« Reachability and Postcondition 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/

THESE

présentée à

Université Scientifique et Médicale de Grenoble
Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR ES SCIENCES MATHEMATIQUES

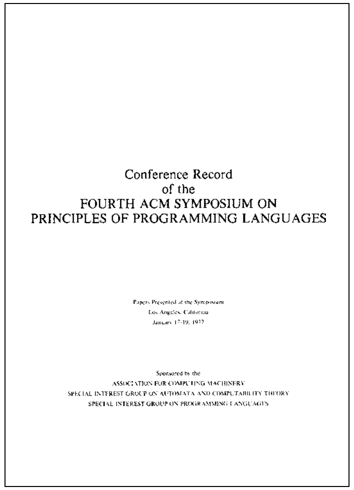
par

Patrick COUSOT

METHODES ITERATIVES DE CONSTRUCTION
ET D'APPROXIMATION DE PCINTS FIXES
D'OPERATEURS MONOTONES SUR UN TREILLIS,
ANALYSE SEMANTIQUE DES PROGRAMMES.

Thèse soutenue le 21 mars 1978 devant la Commission d'Examen :

Président : L. BOLLIET
Examinateurs : C. BENZAKEN
Ph. JORRAND
B. LORHO
C. PAIR
F. ROBERT
M. SINTZOFF



Conference Record
of the
FOURTH ACM SYMPOSIUM ON
PRINCIPLES OF PROGRAMMING LANGUAGES

Papers Presented at the Symposium
Los Angeles, California
January 17-19, 1977

Sponsored by the
ASSOCIATION FOR COMPUTING MACHINERY
SPECIAL INTEREST GROUP ON AUTOMATA AND COMPUTABILITY THEORY
SPECIAL INTEREST GROUP ON PROGRAMMING LANGUAGES

Forward collecting semantics of arithmetic expressions

Definition of the forward collecting semantics of arithmetic expressions

Recall the *forward/bottom-up collecting semantics* of an arithmetic expression from lecture 8:

$$\text{Faexp} \in \text{Aexp} \mapsto \wp(\text{Env}[\![P]\!]) \stackrel{\sqcup}{\longmapsto} \wp(\mathbb{I}_\Omega),$$

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

such that:

$$\text{Faexp}[\![A]\!]\left(\bigcup_{k \in S} R_k\right) = \bigcup_{k \in S} (\text{Faexp}[\![A]\!]R_k)$$

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

Structural specification of the forward collecting semantics of arithmetic expressions

$$\mathrm{Faexp}[\![\mathtt{n}]\!]R = \{\underline{\mathtt{n}}\}$$ [1]

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

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

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

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

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

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

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

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

Forward collecting semantics of boolean expressions

Definition of the forward collecting semantics of boolean expressions

Recall the *collecting semantics* $\mathrm{Cbexp}[\![B]\!]R$ of a boolean expression B from lecture 8:

$$\mathrm{Cbexp} \in \mathrm{Bexp} \mapsto \wp(\mathrm{Env}[\![P]\!]) \stackrel{\sqcup}{\longmapsto} \wp(\mathrm{Env}[\![P]\!]),$$

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

such that:

$$\mathrm{Cbexp}[\![B]\!]\left(\bigcup_{k \in S} R_k\right) = \bigcup_{k \in S} (\mathrm{Cbexp}[\![B]\!]R_k)$$

$$\mathrm{Cbexp}[\![B]\!]\emptyset = \emptyset\ .$$

Structural specification of the forward collecting semantics of boolean expressions

$$\mathrm{Cbexp}[\![\mathtt{true}]\!]R = R$$

$$\mathrm{Cbexp}[\![\mathtt{false}]\!]R = \emptyset$$

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

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

$$\mathrm{Cbexp}[\![B_1\ \&\ B_2]\!]R = \mathrm{Cbexp}[\![B_1]\!]R \cap \mathrm{Cbexp}[\![B_2]\!]R$$

$$\mathrm{Cbexp}[\![B_1 \mid B_2]\!]R = (\mathrm{Cbexp}[\![B_1]\!]R \cap (\mathrm{Cbexp}[\![B_2]\!]R \cup \mathrm{Cbexp}[\![T(\neg B_2)]\!]R)) \cup (\mathrm{Cbexp}[\![B_2]\!]R \cap (\mathrm{Cbexp}[\![B_1]\!]R \cup \mathrm{Cbexp}[\![T(\neg B_1)]\!]R))$$

Big-step operational semantics of commands

Definition of the big-step operational semantics of commands

Recall the the big-step operational semantics of commands defined in course 8 as

$$\tau^\star[\![C]\!] \stackrel{\text{def}}{=} (\tau[\![C]\!])^\star.$$

where $\tau[\![C]\!]$ is the small-step operational semantics of the program components $C \in \text{Cmp}[\![P]\!]$ of program P.

