \circ \tau[\![C]\!]$, ..., $G_n \circ \tau[\![C]\!]$ in turn.

— $A_n \circ \tau[\![C]\!]$

$=$ $\wr$def. (26) of $\tau[\![C]\!]$$\int$

$A_n \circ (\tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}})$

$=$ $\wr$$\circ$ distributes over $\cup$, $n \geq 2$ so $j \geq 1$ whence $A_n = \tau' \circ \tau^R$, $\tau^R \circ \tau[\![S]\!] = \emptyset$ and $\tau^R \circ \tau^R = \emptyset$$\int$

$A_n \circ \tau^B \cup A_n \circ \tau^{\bar{B}}$

$=$ $\wr$def. (28) of A_n and $\tau[\![S]\!]^0 = 1_{\Sigma[\![P]\!]}$$\int$

$$\left(\bigcup_{n+1=(\sum_{i=1}^{j}(k_i+2))+1+0} \bigcirc_{i=1}^{j} (\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R) \right) \circ \tau^B \circ \tau[\![S]\!]^0$$

$$\cup \left(\bigcup_{n+1=(\sum_{i=1}^{j}(k_i+2))+1} \bigcirc_{i=1}^{j} (\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R) \right) \circ \tau^{\bar{B}}$$

$=$ $\wr$def. (29) of B_{n+1} with additional constraint $\ell = 0$ and def. (30) of $C_{n+1}$$\int$

$B_{n+1} \mid \ell = 0 \ \cup\ C_{n+1}$.

— $B_n \circ \tau[\![C]\!]$

$=$ $\wr$def. (26) of $\tau[\![C]\!]$$\int$

$B_n \circ (\tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}})$

$=$ $\wr$$\circ$ distributes over $\cup$, either $\ell = 0$ in B_n, in which case $B_n = \tau' \circ \tau^B$, $\tau^B \circ \tau^B = \emptyset$, $\tau^B \circ \tau^R = \emptyset$ and $\tau^B \circ \tau^{\bar{B}} = \emptyset$ or $\ell > 0$ in B_n, in which case $B_n = \tau'' \circ \tau[\![S]\!]$, $\tau[\![S]\!] \circ \tau^B = \emptyset$ and $\tau[\![S]\!] \circ \tau^{\bar{B}} = \emptyset$$\int$

$(B_n \mid \ell = 0) \circ \tau[\![S]\!] \cup (B_n \mid \ell > 0) \circ \tau[\![S]\!] \cup (B_n \mid \ell > 0) \circ \tau^R$

$=$ $\wr$def. (29) of $B_n$$\int$

$(B_{n+1} \mid \ell = 1) \cup (B_{n+1} \mid \ell > 1) \cup$

$$\left(\bigcup_{n=(\sum_{i=1}^{j}(k_i+2))+1+\ell} \left(\left(\bigcirc_{i=1}^{j} (\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R) \right) \circ \tau^B \circ \tau[\![S]\!]^{\ell} \right) \right) \circ \tau^R$$

$=$ $\wr$$\circ$ distributes over $\cup$$\int$

$(B_{n+1} \mid \ell = 1) \cup (B_{n+1} \mid \ell > 1) \cup$

$$\bigcup_{n+1=(\sum_{i=1}^{j}(k_i+2))+2+\ell} \left(\left(\bigcirc_{i=1}^{j} (\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R) \right) \circ (\tau^B \circ \tau[\![S]\!]^{\ell} \circ \tau^R) \right)$$

$=$ $\wr$by letting $k_{j+1} = \ell \geq 1$$\int$

$$(B_{n+1} \mid \ell = 1) \cup (B_{n+1} \mid \ell > 1) \cup \bigcup_{n+1=\sum_{i=1}^{j+1}(k_i+2)} \left(\bigcirc_{i=1}^{j+1} (\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R) \right)$$

$=$ $\wr$by letting $j' = j + 1$ and def. (29) of $A_{n+1}$$\int$

$(B_{n+1} \mid \ell = 1) \cup (B_{n+1} \mid \ell > 1) \cup A_{n+1}$

$=$ $\wr$associativity of $\cup$$\int$

$(B_{n+1} \mid \ell > 0) \cup A_{n+1}$.

— $C_n \circ \tau[\![C]\!]$

$=$ $\wr$def. (26) of $\tau[\![C]\!]$$\int$

$C_n \circ (\tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}})$

$=$ $\wr$$\circ$ distributes over $\cup$, $C_n = \tau' \circ \tau^{\bar{B}}$ and $\tau^{\bar{B}} \circ \tau^B = \tau^{\bar{B}} \circ \tau[\![S]\!] = \tau^{\bar{B}} \circ \tau^R = \tau^{\bar{B}} \circ \tau^{\bar{B}} = \emptyset$$\int$

$\emptyset$.

— $D_n \circ \tau[\![C]\!]$

$=$ $\wr$def. (26) of $\tau[\![C]\!]$$\int$

$D_n \circ (\tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}})$

$=$ $\wr$$\circ$ distributes over $\cup$, D_n has the form $\tau' \circ \tau^R$ and $\tau^R \circ \tau[\![S]\!] = \tau^R \circ \tau^R = \emptyset$$\int$

$D_n \circ \tau^B \cup D_n \circ \tau^{\bar{B}}$

$=$ $\wr$def. (32) of E_n and (33) of $F_n$$\int$

$(E_{n+1} \mid m = 0) \cup F_{n+1}$.

— $E_n \circ \tau[\![C]\!]$

$=$ $\wr$def. (26) of $\tau[\![C]\!]$$\int$

$E_n \circ (\tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}})$

$=$ $\wr$ $\circ$ distributes over $\cup$, $E_n \mid m = 0$ has the form $\tau' \circ \tau^B$ while $E_n \mid m >= 0$ has the form $\tau'' \circ \tau[\![S]\!]$, $\tau^B \circ \tau^B = \tau^B \circ \tau^R = \tau^B \circ \tau^{\bar{B}} = \emptyset$ and $\tau[\![S]\!] \circ \tau^B = \tau[\![S]\!] \circ \tau^{\bar{B}} = \emptyset$ $\wr$

$(E_n \mid m = 0) \circ \tau[\![S]\!] \cup (E_n \mid m > 0) \circ \tau[\![S]\!] \cup (E_n \mid m > 0) \circ \tau^R$

$=$ $\wr$ def. (32) of E_n and (31) of D_{n+1} where $k_i = m \geq 1$ so that $\ell < n$ $\wr$

$(E_{n+1} \mid m = 1) \cup (E_{n+1} \mid m > 1) \cup (D_{n+1} \mid \ell < n)$

$=$ $\wr$ $\cup$ is associative $\wr$

$(E_{n+1} \mid m > 0) \cup (D_{n+1} \mid \ell < n)$.

— $F_n \circ \tau[\![C]\!]$

$=$ $\wr$ def. (26) of $\tau[\![C]\!]$ $\wr$

$F_n \circ (\tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}})$

$=$ $\wr$ $\circ$ distributes over $\cup$, by def. (33) of F_n has the form $\tau' \circ \tau^{\bar{B}}$ and $\tau^{\bar{B}} \circ \tau^B = \tau^{\bar{B}} \circ \tau[\![S]\!] = \tau^{\bar{B}} \circ \tau^R = \tau^{\bar{B}} \circ \tau^{\bar{B}} = \emptyset$ $\wr$

$\emptyset$.

— $G_n \circ \tau[\![C]\!]$

$=$ $\wr$ def. (34) of G_n and (26) of $\tau[\![C]\!]$ $\wr$

$(\tau[\![S]\!])^n \circ (\tau^B \cup \tau[\![S]\!] \cup \tau^R \cup \tau^{\bar{B}})$

$=$ $\wr$ $\circ$ distributes over $\cup$, $n \geq 1$, $\tau[\![S]\!] \circ \tau^B = \tau[\![S]\!] \circ \tau^{\bar{B}} = \emptyset$ $\wr$

$(\tau[\![S]\!])^n \circ \tau[\![S]\!] \cup (\tau[\![S]\!])^n \circ \tau^R)$

$=$ $\wr$ def. $n+1$-th power and (31) of D_{n+1} $\wr$

$(\tau[\![S]\!])^{n+1} \cup (D_{n+1} \mid \ell = n)$.