Structural big-step operational semantics

$$\tau^\star[\![\texttt{skip}]\!] = 1_{\Sigma[\![P]\!]} \cup \tau[\![\texttt{skip}]\!] \tag{3}$$

where:

$$\tau[\![\texttt{skip}]\!] = \{\langle\langle \text{at}_P[\![\texttt{skip}]\!],\ \rho\rangle,\ \langle \text{after}_P[\![\texttt{skip}]\!],\ \rho\rangle\rangle \mid \rho \in \text{Env}[\![P]\!]\}$$

$$\tau^\star[\![\texttt{X := } A]\!] = 1_{\Sigma[\![P]\!]} \cup \tau[\![\texttt{X := } A]\!] \tag{4}$$

where:

$$\tau[\![\texttt{X := } A]\!] = \{\langle\langle \text{at}_P[\![\texttt{X := } A]\!],\ \rho\rangle,\ \langle \text{after}_P[\![\texttt{X := } A]\!],\ \rho[\texttt{X} := i]\rangle\rangle \mid i \in \mathbb{I} \wedge \rho \vdash A \Rrightarrow i\}$$

$$\tau^\star[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!] = (1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t) \ \cup\ (1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^f) \tag{5}$$

where:

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

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

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

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

$$\tau^\star[\![\texttt{while } B \texttt{ do } S \texttt{ od}]\!] = \qquad (6)$$
$$((1_{\Sigma[\![P]\!]} \cup \tau^\star[\![S]\!] \circ \tau^R) \circ (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \circ$$
$$(1_{\Sigma[\![P]\!]} \cup \tau^B \circ \tau^\star[\![S]\!] \cup \tau^{\bar{B}})) \cup \tau[\![S]\!]^\star$$

where:

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

$$\tau^\star[\![C_1 \ ; \ \ldots \ ; \ C_n]\!] = \tau^\star[\![C_1]\!] \circ \ldots \circ \tau^\star[\![C_n]\!] \qquad (7)$$

$$\tau^\star[\![S \ ;;]\!] = \tau^\star[\![S]\!] \ . \qquad (8)$$

Postcondition collecting semantics of commands

Definition of the postcondition semantics of commands

The postcondition semantics $\text{Pcom}[\![C]\!]$ of a command $C \in \text{Com}$ (of a given program P) specifies the strongest postcondition $\text{Pcom}[\![C]\!]R$ satisfied by environments resulting from the execution of the command C starting in any of the environments satisfying the precondition R, if and when this execution terminates.

$$\text{Pcom} \in \text{Com} \mapsto \wp(\text{Env}[\![P]\!]) \stackrel{\sqcup}{\longmapsto} \wp(\text{Env}[\![P]\!]) \qquad (9)$$
$$\text{Pcom}[\![C]\!]R \stackrel{\text{def}}{=} \{\rho' \mid \exists \rho \in R : \langle\langle \text{at}_P[\![C]\!],\ \rho\rangle,\ \langle \text{after}_P[\![C]\!],\ \rho'\rangle\rangle \in \tau^\star[\![C]\!]\}$$

The postcondition semantics of a command can be understood, up to an interpretation, as a predicate transformer.

Property of the postcondition semantics of commands

The postcondition semantics of a command is a complete join morphism (denoted by $\stackrel{\sqcup}{\longmapsto}$) that is ($\mathcal{S}$ is an arbitrary set):

$$\text{Pcom}[\![C]\!]\left(\bigcup_{k \in \mathcal{S}} R_k\right) = \bigcup_{k \in \mathcal{S}} (\text{Pcom}[\![C]\!]R_k)$$

which implies monotony, continuity and strictness:

$$\text{Pcom}[\![C]\!]\emptyset = \emptyset \ .$$

Structural specification of the postcondition semantics of commands

$$
\begin{aligned}
\text{Pcom}[\![\texttt{skip}]\!]R &= R\\
\text{Pcom}[\![\texttt{X} := A]\!]R &= \{\rho[\texttt{X} := i] \mid \rho \in R \wedge i \in (\text{Faexp}[\![A]\!]\{\rho\}) \cap \mathbb{I}\}\\
\text{Pcom}[\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!]R &= \\
&\quad \text{Pcom}[\![S_t]\!](\text{Cbexp}[\![B]\!]R) \cup \text{Pcom}[\![S_f]\!](\text{Cbexp}[\![T(\neg(B))]\!]R)\\
\text{Pcom}[\![\texttt{while } B \texttt{ do } S \texttt{ od}]\!]R &= \\
&\quad \text{let } I = \text{lfp}_{\emptyset}^{\subseteq}\, \lambda X \cdot R \cup \text{Pcom}[\![S]\!](\text{Cbexp}[\![B]\!]X) \text{ in}\\
&\qquad \text{Cbexp}[\![T(\neg(B))]\!]I)\\
\text{Pcom}[\![C \;;\; S]\!]R &= (\text{Pcom}[\![S]\!] \circ \text{Pcom}[\![C]\!])R\\
\text{Pcom}[\![S \;;;]\!]R &= \text{Pcom}[\![S]\!]
\end{aligned}
$$

Postcondition semantics as an abstraction of the big-step operational semantics

If we let

$$
\begin{aligned}
\alpha[\![C]\!] &\in \wp(\Sigma[\![P]\!] \times \Sigma[\![P]\!]) \mapsto (\wp(\text{Env}[\![P]\!]) \mapsto \wp(\text{Env}[\![P]\!]))\\
\alpha[\![C]\!]X &= \lambda R \cdot \{\rho' \mid \exists \rho \in R : \langle\langle \text{at}_P[\![C]\!],\, \rho\rangle,\, \langle \text{after}_P[\![C]\!],\, \rho'\rangle\rangle \in X\}
\end{aligned}
$$

then $\text{Pcom}[\![C]\!] = \alpha[\![C]\!](\tau^\star[\![C]\!])$ is an abstract interpretation of the big-step operational semantics $\tau^\star[\![C]\!]$.

Proof. – We observe that $\alpha[\![C]\!]$is a complete join morphism

$$
\begin{aligned}
&\alpha[\![C]\!](\bigcup_{i \in \Delta} X_i)\\
=\;& \lambda R \cdot \{\rho' \mid \exists \rho \in R : \langle\langle \text{at}_P[\![C]\!],\, \rho\rangle,\, \langle \text{after}_P[\![C]\!],\, \rho'\rangle\rangle \in \bigcup_{i \in \Delta} X_i\}\\
=\;& \lambda R \cdot \bigcup_{i \in \Delta} \{\rho' \mid \exists \rho \in R : \langle\langle \text{at}_P[\![C]\!],\, \rho\rangle,\, \langle \text{after}_P[\![C]\!],\, \rho'\rangle\rangle \in X_i\}\\
=\;& \dot{\bigcup_{i \in \Delta}} \lambda R \cdot \{\rho' \mid \exists \rho \in R : \langle\langle \text{at}_P[\![C]\!],\, \rho\rangle,\, \langle \text{after}_P[\![C]\!],\, \rho'\rangle\rangle \in X_i\}\\
=\;& \dot{\bigcup_{i \in \Delta}} \alpha[\![C]\!] X_i
\end{aligned}
$$

so that it is a lower-adjoint of a Galois connection

$$
\langle \wp(\Sigma[\![P]\!] \times \Sigma[\![P]\!]),\, \subseteq\rangle \underset{\alpha[\![C]\!]}{\overset{\gamma[\![C]\!]}{\leftrightarrows}} \langle \wp(\text{Env}[\![P]\!]) \mapsto \wp(\text{Env}[\![P]\!]),\, \dot{\subseteq}\rangle
$$

for the pointwise extension $\dot{\subseteq}$ of $\subseteq$.

– We proceed by structural induction on the structure of programs.

$$
\begin{aligned}
-\;& \text{Pcom}[\![\texttt{skip}]\!]R\\
=\;& \alpha[\![\texttt{skip}]\!](\tau^\star[\![\texttt{skip}]\!])R\\
=\;& \alpha[\![\texttt{skip}]\!](1_{\Sigma[\![P]\!]} \cup \tau[\![\texttt{skip}]\!])R\\
=\;& \{\rho' \mid \exists \rho \in R : \langle\langle \text{at}_P[\![C]\!],\, \rho\rangle,\, \langle \text{after}_P[\![C]\!],\, \rho'\rangle\rangle \in (1_{\Sigma[\![P]\!]} \cup \tau[\![\texttt{skip}]\!])\}\\
=\;& \{\rho' \mid \exists \rho \in R : \rho' = \rho\} \qquad \wr \text{at}_P[\![C]\!] \neq \text{after}_P[\![C]\!] \text{ and def. } \tau[\![\texttt{skip}]\!] \wr\\
=\;& R
\end{aligned}
$$