Grouping all cases together, we get

$(\tau[\![C]\!])^{n+1}$

$=$ $\wr$ def. $n+1$-th power and (27) $\wr$

$(A_n \cup B_n \cup C_n \cup D_n \cup E_n \cup F_n \cup G_n) \circ (\tau[\![C]\!])^n$

$=$ $\wr$ $\circ$ distributes over $\cup$, def. (34) of G_n $\wr$

$(A_n \circ \tau[\![C]\!] \cup B_n \circ \tau[\![C]\!] \cup C_n \circ \tau[\![C]\!] \cup D_n \circ \tau[\![C]\!] \cup E_n \circ \tau[\![C]\!] \cup F_n \circ \tau[\![C]\!] \circ (\tau[\![C]\!])^n \circ \tau[\![C]\!]$

$=$ $\wr$ replacing according to the above lemmata $\wr$

$(B_{n+1} \mid \ell = 0 \;\cup\; C_{n+1}) \cup ((B_{n+1} \mid \ell > 0) \cup A_{n+1}) \cup \emptyset \cup ((E_{n+1} \mid m = 0) \cup F_{n+1}) \cup ((E_{n+1} \mid m > 0) \cup (D_{n+1} \mid \ell < n)) \cup \emptyset \cup ((\tau[\![S]\!])^{n+1} \cup (D_{n+1} \mid \ell = n))$

$=$ $\wr$ $\cup$ is associative and commutative and $(D_{n+1} \mid \ell > n) = \emptyset$ $\wr$

$A_{n+1} \cup B_{n+1} \cup C_{n+1} \cup D_{n+1} \cup E_{n+1} \cup F_{n+1} \cup G_{n+1}$

By recurrence on $n \geq 1$, we have proved that

$$(\tau[\![C]\!])^n \stackrel{\text{def}}{=} A_n \cup B_n \cup C_n \cup D_n \cup E_n \cup F_n \cup (\tau[\![C]\!])^n$$

so that

$\tau^\star[\![C]\!]$

$= (\tau[\![C]\!])^\star$

$= (\tau[\![C]\!])^0 \cup \bigcup_{n \geq 1} (A_n \cup B_n \cup C_n \cup D_n \cup E_n \cup F_n \cup (\tau[\![S]\!])^n)$

$= (\tau[\![C]\!])^0 \cup (\bigcup_{n \geq 1} A_n \cup \bigcup_{n \geq 1} B_n \cup \bigcup_{n \geq 1} C_n \cup \bigcup_{n \geq 1} D_n \cup \bigcup_{n \geq 1} E_n \cup \bigcup_{n \geq 1} F_n \cup (\tau[\![S]\!])^\star)$.

We now compute each of these terms.

$\bigcup_{n \geq 1} A_n$

$=$ $\wr$ def. (28) of A_n $\wr$

$\bigcup_{n \geq 1} \; \bigcup_{n = \sum_{i=1}^{j} (k_i + 2)} \; \bigcirc_{i=1}^{j} (\tau^B \circ \tau[\![S]\!]^{k_i} \circ \tau^R)$

$=$ $\wr$ for $n \in [1,3]$ this is $\emptyset$ while for $n > 3$, we can always find j and $k_1 \geq 1$, ...,$k_j \geq 1$ such that $n = \sum_{i=1}^{j} (k_i + 2)$. Reciprocally, for all choices of j and $k_1 \geq 1$, ...,$k_j \geq 1$ there exists an $n > 3$ such that $n = \sum_{i=1}^{j} (k_i + 2)$. $\wr$

$(\tau^B \circ \tau[\![S]\!]^+ \circ \tau^R)^+$

$=$ $\wr$ $\tau^B \circ \tau^R = \emptyset$ $\wr$

$$(\tau^B \circ \tau\llbracket S \rrbracket^\star \circ \tau^R)^+ .$$

By the same reasoning, we get

$$
\begin{aligned}
\bigcup_{n \geq 1} B_n &= (\tau^B \circ \tau\llbracket S \rrbracket^\star \circ \tau^R)^\star \circ \tau^B \circ \tau\llbracket S \rrbracket^\star, \\
\bigcup_{n \geq 1} C_n &= (\tau^B \circ \tau\llbracket S \rrbracket^\star \circ \tau^R)^\star \circ \tau^{\bar{B}}, \\
\bigcup_{n \geq 1} D_n &= \tau\llbracket S \rrbracket^\star \circ \tau^R \circ (\tau^B \circ \tau\llbracket S \rrbracket^\star \circ \tau^R)^\star, \\
\bigcup_{n \geq 1} E_n &= \tau\llbracket S \rrbracket^\star \circ \tau^R \circ (\tau^B \circ \tau\llbracket S \rrbracket^\star \circ \tau^R)^\star \circ \tau^B \circ \tau\llbracket S \rrbracket^\star, \\
\bigcup_{n \geq 1} F_n &= \tau\llbracket S \rrbracket^\star \circ \tau^R \circ (\tau^B \circ \tau\llbracket S \rrbracket^\star \circ \tau^R)^\star \circ \tau^{\bar{B}} .
\end{aligned}
$$

Grouping now all cases together and using the fact that $\circ$ distributes over $\cup$, we finally get

$$
\begin{aligned}
\tau^\star\llbracket C \rrbracket &= \tau\llbracket S \rrbracket^0 \cup (\tau^B \circ \tau\llbracket S \rrbracket^\star \circ \tau^R)^+ \cup (\tau^B \circ \tau\llbracket S \rrbracket^\star \circ \tau^R)^\star \circ \tau^B \circ \tau\llbracket S \rrbracket^\star \\
&\quad \cup (\tau^B \circ \tau\llbracket S \rrbracket^\star \circ \tau^R)^\star \circ \tau^{\bar{B}} \cup \tau\llbracket S \rrbracket^\star \circ \tau^R \circ (\tau^B \circ \tau\llbracket S \rrbracket^\star \circ \tau^R)^\star \\
&\quad \cup \tau\llbracket S \rrbracket^\star \circ \tau^R \circ (\tau^B \circ \tau\llbracket S \rrbracket^\star \circ \tau^R)^\star \circ \tau^B \circ \tau\llbracket S \rrbracket^\star \\
&\quad \cup \tau\llbracket S \rrbracket^\star \circ \tau^R \circ (\tau^B \circ \tau\llbracket S \rrbracket^\star \circ \tau^R)^\star \circ \tau^{\bar{B}} \\
&\quad \cup \tau\llbracket S \rrbracket^\star \\
&= (1_{\Sigma\llbracket P \rrbracket} \cup \tau\llbracket S \rrbracket^\star \circ \tau^R) \circ (\tau^B \circ \tau\llbracket S \rrbracket^\star \circ \tau^R)^\star \circ (1_{\Sigma\llbracket P \rrbracket} \cup \tau^B \circ \tau\llbracket S \rrbracket^\star \cup \tau^{\bar{B}}) \\
&\quad \cup \tau\llbracket S \rrbracket^\star .
\end{aligned}
$$

$\square$

Structural big-step operational semantics: sequence

$$\boxed{\tau^\star\llbracket C_1 \ ; \ \ldots \ ; \ C_n \rrbracket = \tau^\star\llbracket C_1 \rrbracket \circ \ldots \circ \tau^\star\llbracket C_n \rrbracket \quad (35)}$$

Proof. Let us recall from Lecture 5 that if $S = C_1 \ ; \ \ldots \ ; \ C_n$ where $n \geq 1$ is a sequence of commands and $\ell_i, \ell_{i+1} \in \text{in}_P\llbracket C_i \rrbracket$ for all $i \in [1, n]$, then

$$\frac{\langle \ell_i,\ \rho_i \rangle \overset{\llbracket C_i \rrbracket}{\Longrightarrow} \langle \ell_{i+1},\ \rho_{i+1} \rangle}{\langle \ell_i,\ \rho_i \rangle \overset{\llbracket C_1 \ ; \ \ldots \ ; \ C_n \rrbracket}{\Longrightarrow} \langle \ell_{i+1},\ \rho_{i+1} \rangle} . \tag{36}$$

We also have the labelling scheme

$$
\begin{aligned}
\text{at}_P\llbracket S \rrbracket &= \text{at}_P\llbracket C_1 \rrbracket, \\
\text{after}_P\llbracket S \rrbracket &= \text{after}_P\llbracket C_n \rrbracket, \\
\text{in}_P\llbracket S \rrbracket &= \bigcup_{i=1}^{n} \text{in}_P\llbracket C_i \rrbracket, \\
\forall i \in [1, n[&: \text{after}_P\llbracket C_i \rrbracket = \text{at}_P\llbracket C_{i+1} \rrbracket = \text{in}_P\llbracket C_i \rrbracket \cap \text{in}_P\llbracket C_{i+1} \rrbracket, \qquad (37) \\
\forall i, j \in [1, n] &: (j \neq i - 1 \wedge j \neq i + 1) \Longrightarrow (\text{in}_P\llbracket C_i \rrbracket \cap \text{in}_P\llbracket C_j \rrbracket = \emptyset) .
\end{aligned}
$$

1 —— Let S be the sequence $C_1 \ ; \ \ldots \ ; \ C_n$, $n \geq 1$, we first prove a lemma.

1.1 —— Let P be the program with subcommand $S = C_1 \ ; \ \ldots \ ; \ C_n$. Successive small steps in S must be made in sequence since, by the definition (4) and (36) of $\tau\llbracket S \rrbracket$ and the labelling scheme (37), it is impossible to jump from one command into a different one

$$
\begin{aligned}
&\tau^{k_1}\llbracket C_1 \rrbracket \circ \ldots \circ \tau^{k_n}\llbracket C_n \rrbracket = \qquad (38) \\
&\quad (\!| \forall i \in [1, n] : k_i = 0 \ ?\ 1_{\Sigma\llbracket P \rrbracket} \\
&\quad \| \ \exists 1 \leq i \leq j \leq n : \forall \ell \in [1, n] : (k_\ell \neq 0 \iff \ell \in [i, j]) \ ?\ \tau^{k_i}\llbracket C_i \rrbracket \circ \ldots \circ \tau^{k_j}\llbracket C_j \rrbracket \ :\ \emptyset |\!) .
\end{aligned}
$$

The proof is by recurrence on n.

1.1.1 —— If, for the basis, $n = 1$ then either $k_1 = 0$ and $\tau^0\llbracket C_1 \rrbracket = 1_{\Sigma\llbracket P \rrbracket}$ or $k_1 > 0$ and then $\tau^{k_1}\llbracket C_1 \rrbracket = \tau^{k_i}\llbracket C_i \rrbracket \circ \ldots \circ \tau^{k_j}\llbracket C_j \rrbracket$ by choosing $i = j = 1$.

1.1.2 —— For the induction step, assuming (38), we prove that

$$T = \tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \circ \tau^{k_{n+1}}[\![C_{n+1}]\!]$$

is of the form (38) with $n+1$ substituted for n. Two cases, with several subcases have to be considered.

1.1.2.1 — If $\forall i \in [1, n] : k_i = 0$ then we consider two subcases.

1.1.2.1.1 — If $k_{n+1} = 0$ then $\forall i \in [1, n+1] : k_i = 0$ and $T = \tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \circ \tau^{k_{n+1}}[\![C_{n+1}]\!] = 1_{\Sigma[\![P]\!]} \circ \tau^0[\![C_{n+1}]\!] = 1_{\Sigma[\![P]\!]}$.

1.1.2.1.2 — Otherwise $k_{n+1} > 0$ and then $\forall \ell \in [1, n+1] : (k_\ell \neq 0 \iff \ell \in [n+1, n+1])$ and $T = \tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \circ \tau^{k_{n+1}}[\![C_{n+1}]\!] = 1_{\Sigma[\![P]\!]} \circ \tau^{k_{n+1}}[\![C_{n+1}]\!] = \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j}[\![C_j]\!]$ by choosing $i = j = n+1$.

1.1.2.2 — Otherwise, $\exists i \in [1, n] : k_i \neq 0$.

1.1.2.2.1 — If $\exists 1 \leq i \leq j \leq n : \forall \ell \in [1, n] : (k_\ell \neq 0 \iff \ell \in [i, j])$ then by (38), we have

$$T = \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j}[\![C_j]\!] \circ \tau^{k_{n+1}}[\![C_{n+1}]\!] \ .$$

1.1.2.2.1.1 – If $k_{n+1} = 0$ then $\exists 1 \leq i \leq j \leq n+1 : \forall \ell \in [1, n+1] : (k_\ell \neq 0 \iff \ell \in [i, j])$ and:

$$\begin{aligned} T &= \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j}[\![C_j]\!] \circ \tau^{k_{n+1}}[\![C_{n+1}]\!], \\ &= \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j}[\![C_j]\!] \circ 1_{\Sigma[\![P]\!]}, \\ &= \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j}[\![C_j]\!] \ . \end{aligned}$$

1.1.2.2.1.2 – Otherwise $k_{n+1} > 0$ and we distinguish two subcases.

1.1.2.2.1.2.1 - If $j < n$ then $t^{k+1} = t \circ t^k = t^k \circ t$ so

$$T = \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j - 1}[\![C_j]\!] \circ\circ \tau[\![C_j]\!] \circ \tau[\![C_{n+1}]\!] \circ \tau^{k_{n+1}-1}[\![C_{n+1}]\!] \ .$$

By the definition (4) and (36) of $\tau[\![C]\!]$ and the labelling scheme (37), we have $\tau[\![C_j]\!] \circ \tau[\![C_{n+1}]\!] = \emptyset$ since $j < n$ so that in that case $T = \emptyset$.

1.1.2.2.1.2.2 - Otherwise $j = n$ so $\forall \ell \in [0, i[: k_\ell = 0$, $\forall \ell \in [i, n+1] : k_\ell > 0$ and $T = \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \circ \tau^{k_{n+1}}[\![C_{n+1}]\!]$ whence $\forall \ell \in [1, n+1] : (k_\ell \neq 0 \iff \ell \in [i, j])$ with $1 \leq i < j = n+1$ and $T = \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j}[\![C_j]\!]$.

1.1.2.2.2 — Otherwise $\forall 1 \leq i \leq j \leq n : \exists \ell \in [1, n] : (k_\ell \neq 0 \wedge \ell \notin [i, j]) \vee (\ell \in [i, j] \wedge k_\ell = 0)$.

1.1.2.2.2.1 – This excludes $n = 1$ since then $i = j = \ell = 1$ and $k_1 = 0$ in contradiction with $\exists i \in [1, n] : k_i \neq 0$.

1.1.2.2.2.2 – If $n = 2$ then $k_1 = 0$ and $k_2 > 0$ or $k_1 > 0$ and $k_2 = 0$ which corresponds to case 1.1.2.2.1, whence is impossible.