$$
\begin{aligned}
-\;& \text{Pcom}[\![\texttt{X} := A]\!]R\\
=\;& \alpha[\![\texttt{X} := A]\!](\tau^\star[\![\texttt{X} := A]\!])R\\
=\;& \alpha[\![\texttt{X} := A]\!](1_{\Sigma[\![P]\!]} \cup \tau[\![\texttt{X} := A]\!])R\\
=\;& \{\rho' \mid \exists \rho \in R : \langle\langle \text{at}_P[\![C]\!],\, \rho\rangle,\, \langle \text{after}_P[\![C]\!],\, \rho'\rangle\rangle \in (1_{\Sigma[\![P]\!]} \cup \tau[\![\texttt{X} := A]\!])\}\\
=\;& \{\rho' \mid \exists \rho \in R : \langle\langle \text{at}_P[\![C]\!],\, \rho\rangle,\, \langle \text{after}_P[\![C]\!],\, \rho'\rangle\rangle \in \tau[\![\texttt{X} := A]\!]\} \qquad \wr \text{at}_P[\![C]\!] \neq \text{after}_P[\![C]\!] \wr
\end{aligned}
$$

$= \{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P[\![C]\!],\ \rho\rangle,\ \langle \mathrm{after}_P[\![C]\!],\ \rho'\rangle\rangle \in \{\langle\langle \mathrm{at}_P[\![\mathtt{X} := A]\!],\ \rho\rangle,\ \langle \mathrm{after}_P[\![\mathtt{X} := A]\!],\ \rho[\mathtt{X} := i]\rangle\rangle \mid i \in \mathbb{I} \wedge \rho \vdash A \Rightarrow i\}\}$

$\wr$def. $\tau[\![\mathtt{X} := A]\!]$$\int$

$= \{\rho[\mathtt{X} := i] \mid \rho \in R \wedge i \in \mathbb{I} \wedge \rho \vdash A \Rightarrow i\}$

$= \{\rho[\mathtt{X} := i] \mid \rho \in R \wedge i \in \{v \mid \rho \vdash A \Rightarrow v\} \cap \mathbb{I}\}$

$= \{\rho[\mathtt{X} := i] \mid \rho \in R \wedge i \in \{v \mid \exists \rho' \in \{\rho\} : \rho' \vdash A \Rightarrow v\} \cap \mathbb{I}\}$

$= \{\rho[\mathtt{X} := i] \mid \rho \in R \wedge i \in (\mathrm{Faexp}[\![A]\!](\{\rho\})) \cap \mathbb{I}\}$

— $\mathrm{Pcom}[\![C]\!]R$ where $C = \mathtt{if}\ B\ \mathtt{then}\ S_t\ \mathtt{else}\ S_f\ \mathtt{fi}$

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

$= \alpha[\![C]\!]((1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t)\ \cup$
$(1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^f))R$

$=$ $\wr\alpha[\![C]\!]$ is a complete join morphism$\int$

$= \alpha[\![C]\!]((1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t))R\ \cup$
$\alpha[\![C]\!]((1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^f))R$

We handle the case of the true alternative only, since the false alternativce can be handled in the same way.

$\alpha[\![C]\!]((1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t))R$

$= \{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P[\![C]\!],\ \rho\rangle,\ \langle \mathrm{after}_P[\![C]\!],\ \rho'\rangle\rangle \in ((1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t))\}$ $\wr$by def. $\alpha[\![C]\!]$$\int$

$= \{\rho''' \mid \exists \rho \in R : \exists \rho', \rho'' \in \mathrm{Env}[\![P]\!] : \exists \ell', \ell'' \in \mathrm{in}_P[\![C]\!] : \langle\langle \mathrm{at}_P[\![C]\!],\ \rho\rangle,\ \langle \ell',\ \rho'\rangle\rangle \in (1_{\Sigma[\![P]\!]} \cup \tau^B) \wedge \langle\langle \ell',\ \rho'\rangle,\ \langle \ell'',\ \rho''\rangle\rangle \in \tau^\star[\![S_t]\!] \wedge \langle\langle \ell'',\ \rho''\rangle,\ \langle \mathrm{after}_P[\![C]\!],\ \rho'''\rangle\rangle \in (1_{\Sigma[\![P]\!]} \cup \tau^t)\}$ $\wr$def. composition $\circ$$\int$