1.1.2.2.2.3 – So necessarily $n \geq 3$. Let $p \in [1, n]$ be minimal and $q \in [1, n]$ be maximal such that $k_p \neq 0$ ad $k_q \neq 0$. There exists $m \in [p, q]$ such that $k_m = 0$ since otherwise $k_\ell \neq 0$ and either $\ell < p$ in contradiction with the minimality of$p$ or $\ell > j$ in contradiction with the maximality of q. We have $p < m < q$ with $k_p \neq 0$, $k_m = 0$ and $k_j = 0$. Assume m to be minimal with that property, so that $k_{m-1} \neq 0$ and then that q' is the minimal q with this property so that $k_{q'-1} = 0$. We have $k_1 = 0, \ldots, k_{p-1} = 0$, $k_p \neq 0, \ldots, k_{m-1} = 0$, $k_m = 0$, $k_{q'-1} = 0$ $k_{q'} \neq 0, \ldots$ It follows, by the definition (4) and (36) of $\tau[\![C]\!]$ and the labelling scheme (37) that $\tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] = \emptyset$ that $T = \emptyset \circ \tau^{k_{n+1}}[\![C_{n+1}]\!] = \emptyset$.

It remains to prove that

$$\forall 1 \le i \le j \le n+1 : \exists \ell \in [1, n+1] : (k_\ell \neq 0 \wedge \ell \notin [i,j]) \vee (\ell \in [i,j] \wedge k_\ell = 0) .$$

1.1.2.2.2.3.1 - If $j < n+1$ then this follows from (38).

1.1.2.2.2.3.2 - Otherwise $j = n+1$ in which case either $k_{n+1} = 0$ and then we choose $\ell = j$ or $k_{n+1} > 0$ so that $q' = j = n+1$. If $j \le m$ then for $\ell = m$, we have $k_\ell = k_m = 0$. Otherwise $m < i \le q'$. Choosing $\ell = p$, we have $\ell \in [1,j]$ with $k_\ell = k_p \neq 0$.

1.2 — We will need a second lemma, stating that k small steps in C_1 ; ... ; C_n must be made in sequence with k_1 steps in C_1, followed by k_2 in C_2, ..., followed by k_n in C_n such that the total number $k_1 + \ldots + k_n$ of these steps is precisely k

$$\tau^k[\![C_1 \;;\; \ldots \;;\; C_n]\!] = \bigcup_{k=k_1+\ldots+k_n} \tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] . \qquad (39)$$

The proof is by recurrence on $k \ge 0$.

1.2.1 — For $k = 0$, we get $k_1 = \ldots = k_n = 0$ and $1_{\Sigma[\![P]\!]}$ on both sides of the equality.

1.2.2 — For $k = 1$, there must exist $m \in [1, n]$ such that $k_m = 1$ while for all $j \in [1,n] - \{m\}$, $k_j = 0$. By the definition (4) and (36) of $\tau[\![C_1 \;;\; \ldots \;;\; C_n]\!]$, we have

$$\tau[\![C_1 \;;\; \ldots \;;\; C_n]\!] = \bigcup_{m=1}^{n} \tau[\![C_m]\!] .$$

1.2.3 — For the induction step $k \ge 2$, we have

$$\begin{aligned}
& \tau^{k+1}[\![C_1 \;;\; \ldots \;;\; C_n]\!] \\
= \quad & \langle\text{def. } t^{k+1} = t^k \circ t \text{ of powers}\rangle \\
& \tau^{k}[\![C_1 \;;\; \ldots \;;\; C_n]\!] \circ \tau[\![C_1 \;;\; \ldots \;;\; C_n]\!] \\
= \quad & \langle\text{def. (4) and (36) of } \tau[\![C_1 \;;\; \ldots \;;\; C_n]\!]\rangle \\
& \tau^{k}[\![C_1 \;;\; \ldots \;;\; C_n]\!] \circ \bigcup_{m=1}^{n} \tau[\![C_m]\!] \\
= \quad & \langle\circ \text{ distributes over } \cup\rangle \\
& \bigcup_{m=1}^{n} \tau^{k}[\![C_1 \;;\; \ldots \;;\; C_n]\!] \circ \tau[\![C_m]\!] \\
= \quad & \langle\text{induction hypothesis (39)}\rangle \\
& \bigcup_{m=1}^{n} \left(\bigcup_{k=k_1+\ldots+k_n} \tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \right) \circ \tau[\![C_m]\!] \\
= \quad & \langle\circ \text{ distributes over } \cup\rangle
\end{aligned}$$

$$\begin{aligned}
& \bigcup_{k=k_1+\ldots+k_n} \bigcup_{m=1}^{n} \tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \circ \tau[\![C_m]\!] \\
\stackrel{\text{def}}{=} \quad & \langle\text{by definition}\rangle \\
& T .
\end{aligned}$$

1.2.3.1 — We first show that

$$T \subseteq \bigcup_{k+1=k'_1+\ldots+k'_n} \tau^{k'_1}[\![C_1]\!] \circ \ldots \circ \tau^{k'_n}[\![C_n]\!] .$$

According to lemma (38), three cases have to be considered for

$$t \stackrel{\text{def}}{=} \tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \circ \tau[\![C_m]\!] .$$

1.2.3.1.1 — The case $\forall i \in [1, n] : k_i = 0$ is impossible since then $k = \sum_{j=1}^{n} k_j = 0$ in contradiction with $k \ge 2$.

1.2.3.1.2 — Else if $\exists 1 \leq i \leq j \leq n : \forall \ell \in [1, n] : (k_\ell \neq 0 \iff \ell \in [i, j])$ then

$$t \stackrel{\text{def}}{=} \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j}[\![C_j]\!] \circ \tau[\![C_m]\!] \; .$$

We discriminate according to the value of m.

1.2.3.1.2.1 – If $m = j$, we get

$$\begin{aligned} t &= \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j+1}[\![C_j]\!], \\ &= \tau^{k'_1}[\![C_1]\!] \circ \ldots \circ \tau^{k'_n}[\![C_n]\!] \end{aligned}$$

with $k+1 = k'_1 + \ldots + k'_n$ where $k'_1 = 0, \ldots, k'_{i-1} = 0$, $k'_i = k_i$, $\ldots$, $k'_j = k_j + 1$, $k'_{j+1} = 0$, $\ldots$, $k'_n = 0$.

1.2.3.1.2.2 – If $m = j + 1$, we get

$$\begin{aligned} t &= \tau^{k_i}[\![C_i]\!] \circ \ldots \circ \tau^{k_j}[\![C_j]\!] \circ \tau^1[\![C_{j+1}]\!], \\ &= \tau^{k'_1}[\![C_1]\!] \circ \ldots \circ \tau^{k'_n}[\![C_n]\!] \end{aligned}$$

with $k+1 = k'_1 + \ldots + k'_n$ where $k'_1 = 0, \ldots, k'_{i-1} = 0$, $k'_i = k_i$, $\ldots$, $k'_j = k_j$, $k'_{j+1} = 1$, $k'_{j+2} = 0$, $\ldots$, $k'_n = 0$.

1.2.3.1.2.3 – Otherwise, by the definition (4) and (36) of $\tau[\![C]\!]$ and the labelling scheme (37), $\tau[\![C_j]\!] \circ \tau[\![C_m]\!] = \emptyset$ so that $T = \emptyset$ that is $t = \tau^{k'_1}[\![C_1]\!] \circ \ldots \circ \tau^{k'_n}[\![C_n]\!]$ with $k'_\ell = k_\ell$ for $\ell \in [1, n] - \{m\}$ and $k'_m = k_m + 1$.

1.2.3.1.3 — Otherwise $T = \emptyset$ so that the inclusion is trivial.

1.2.3.2 — Inversely, we now show that

$$\bigcup_{k+1 = k'_1 + \ldots + k'_n} \tau^{k'_1}[\![C_1]\!] \circ \ldots \circ \tau^{k'_n}[\![C_n]\!] \subseteq T \; .$$

According to lemma (38), three cases have to be considered for

$$t \stackrel{\text{def}}{=} \tau^{k'_1}[\![C_1]\!] \circ \ldots \circ \tau^{k'_n}[\![C_n]\!] \; .$$

1.2.3.2.1 — If $\forall i \in [1, n] : k'_i = 0$ then $k + 1 = \sum_{i=1}^{n} k'_i = 0$ which is impossible with $k \geq 0$.

1.2.3.2.2 — Else if $\exists 1 \leq i \leq j \leq n : \forall \ell \in [1, n] : (k'_\ell \neq 0 \iff \ell \in [i, j])$ then

$$t \stackrel{\text{def}}{=} \tau^{k'_i}[\![C_i]\!] \circ \ldots \circ \tau^{k'_j}[\![C_j]\!] \; .$$

with all $k'_i > 0$, $\ldots$, $k'_j > 0$.

1.2.3.2.2.1 – If $k'_j = 1$ then t is

$$t \stackrel{\text{def}}{=} \tau^{k'_i}[\![C_i]\!] \circ \ldots \circ \tau^{k'_{j-1}}[\![C_{j-1}]\!] \circ \tau^1[\![C_j]\!] \; .$$

so we choose $k_1 = 0$, $\ldots$, $k_{i-1} = 0$, $k_i = k'_i$, $\ldots$, $k_{j-1} = k'_{j-1}$, $k_j = 0$, $\ldots$, $k_n = 0$ and $m = j$ with $k + 1 = k'_1 + \ldots + k'_n$ whence $k = k_1 + \ldots + k_n$.

1.2.3.2.2.2 – Otherwise $k'_j > 1$ then we have t of the form required for T by choosing $k_1 = 0$, $\ldots$, $k_{i-1} = 0$, $k_i = k'_i$, $\ldots$, $k_j = k'_j - 1$, $k_{j+1} = 0$, $\ldots$, $k_n = 0$ and $m = j$ with $k + 1 = k'_1 + \ldots + k'_n$ whence $k = k_1 + \ldots + k_n$.

1.2.3.2.3 — Otherwise $t = \tau^{k'_1}[\![C_1]\!] \circ \ldots \circ \tau^{k'_n}[\![C_n]\!]$ is $\emptyset$ which is obviously included in T.

1.3 —— We can now consider the case 1 of the sequence $S = C_1 \; ; \ldots ; \; C_n$, $n \geq 1$

$\tau^\star[\![C_1 \; ; \ldots ; \; C_n]\!]$

$$
\begin{aligned}
= &\quad \wr\text{def. reflexive transitive closure}\wr \\
& \bigcup_{k\geq 0} \tau^k[\![C_1 \;;\; \ldots \;;\; C_n]\!] \\
= &\quad \wr\text{lemma (39)}\wr \\
& \bigcup_{k_1+\ldots+k_n\geq 0} \tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \\
= & \bigcup_{k_1\geq 0} \ldots \bigcup_{k_n\geq 0} \tau^{k_1}[\![C_1]\!] \circ \ldots \circ \tau^{k_n}[\![C_n]\!] \\
= &\quad \wr\circ \text{ distributes over } \cup\wr \\
& \left(\bigcup_{k_1\geq 0} \tau^{k_1}[\![C_1]\!]\right) \circ \ldots \circ \left(\bigcup_{k_n\geq 0} \tau^{k_n}[\![C_n]\!]\right) \\
= &\quad \wr\text{def. reflexive transitive closure}\wr \\
& \tau^\star[\![C_1]\!] \circ \ldots \circ \tau^\star[\![C_n]\!] \;.
\end{aligned}
$$

□

Structural big-step operational semantics: programs

$$\tau^\star[\![S \;;;]\!] = \tau^\star[\![S]\!] \;. \tag{40}$$

PROOF. Let us recall from Lecture 5 that for programs $P = S \;;;$, we have:

$$\frac{\langle \ell,\, \rho\rangle \Longmapsto[\![S]\!]\Longrightarrow \rho'}{\langle \ell,\, \rho\rangle \Longmapsto[\![S \;;;]\!]\Longrightarrow \langle \ell',\, \rho'\rangle} \;. \tag{41}$$

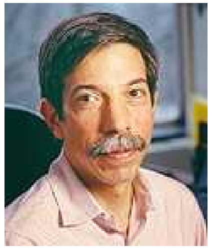
For programs $P = S \;;;$, we have

$$
\begin{aligned}
& \tau^\star[\![S \;;;]\!] \\
= &\quad \wr\text{def. reflexive transitive closure}\wr \\
& \bigcup_{k\geq 0} \tau^k[\![S \;;;]\!] \\
= &\quad \wr\text{by the definition (4) and (41) of } \tau[\![S \;;;]\!]\wr
\end{aligned}
$$

$$
\begin{aligned}
& \bigcup_{k\geq 0} \tau^k[\![S]\!] \\
= &\quad \wr\text{def. reflexive transitive closure}\wr \\
& \tau^\star[\![S]\!] \;.
\end{aligned}
$$

□

Classification of Program Trace Properties: Safety & Liveness



Fred B. Schneider

Reference

[4] Fred B. Schneider. "Decomposing Properties into Safety and Liveness using Predicate Logic'. Cornell University Computer Science Department Technical Report TR 87-874, October 1987.

Safety and Liveness, informally

- safety properties: informally, "bad things" cannot happen during program execution [5]
- liveness properties: informally, "good things" eventually do happen during program execution [5]);

Reference

[5] L. Lamport.
Proving the correctness of multiprocess programs. , I.E.E.E. Trans. on Software Engineering **SE-3**:2, p. 125–143, March 1977.

Safety

Trace properties [7]

- Σ: set of states
- Σ^∞: non-empty finite or infinite traces over states in Σ
- A trace property P is the set of traces which have that property so

$$P \in \wp(\Sigma^\infty)$$

[7] Recall that if the trace semantics is in $\wp(\Sigma^\infty)$ then properties of this semantics are in $\wp(\wp(\Sigma^\infty))$ whence not all of these properties can be expressed as trace properties. An example is "to be a deterministic program". Formally, if Σ is a set of states and Σ^∞ is the set of finite or infinite executions. Then "to be a deterministic program" is to have a single possible execution, indeed any one in Σ^∞. So the trace semantics of deterministic programs has the form $\{\sigma\}$ for any trace $\sigma \in \Sigma^\infty$. The corresponding property, that is the set of all such semantics is therefore $\{\{\sigma\} \mid \sigma \in \Sigma^\infty\}$. If we where to express this with some $P \subseteq \Sigma^\infty$ we would be allowed only $P = \{\sigma\}$ for some trace $\sigma \in \Sigma^\infty$. So we have to describe exactly the execution of the program while $\{\{\sigma\} \mid \sigma \in \Sigma^\infty\}$ allows us to state that this paricular execution trace is indeed unkown.

Definition of Prefix Closure

– The prefix closure $\mathsf{Pcl}(S)$ of a set $S \in \wp(\Sigma^{\infty})$ of nonempty traces, is the set of all nonempty finite *prefixes* (also called *left factors*) of traces in S

$$\mathsf{Pcl}(S) \stackrel{\text{def}}{=} \{\sigma \in \Sigma^{\vec{+}} \mid \exists \sigma' \in \Sigma^{\vec{\propto}} : \sigma \cdot \sigma' \in S\}$$

Properties of the prefix closure

– For finite sequences, Pcl is a *topological closure operator* on $\wp(\Sigma^{\vec{*}})$ and $\wp(\Sigma^{\vec{+}})$:

- $\mathsf{Pcl} \in \wp(\Sigma^{\vec{*}}) \mapsto \wp(\Sigma^{\vec{*}})$
- $\mathsf{Pcl} \in \wp(\Sigma^{\vec{+}}) \mapsto \wp(\Sigma^{\vec{+}})$
- $X \subseteq \mathsf{Pcl}(X)$ increasing/extensive
- $\mathsf{Pcl}(\mathsf{Pcl}(X)) = \mathsf{Pcl}(X)$ idempotent
- $\mathsf{Pcl}(X \cup Y) = \mathsf{Pcl}(X) \cup \mathsf{Pcl}(Y)$ [8] additive
- $\mathsf{Pcl}(\emptyset) = \emptyset$ $\emptyset$-preserving

[8] This implies $X \subseteq Y \Rightarrow \mathsf{Pcl}(X) \subseteq \mathsf{Pcl}(Y)$.