$=$ $\wr$We observe that $\mathrm{at}_P[\![C]\!] = \ell'$ and $\mathrm{after}_P[\![C]\!] = \ell''$ is impossible since $\langle\langle \ell',\ \rho'\rangle,\ \langle \ell'',\ \rho''\rangle\rangle \in \tau^\star[\![S_t]\!]$ so $\ell', \ell'' \in \mathrm{in}_P[\![S_t]\!]$ in contradiction with $\{\mathrm{at}_P[\![C]\!], \mathrm{after}_P[\![C]\!]\} \cap (\mathrm{in}_P[\![S_t]\!] \cup \mathrm{in}_P[\![S_f]\!]) = \emptyset$$\int$

$\{\rho''' \mid \exists \rho \in R : \exists \rho', \rho'' \in \mathrm{Env}[\![P]\!] : \exists \ell', \ell'' \in \mathrm{in}_P[\![C]\!] : \langle\langle \mathrm{at}_P[\![C]\!],\ \rho\rangle,\ \langle \ell',\ \rho'\rangle\rangle \in \tau^B \wedge \langle\langle \ell',\ \rho'\rangle,\ \langle \ell'',\ \rho''\rangle\rangle \in \tau^\star[\![S_t]\!] \wedge \langle\langle \ell'',\ \rho''\rangle,\ \langle \mathrm{after}_P[\![C]\!],\ \rho'''\rangle\rangle \in \tau^t\}$

$=$ $\wr$By def. of τ^B, we have $\mathrm{at}_P[\![S_t]\!] = \ell'$ and by def. of τ^t, we have $\mathrm{after}_P[\![S_t]\!] = \ell''$$\int$

$\{\rho''' \mid \exists \rho \in R : \exists \rho', \rho'' \in \mathrm{Env}[\![P]\!] : \langle\langle \mathrm{at}_P[\![C]\!],\ \rho\rangle,\ \langle \mathrm{at}_P[\![S_t]\!],\ \rho'\rangle\rangle \in \tau^B \wedge \langle\langle \mathrm{at}_P[\![S_t]\!],\ \rho'\rangle,\ \langle \mathrm{after}_P[\![S_t]\!],\ \rho''\rangle\rangle \in \tau^\star[\![S_t]\!] \wedge \langle\langle \mathrm{after}_P[\![S_t]\!],\ \rho''\rangle,\ \langle \mathrm{after}_P[\![C]\!],\ \rho'''\rangle\rangle \in \tau^t\}$

$=$ $\wr$By def. τ^t so that $\rho'' = \rho'''$$\int$

$\{\rho''' \mid \exists \rho \in R : \exists \rho' \in \mathrm{Env}[\![P]\!] : \langle\langle \mathrm{at}_P[\![C]\!],\ \rho\rangle,\ \langle \mathrm{at}_P[\![S_t]\!],\ \rho'\rangle\rangle \in \tau^B \wedge \langle\langle \mathrm{at}_P[\![S_t]\!],\ \rho'\rangle,\ \langle \mathrm{after}_P[\![S_t]\!],\ \rho'''\rangle\rangle \in \tau^\star[\![S_t]\!]\}$

$=$ $\wr$By def. τ^B so that $\rho' = \rho$ and $\rho' \vdash B \Rightarrow \mathtt{tt}$$\int$

$\{\rho''' \mid \exists \rho \in R : \rho \vdash B \Rightarrow \mathtt{tt} \wedge \langle\langle \mathrm{at}_P[\![S_t]\!],\ \rho\rangle,\ \langle \mathrm{after}_P[\![S_t]\!],\ \rho'''\rangle\rangle \in \tau^\star[\![S_t]\!]\}$

$= \{\rho' \mid \exists \rho \in \{\rho'' \in R \mid \rho'' \vdash B \Rightarrow \mathtt{tt}\} : \langle\langle \mathrm{at}_P[\![S_t]\!],\ \rho\rangle,\ \langle \mathrm{after}_P[\![S_t]\!],\ \rho'\rangle\rangle \in \tau^\star[\![S_t]\!]\}$

$\wr$By def. (2) of $\mathrm{Cbexp}[\![B]\!]R$$\int$

$= \{\rho' \mid \exists \rho \in \mathrm{Cbexp}[\![B]\!]R : \langle\langle \mathrm{at}_P[\![S_t]\!],\ \rho\rangle,\ \langle \mathrm{after}_P[\![S_t]\!],\ \rho'\rangle\rangle \in \tau^\star[\![S_t]\!]\}$

$= \mathrm{Pcom}[\![S_t]\!](\mathrm{Cbexp}[\![B]\!]R)$

The false alternative is similar with S_f for S_t and $\neg(B)$ for B.