– For bifinitary sequences, Pcl satisfies:

- $\mathsf{Pcl} \in \wp(\Sigma^{\vec{\propto}}) \mapsto \wp(\Sigma^{\vec{*}})$
- $\mathsf{Pcl} \in \wp(\Sigma^{\vec{\infty}}) \mapsto \wp(\Sigma^{\vec{+}})$
- $X \not\subseteq \mathsf{Pcl}(X)$, when $X \cap \Sigma^{\vec{\omega}} \neq \emptyset$
- $\mathsf{Pcl}(\mathsf{Pcl}(X)) = \mathsf{Pcl}(X)$ idempotent
- $\mathsf{Pcl}(X \cup Y) = \mathsf{Pcl}(X) \cup \mathsf{Pcl}(Y)$ [9] additive
- $\mathsf{Pcl}(\emptyset) = \emptyset$ $\emptyset$-preserving

[9] This implies $X \subseteq Y \Rightarrow \mathsf{Pcl}(X) \subseteq \mathsf{Pcl}(Y)$.

Definition of Limit Closure

– The limit $\mathsf{Lim}(S)$ of a set $S \in \wp(\Sigma^{\infty})$ of nonempty traces, is the set S augmented with all infinite traces which have infinitely many finite prefixes in S

$$\mathsf{Lim}(S) \stackrel{\text{def}}{=} S \cup \{\sigma \in \Sigma^{\vec{\omega}} \mid \forall i : \exists j \geq i :\in \mathbb{N} : \sigma_0 \ldots \sigma_j \in S\}$$

Properties of the limit closure

– Lim is a topological closure operator on $\wp(\Sigma^{\vec{\infty}})$.

PROOF. – $X \subseteq \mathsf{Lim}(X)$ extensive

– $\mathsf{Lim}(X \cup Y) = \mathsf{Lim}(X) \cup \mathsf{Lim}(Y)$ additive [10]

– $\mathsf{Lim}(\mathsf{Lim}(X)) \;=\; \mathsf{Lim}(X)$ idempotent

– $\mathsf{Lim}(\emptyset) = \emptyset$ $\emptyset$-strict

$\square$

[10] Since any infinite sequence in $\mathsf{Lim}(X \cup Y)$ not in $X \cup Y$ has infinitely many different prefixes in $X \cup Y$. If there are finitely many in X then there are infinitely many in Y so the limit is in $\mathsf{Lim}(Y)$. Same if there are finitely many in Y then there are infinitely many in X so the limit is in $\mathsf{Lim}(X)$. If there are infinitely many in both X and Y then there milts are identical and in both $\mathsf{Lim}(X)$ and $\mathsf{Lim}(Y)$. So $\mathsf{Lim}((X \cup Y)) \subseteq \mathsf{Lim}(X) \cup \mathsf{Lim}(Y)$. The inverse is trivial.

Formal Definition of Safety

– $S \subseteq \Sigma^{\vec{\infty}}$ is a *safety* property if and only if [6]:

$$\mathsf{Safe}(S) = S$$

where:

$$\mathsf{Safe} \stackrel{\text{def}}{=} \mathsf{Lim} \circ \mathsf{PCl} \qquad (42)$$

i.e. S is closed by limits of prefixes

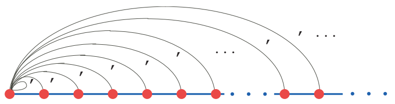
Reference

[6] B. Alpern & F.B. Schneider.
Defining Liveness. Information Processing Letters 21 (1985) 181–185.

Example of safety: invariance

– $\varphi \subseteq \Sigma \times \Sigma$ state relation

– $S_\varphi = \{\sigma \in \Sigma^{\vec{\infty}} \mid \forall i \in [0, |\sigma|[: \langle \sigma_0, \sigma_i \rangle \in \varphi\}$ invariance of φ



– $\mathsf{Lim} \circ \mathsf{PCl}(S_\varphi) = S_\varphi$ safety property

– $\tau^{\vec{\infty}} \subseteq S_\varphi$ if and only if all reachable states are linked to initial states by φ:

$$\tau^* \subseteq \varphi$$

Counter-example of safety property

– $\Sigma = \{a, b\}$

– $S = \{a\}^{\vec{+}} \cdot \{b\}^{\vec{*}}$
$\quad = \{a^n b^m \mid n > 0 \wedge m \geq 0\}$

– All traces in S have a finite number of a's followed by zero or more b's

– The infinite trace $\sigma = aaaaa \ldots$ is thus excluded from S

– Its impossible to discover this fact by observing finite prefixes of traces in S

– So S is <u>not</u> a safety property

PROOF. – $S = \{a\}^{\vec{+}} \cdot \{b\}^{\vec{*}}$

– $\mathsf{PCl}(S) = \{a\}^{\vec{+}} \cup \{a\}^{\vec{+}} \cdot \{b\}^{\vec{*}}$
$= \{a\}^{\vec{+}} \cdot \{b\}^{\vec{*}}$
$= S$

– $\mathsf{Lim}(\mathsf{PCl}(S)) = \mathsf{Lim}(S)$
$= S \cup \{a\}^{\vec{\omega}} \cup \{a\}^{\vec{+}} \cdot \{b\}^{\vec{\omega}}$
$= \{a\}^{\vec{\infty}} \cup \{a\}^{\vec{+}} \cdot \{b\}^{\vec{\propto}}$

– $S \neq \mathsf{Lim}(\mathsf{PCl}(S))$

$\square$

Prefix of a trace

$\sigma \swarrow n$ is the prefix of length $n \in \mathbb{N}$ of trace σ

– $\sigma \swarrow n = \sigma$ iff $|\sigma| \leq n$ [11]

– $\sigma \swarrow n = \sigma_0 \ldots \sigma_{n-1}$ iff $|\sigma| \geq n$

[11] Recall that $|\sigma|$ is the length of σ, ω if infinite.

Suffix of a trace

$\sigma \nearrow n$ is the suffix of trace σ beyond $n \in \mathbb{N}$

– $\sigma \nearrow n = \vec{\epsilon}$ if $n > |\sigma|$

– $\sigma \nearrow n = \sigma_n \ldots \sigma_{\ell-1}$ if $n \leq \ell = |\sigma| < \omega$

– $\sigma \nearrow n = \sigma_n \sigma_{n+1} \sigma_{n+2} \ldots$ if $|\sigma| = \omega$

Characterization of safety properties

Safety properties S can be disproved by looking only at some finite partial program behavior:

$$\forall \sigma \in \Sigma^{\vec{\infty}} : (\sigma \notin S) \iff (\exists i \geq 1 : \sigma \swarrow i \notin S)$$

PROOF. $\mathsf{Lim} \circ \mathsf{PCl}(S) = S$

$\iff \mathsf{Lim} \circ \mathsf{PCl}(S) \subseteq S$

$\iff \{\sigma \in \Sigma^{\vec{\infty}} \mid \forall i \geq 1 : \sigma \swarrow i \in \mathsf{PCl}(S)\} \subseteq S$

$\iff \{\sigma \in \Sigma^{\vec{\infty}} \mid \forall i \geq 1 : \exists \beta \in \Sigma^{\vec{\propto}} : \sigma \swarrow i \bullet \beta \in S\} \subseteq S$

$\iff \forall \sigma \in \Sigma^{\vec{\infty}} : (\forall i \geq 1 : \exists \beta \in \Sigma^{\vec{\propto}} : \sigma \swarrow i \bullet \beta \in S) \Longrightarrow (\sigma \in S)$

$\iff \forall \sigma \in \Sigma^{\vec{\infty}} : (\sigma \notin S) \Longrightarrow (\exists i \geq 1 : \forall \beta \in \Sigma^{\vec{\propto}} : \sigma \swarrow i \bullet \beta \notin S)$

$\Longleftrightarrow\ \forall \sigma \in \Sigma^{\vec{\infty}} : (\sigma \notin S) \Longleftrightarrow (\exists i \geq 1 : \forall \beta \in \Sigma^{\vec{\propto}} : \sigma\swarrow i \bullet \beta \notin S)$

since if $\exists i \geq 1 : \forall \beta \in \Sigma^{\vec{\propto}} : \sigma\swarrow i \bullet \beta \notin S$ then in particular for $\beta = \sigma\nearrow n$, we have $\sigma = \sigma\swarrow i \bullet \sigma\nearrow n \notin S$.