— $\mathrm{Pcom}[\![C]\!]R$ where $C = \mathtt{while}\ B\ \mathtt{do}\ S\ \mathtt{od}$

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

$= \alpha[\![C]\!](((1_{\Sigma[\![P]\!]} \cup \tau^\star[\![S]\!] \circ \tau^R) \circ (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \circ (1_{\Sigma[\![P]\!]} \cup \tau^B \circ \tau^\star[\![S]\!] \cup \tau^{\bar{B}})) \cup \tau[\![S]\!]^\star)R$

$=$ $\wr\alpha[\![C]\!]$ is a complete join morphism$\int$

$\alpha[\![C]\!]((1_{\Sigma[\![P]\!]} \cup \tau^\star[\![S]\!] \circ \tau^R) \circ (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \circ (1_{\Sigma[\![P]\!]} \cup \tau^B \circ \tau^\star[\![S]\!] \cup \tau^{\bar{B}}))R \cup \alpha[\![C]\!](\tau[\![S]\!]^\star)R$

$=$ $\wr$def. $\alpha[\![C]\!]$$\int$

$\alpha[\![C]\!]((1_{\Sigma[\![P]\!]} \cup \tau^\star[\![S]\!] \circ \tau^R) \circ (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \circ (1_{\Sigma[\![P]\!]} \cup \tau^B \circ \tau^\star[\![S]\!] \cup \tau^{\bar{B}}))R \cup \{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P[\![C]\!],\ \rho\rangle,\ \langle \mathrm{after}_P[\![C]\!],\ \rho'\rangle\rangle \in \tau[\![S]\!]^\star\}$

$=$ $\wr$For the second term, we have $\langle\langle \mathrm{at}_P[\![C]\!],\ \rho\rangle,\ \langle \mathrm{after}_P[\![C]\!],\ \rho'\rangle\rangle \in \tau[\![S]\!]^\star$ which implies $\{\mathrm{at}_P[\![C]\!], \mathrm{after}_P[\![C]\!]\} \subseteq \mathrm{in}_P[\![S]\!]$ and so $\{\mathrm{at}_P[\![C]\!], \mathrm{after}_P[\![C]\!]\} \cap \mathrm{in}_P[\![S]\!] = \emptyset$ implies that this term is $\emptyset$$\int$

$\alpha[\![C]\!]((1_{\Sigma[\![P]\!]} \cup \tau^\star[\![S]\!] \circ \tau^R) \circ (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \circ (1_{\Sigma[\![P]\!]} \cup \tau^B \circ \tau^\star[\![S]\!] \cup \tau^{\bar{B}}))R$

$=$ $\wr$def. $\alpha[\![C]\!]$$\int$

$\{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P[\![C]\!], \rho\rangle, \langle \mathrm{after}_P[\![C]\!], \rho'\rangle\rangle \in ((1_{\Sigma[\![P]\!]} \cup \tau^\star[\![S]\!] \circ \tau^R) \circ (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \circ (1_{\Sigma[\![P]\!]} \cup \tau^B \circ \tau^\star[\![S]\!] \cup \tau^{\bar{B}}))\}$

$=$ $\wr$ $\mathrm{at}_P[\![C]\!] \notin \mathrm{in}_P[\![S]\!]$ so $\langle\langle \mathrm{at}_P[\![C]\!], \rho\rangle, \langle \ell'', \rho''\rangle\rangle \notin \tau^\star[\![S]\!] \circ \tau^R$ and $\mathrm{after}_P[\![C]\!] \notin \mathrm{in}_P[\![S]\!]$ so $\langle\langle \ell, \rho\rangle, \langle \mathrm{after}_P[\![C]\!], \rho'\rangle\rangle \notin \tau^B \circ \tau^\star[\![S]\!]$ $\wr$

$\{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P[\![C]\!], \rho\rangle, \langle \mathrm{after}_P[\![C]\!], \rho'\rangle\rangle \in ((1_{\Sigma[\![P]\!]}) \circ (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \circ (1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}))\}$

$=$ $\wr$ $1_{\Sigma[\![P]\!]}$ neutral element of $\circ$ $\wr$

$\{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P[\![C]\!], \rho\rangle, \langle \mathrm{after}_P[\![C]\!], \rho'\rangle\rangle \in ((\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \circ (1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}))\}$