$\Longleftrightarrow\ \forall \sigma \in \Sigma^{\vec{\infty}} : (\sigma \notin S) \Longleftrightarrow (\exists i \geq 1 : \sigma\swarrow i \notin S)$ [12]

since $\forall \beta \in \Sigma^{\vec{\propto}} : \sigma\swarrow i \bullet \beta \notin S \Longleftrightarrow \sigma\swarrow i \notin S$

- $\Rightarrow$ choose $\beta = \vec{\epsilon}$
- $\Leftarrow$ S is a safety property so $\mathsf{PCl}(S) = S$ hence $(\sigma\swarrow i \bullet \beta \in S) \Longrightarrow (\sigma\swarrow i \in S)$ so $(\sigma\swarrow i \notin S) \Longrightarrow (\sigma\swarrow i \bullet \beta \notin S)$.

$\square$

[12] This corresponds to the usual explanation of safety: if a "bad thing" does occur (i.e. $\sigma \notin S$) then this can be recognized in finite time. Otherwise stated, there is a finite observation where something undesired happened which is irremediable, because it cannot be fixed in the future no matter how it is extended.

Liveness

Definition of Liveness

– $S \subseteq \Sigma^{\vec{\infty}}$ is a *liveness* property if and only if [7]:

$$\begin{array}{rrcl} & \mathsf{Lim} \circ \mathsf{PCl}(S) & = & \Sigma^{\vec{\infty}} \text{ [13]} \\ \Longleftrightarrow & \mathsf{PCl}(S) & = & \Sigma^{\vec{+}} \\ \Longleftrightarrow & \Sigma^{\vec{+}} & \subseteq & \mathsf{PCl}(S) \end{array}$$

Reference

[7] B. Alpern & F.B. Schneider. *Defining Liveness.* Information Processing Letters 21 (1985) 181–185.

[13] Otherwise stated S is *dense* in the topology induced by the topological closure operator $\mathsf{Lim} \circ \mathsf{PCl}$ which fixpoints are the closed sets.

Examples of liveness properties

- $\Sigma = \{a, b\}$, $\quad \Sigma^{\vec{*}} \bullet \{b\}$
- $\Sigma = \{a, b, c\}$, $\quad \{a\}^{\vec{*}} \bullet \{b, c\} \bullet \Sigma^{\vec{\omega}}$
- $\Sigma = \{a, b\}$, $\quad \{a\} \bullet \Sigma^{\vec{*}} \bullet \{aa\} \bullet \Sigma^{\vec{\omega}} \cup \{b\} \bullet \Sigma^{\vec{*}} \bullet \{bb\} \bullet \Sigma^{\vec{\omega}}$

Characterization 1 of liveness

– Proving liveness properties S imperatively requires the consideration of infinite behaviors:

$$\forall \alpha \in \Sigma^{\vec{+}} : \exists \beta \in \Sigma^{\vec{\propto}} : \alpha \bullet \beta \in S$$

Proof. $\mathsf{Lim} \circ \mathsf{Pcl}(S) = \Sigma^{\vec{\infty}}$

$\Longleftrightarrow \quad \Sigma^{\vec{\infty}} \subseteq \mathsf{Lim} \circ \mathsf{Pcl}(S)$

$\Longleftrightarrow \quad \Sigma^{\vec{\infty}} \subseteq \{\sigma \in \Sigma^{\vec{\infty}} \mid \forall i \geq 1 : \sigma{\swarrow}i \in \mathsf{Pcl}(S)\}$

$\Longleftrightarrow \quad \Sigma^{\vec{\infty}} \subseteq \{\sigma \in \Sigma^{\vec{\infty}} \mid \forall i \geq 1 : \exists \beta \in \Sigma^{\vec{\propto}} : \sigma{\swarrow}i \bullet \beta \in S\}$

$\Longleftrightarrow \quad \forall \sigma \in \Sigma^{\vec{\infty}} : \forall i \geq 1 : \exists \beta \in \Sigma^{\vec{\propto}} : \sigma{\swarrow}i \bullet \beta \in S$

$\Longleftrightarrow \quad \forall \alpha \in \Sigma^{\vec{+}} : \exists \beta \in \Sigma^{\vec{\propto}} : \alpha \bullet \beta \in S$ [14]

□

[14] A liveness property stipulates that a “good thing” eventually happens. For a liveness property, no partial execution is irremediable: it always remains possible for the “good thing” required by the liveness property (termination, receiving service, progress of a computation) to happen in the future. So disproving liveness requires considering all possible infinite executions.

Example of Liveness Property: Termination

– $L = \Sigma^{\vec{+}}$ termination

– $\mathsf{Pcl}(L) = \mathsf{Pcl}(\Sigma^{\vec{+}}) = \Sigma^{\vec{+}}$ liveness property

– $\tau^{\vec{\infty}} \subseteq L \iff \tau^{\vec{\infty}} \subseteq \Sigma^{\vec{+}} \iff \tau^{\vec{\omega}} = \emptyset$ termination (there is no possible infinite execution).

A liveness property cannot be checked by a program during its execution so liveness is inobservable at execution.

Dual Limit

$$\widetilde{\mathsf{Lim}}(P) \stackrel{\mathrm{def}}{=} \neg(\mathsf{Lim}(\neg P))$$

– $\widetilde{\mathsf{Lim}}(P) = \neg\{\sigma \in \Sigma^{\vec{\infty}} \mid \forall i \in \mathbb{N}_+ : \sigma{\swarrow}i \in \neg P\}$

$= \{\sigma \in \Sigma^{\vec{\infty}} \mid \exists i \in \mathbb{N}_+ : \sigma{\swarrow}i \in P\}$

so that $P \subseteq \widetilde{\mathsf{Lim}}(P)$ since whenever $\sigma \in P$, we have $\sigma{\swarrow}|\sigma| = \sigma \in P$ proving:

$$\mathsf{Pcl} \circ \widetilde{\mathsf{Lim}} \text{ is extensive}$$

since $P \subseteq \mathsf{Pcl}(P) \subseteq \mathsf{Pcl}(\widetilde{\mathsf{Lim}}(P))$ follows from extensivity and monotonicity of Pcl.

Characterization 2 of liveness

If we define:

$$\mathsf{Live}(P) \stackrel{\text{def}}{=} \neg\mathsf{Safe}(P) \cup P$$

then

> $P \subseteq \Sigma^{\vec{\infty}}$ is a liveness property if and only if $\mathsf{Live}(P) = P$.

Proof. – $\mathsf{Live}(P)$ is a liveness property: (43)

$\Sigma^{\vec{+}}$

$= \mathsf{PCl}(P) \cup \neg\mathsf{PCl}(P)$

$\subseteq \mathsf{PCl}(P) \cup \mathsf{PCl} \circ \widetilde{\mathsf{Lim}}(\neg\mathsf{PCl}(P))$ — $\mathsf{PCl} \circ \widetilde{\mathsf{Lim}}$ is extensive

$= \mathsf{PCl}(P \cup \widetilde{\mathsf{Lim}}(\neg\mathsf{PCl}(P)))$ — PCl is a complete join morphism

$= \mathsf{PCl}(P \cup \neg\mathsf{Lim}\neg \circ \neg\mathsf{PCl}(P))$ — def. $\widetilde{\mathsf{Lim}}$

$= \mathsf{PCl}(P \cup \neg\mathsf{Lim} \circ \mathsf{PCl}(P))$

$= \mathsf{PCl}(P \cup \neg\mathsf{Safe}(P))$ — def. Safe

$= \mathsf{PCl}(\mathsf{Live}(P))$ — def. Live, Q.E.D.

- $\mathsf{Live}(P)$ is a liveness property so if $\mathsf{Live}(P) = P$ then P is also a liveness property;
- Reciprocally, if P is a liveness property then $\Sigma^{\vec{+}} \subseteq \mathsf{PCl}(P)$ hence $\Sigma^{\vec{\infty}} = \mathsf{Lim}(\Sigma^{\vec{+}}) \subseteq \mathsf{Lim} \circ \mathsf{PCl}(P) = \mathsf{Safe}(P)$ whence $\neg\mathsf{Safe}(P) = \emptyset$ so that $\mathsf{Live}(P) = \neg\mathsf{Safe}(P) \cup P = \emptyset \cup P = P$;

$\square$

An immediate consequence is that Live is extensive and idempotent. However it is not monotonic ($\emptyset \subseteq \Sigma^{\vec{+}}$ but $\mathsf{Live}(\emptyset) = \neg\mathsf{Safe}(\emptyset) \cup \emptyset = \neg\emptyset\ \Sigma^{\vec{\infty}} \not\subseteq \mathsf{Live}(\Sigma^{\vec{+}}) = \neg\mathsf{Safe}(\Sigma^{\vec{+}}) \cup \Sigma^{\vec{+}} = \neg\Sigma^{\vec{\infty}} \cup \Sigma^{\vec{+}} = \emptyset \cup \Sigma^{\vec{+}} = \Sigma^{\vec{+}}$). This also shows that $\mathsf{Live}(P)$ may not be the least liveness property including P (since $\emptyset \subseteq \Sigma^{\vec{+}} = \mathsf{Live}(\Sigma^{\vec{+}})$ but $\mathsf{Live}(\emptyset) \not\subseteq \Sigma^{\vec{+}}$) [15].

[15] In contradiction with the claim on bottom of page 157 of [9].

Decomposition into Safety and Liveness

- Any program property P can be decomposed into the conjunction of a safety and a liveness property [8]:
  - $\mathsf{Safe}(P)$ — safety property
  - $\mathsf{Live}(P)$ — liveness property
  - $P = \mathsf{Safe}(P) \cap \mathsf{Live}(P)$

Proof. $\mathsf{Safe}(P) \cap \mathsf{Live}(P) = (\neg\mathsf{Safe}(P) \cup P) \cap \mathsf{Safe}(P) = (\neg\mathsf{Safe}(P) \cap \mathsf{Safe}(P)) \cup (P \cap \mathsf{Safe}(P)) = P \cap \mathsf{Safe}(P) = P$ since Safe is extensive. $\square$

Reference

[8] B. Alpern & F.B. Schneider. *Defining Liveness.* Information Processing Letters 21 (1985) 181–185.

A Simple Example [9]

- $\Sigma = \{a, b, c\}$
- $S = \{a\}^{\vec{*}} \bullet \{b\} \bullet \Sigma^{\dot{\vec{\infty}}}$
- $\mathsf{Safe}(S) = \{a\}^{\dot{\vec{\infty}}} \cup \{a\}^{\vec{*}} \bullet \{b\} \bullet \Sigma^{\dot{\vec{\infty}}}$
- $\mathsf{Live}(S) = \{a\}^{\vec{*}} \bullet \{b, c\} \bullet \Sigma^{\dot{\vec{\infty}}}$
- $S = \mathsf{Safe}(S) \cap \mathsf{Live}(S)$

Reference

[9] E. Chang, Z. Manna & A. Pnueli. "*The Safety-Progress Classification*". In *Logic and Algebra of Specifications*, F.L. Bauer, W. Brauer & H. Schwichtengerg (Eds.), NATO Advanced Science Institutes Series, Springer-Verlag, pages 143–202, 1991.

Example: decomposition of total correctness into partial correctness (safety) and termination (liveness)

- Total correctness specification $\langle \Upsilon, \Psi \rangle$:
  - $\Upsilon \subseteq \Sigma$ initial states
  - $\Psi \subseteq \Upsilon \times \Xi$ partial correctness relation
  - $\Xi \stackrel{\text{def}}{=} \check{\tau} \subseteq \Sigma$ final/blocking states
  - $\Phi \stackrel{\text{def}}{=} \{\langle s, s' \rangle \mid (s \in \Upsilon) \Longrightarrow (s' \in \Xi \wedge \langle s, s' \rangle \in \Psi\}$ total correctness relation

- Total correctness proof:
  $\forall \sigma \in \tau^{\check{\vec{\infty}}} : \exists i < |\sigma| : \langle \sigma_0, \sigma_i \rangle \in \Phi$ inevitability of Φ
- Partial correctness proof:
  $\forall s, s' \in \Sigma : (s \in \Upsilon \wedge \langle s, s' \rangle \in \tau^* \wedge s' \in \Xi) \Longrightarrow (\langle s, s' \rangle \in \Psi)\}$
- Termination proof:

$$\begin{aligned} & \forall \sigma \in \tau^{\check{\vec{\infty}}} : (\sigma_0 \in \Upsilon) \Longrightarrow (\exists i < |\sigma| : \sigma_i \in \check{\tau}) \\ \Longleftrightarrow\ & \forall \sigma \in \tau^{\check{\vec{\infty}}} : (\sigma_0 \in \Upsilon) \Longrightarrow (\sigma \in \Sigma^{\vec{+}}) \\ \Longleftrightarrow\ & \tau^{\check{\vec{\infty}}} \subseteq \{\sigma \in \Sigma^{\vec{+}} \mid \sigma_0 \in \Upsilon\} \cup \{\sigma \in \Sigma^{\vec{\infty}} \mid \sigma_0 \notin \Upsilon\} \end{aligned}$$

which is a liveness property, since:

$$\Sigma^{\vec{+}} \subseteq \{\sigma \in \Sigma^{\vec{+}} \mid \sigma_0 \in \Upsilon\} \cup \{\sigma \in \Sigma^{\vec{\infty}} \mid \sigma_0 \notin \Upsilon\}$$

PROOF. – $\forall \sigma \in \tau^{\check{\vec{\infty}}}, (\sigma_0 \in \Upsilon)$ initial states hypothesis

$\Longrightarrow \sigma_0 \in \Upsilon \wedge \sigma \in \tau^{\check{\vec{\infty}}} \wedge \sigma \in \Sigma^{\vec{+}}$ by termination

$\Longrightarrow \sigma_0 \in \Upsilon \wedge \sigma \in \tau^{\check{\vec{+}}}$ by def. $\tau^{\check{\vec{\infty}}}$

$\Longrightarrow \sigma_0 \in \Upsilon \wedge \langle \sigma_0, \sigma_{|\sigma|-1} \rangle \in \tau^\star \wedge \sigma_{|\sigma|-1} \in \check{\tau}$ by def. $\tau^{\check{\vec{+}}}$ and $\tau^\star$

$\Longrightarrow \sigma_0 \in \Upsilon \wedge \langle \sigma_0, \sigma_{|\sigma|-1} \rangle \in \tau^\star \wedge \sigma_{|\sigma|-1} \in \Xi$ by def. Ξ

$\Longrightarrow \langle \sigma_0, \sigma_{|\sigma|-1} \rangle \in \Psi$ by partial correctness

$\Longrightarrow \exists i < |\sigma| : \langle \sigma_0, \sigma_i \rangle \in \Psi$ proving total correctness

$\square$

THE END

My MIT web site is http://www.mit.edu/~cousot/

The course web site is http://web.mit.edu/afs/athena.mit.edu/course/16/16.399/www/.