$=$ $\wr$ $\langle\langle \mathrm{at}_P[\![C]\!], \rho\rangle, \langle \mathrm{after}_P[\![C]\!], \rho'\rangle\rangle \notin 1_{\Sigma[\![P]\!]}$ since $\mathrm{at}_P[\![C]\!] \neq \mathrm{after}_P[\![C]\!]$ so the term $((\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \circ (1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}))$ cannot reduce to $1_{\Sigma[\![P]\!]}$. Moreover $\langle\langle \ell, \rho\rangle, \langle \mathrm{after}_P[\![C]\!], \rho'\rangle\rangle \not{f} \circ \tau^R$ so the term $((\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \circ (1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}))$ cannot either reduce to $(\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^+$ $\wr$

$\{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P[\![C]\!], \rho\rangle, \langle \mathrm{after}_P[\![C]\!], \rho'\rangle\rangle \in ((\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \circ \tau^{\bar{B}})\}$

$=$ $\wr$ def. composition $\circ$ $\wr$

$\{\rho' \mid \exists \rho \in R : \exists \ell'' \in \mathrm{in}_P[\![C]\!] : \exists \rho'' \in \mathrm{Env}[\![P]\!] : \langle\langle \mathrm{at}_P[\![C]\!], \rho\rangle, \langle \ell'', \rho''\rangle\rangle \in (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \wedge \langle\langle \ell'', \rho''\rangle, \langle \mathrm{after}_P[\![C]\!], \rho'\rangle\rangle \in \tau^{\bar{B}}\}$

$=$ $\wr$ def. $\tau^{\bar{B}}$ so that $\ell'' = \mathrm{at}_P[\![C]\!]$ and $\rho'' = \rho'$ $\wr$

$\{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P[\![C]\!], \rho\rangle, \langle \mathrm{at}_P[\![C]\!], \rho'\rangle\rangle \in (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \wedge \langle\langle \mathrm{at}_P[\![C]\!], \rho'\rangle, \langle \mathrm{after}_P[\![C]\!], \rho'\rangle\rangle \in \tau^{\bar{B}}\}$

$=$ $\wr$ def. $\tau^{\bar{B}}$ $\wr$

$\{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P[\![C]\!], \rho\rangle, \langle \mathrm{at}_P[\![C]\!], \rho'\rangle\rangle \in (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \wedge \rho' \vdash T(\neg B) \Rightarrow \mathrm{tt}\}$

$=$ let $I \stackrel{\mathrm{def}}{=} \{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P[\![C]\!], \rho\rangle, \langle \mathrm{at}_P[\![C]\!], \rho'\rangle\rangle \in (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star\}$ in
$\{\rho' \in I \mid \rho' \vdash T(\neg B) \Rightarrow \mathrm{tt}\}$

$=$ let $I \stackrel{\mathrm{def}}{=} \{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P[\![C]\!], \rho\rangle, \langle \mathrm{at}_P[\![C]\!], \rho'\rangle\rangle \in (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star\}$ in
$\mathrm{Cbexp}[\![T(\neg B)]\!]I$

$=$ let $I \stackrel{\mathrm{def}}{=} \alpha'((\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star)$ in
$\mathrm{Cbexp}[\![T(\neg B)]\!]I$

by defining (for a given program P, command C and set of environments R):

$$\alpha'(t) \stackrel{\mathrm{def}}{=} \{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P[\![C]\!], \rho\rangle, \langle \mathrm{at}_P[\![C]\!], \rho'\rangle\rangle \in t\}$$

We have

$$\alpha'(\bigcup_{i \in \Delta} t_i) = \bigcup_{i \in \Delta} \alpha'(t_i)$$

whence a Galois connection

$$\langle \wp(\Sigma[\![P]\!] \times \Sigma[\![P]\!]), \subseteq\rangle \underset{\alpha'}{\overset{\gamma'}{\leftrightarrows}} \langle \wp(\mathbb{R}), \subseteq\rangle$$

such that

$$I = \alpha'(t^\star) \quad \text{where} \quad t = (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)$$
$$= \alpha'(\mathbf{lfp}\, \lambda X \cdot 1_{\Sigma[\![P]\!]} \cup X \circ t)$$
$$= \mathbf{lfp}\, F'$$

where $\alpha'(1_{\Sigma[\![P]\!]} \cup X \circ t) = F'(\alpha'(X))$ that is $\alpha'(1_{\Sigma[\![P]\!]}) \cup \alpha'(X \circ t) = F'(\alpha'(X))$ since α' is a complete join morphism.

— $\alpha'(X \circ t)$

$= \{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P[\![C]\!], \rho\rangle, \langle \mathrm{at}_P[\![C]\!], \rho'\rangle\rangle \in X \circ t\}$ $\wr$ def. α' $\wr$

$= \{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P[\![C]\!], \rho\rangle, \langle \mathrm{at}_P[\![C]\!], \rho'\rangle\rangle \in X \circ \tau^B \circ \tau^\star[\![S]\!] \circ \tau^R\}$ $\wr$ def. t $\wr$

$= \{\rho' \mid \exists \rho \in R : \exists \ell_1, \ell_2, \ell_3 \in \mathrm{in}_P[\![C]\!] : \exists \rho_1, \rho_2, \rho_3 \in \mathrm{Env}[\![P]\!] : \langle\langle \mathrm{at}_P[\![C]\!], \rho\rangle, \langle \ell_1, \rho_1\rangle\rangle \in X \wedge \langle\langle \ell_1, \rho_1\rangle, \langle \ell_2, \rho_2\rangle\rangle \in \tau^B \wedge \langle\langle \ell_2, \rho_2\rangle, \langle \ell_3, \rho_3\rangle\rangle \in \tau^\star[\![S]\!] \wedge \langle\langle \ell_3, \rho_3\rangle, \langle \mathrm{at}_P[\![C]\!], \rho'\rangle\rangle \in \tau^R\}$ $\wr$ def. $\circ$ $\wr$

$=$ $\wr$ by def. τ^B, we have $\ell_1 = \mathrm{at}_P[\![C]\!]$, $\ell_2 = \mathrm{at}_P[\![S]\!]$, $\rho_2 = \rho_1$ and $\rho_1 \vdash B \Rightarrow \mathrm{tt}$ $\wr$

$\{\rho' \mid \exists \rho \in R : \exists \ell_3 \in \mathrm{in}_P[\![C]\!] : \exists \rho_1, \rho_3 \in \mathrm{Env}[\![P]\!] : \langle\langle \mathrm{at}_P[\![C]\!], \rho\rangle, \langle \mathrm{at}_P[\![C]\!], \rho_1\rangle\rangle \in X \wedge \rho_1 \vdash B \Rightarrow \mathrm{tt} \wedge \langle\langle \mathrm{at}_P[\![S]\!], \rho_1\rangle, \langle \ell_3, \rho_3\rangle\rangle \in \tau^\star[\![S]\!] \wedge \langle\langle \ell_3, \rho_3\rangle, \langle \mathrm{at}_P[\![C]\!], \rho'\rangle\rangle \in \tau^R\}$

$=$ $\wr$ by def. τ^R, we have $\ell_3 = \mathrm{after}_P[\![S]\!]$ and $\rho_3 = \rho'$ $\wr$

$\{\rho' \mid \exists \rho \in R : \exists \rho_1 \in \mathrm{Env}[\![P]\!] : \langle\langle \mathrm{at}_P[\![C]\!], \rho\rangle, \langle \mathrm{at}_P[\![C]\!], \rho_1\rangle\rangle \in X \wedge \rho_1 \vdash B \Rightarrow \mathrm{tt} \wedge \langle\langle \mathrm{at}_P[\![S]\!], \rho_1\rangle, \langle \mathrm{after}_P[\![S]\!], \rho'\rangle\rangle \in \tau^\star[\![S]\!]\}$

$= \{\rho' \mid \exists \rho_1 \in \{\rho_1 \in \mathrm{Env}[\![P]\!] \mid \exists \rho \in R : \langle\langle \mathrm{at}_P[\![C]\!], \rho\rangle, \langle \mathrm{at}_P[\![C]\!], \rho_1\rangle\rangle \in X\} : \rho_1 \vdash B \Rightarrow \mathrm{tt} \wedge \langle\langle \mathrm{at}_P[\![S]\!], \rho_1\rangle, \langle \mathrm{after}_P[\![S]\!], \rho'\rangle\rangle \in \tau^\star[\![S]\!]\}$

$= \{\rho' \mid \exists \rho_1 \in \alpha'(X) : \rho_1 \vdash B \Rrightarrow \mathrm{tt} \wedge \langle\langle \mathrm{at}_P\llbracket S\rrbracket,\ \rho_1\rangle,\ \langle \mathrm{after}_P\llbracket S\rrbracket,\ \rho'\rangle\rangle \in \tau^\star\llbracket S\rrbracket\}$
$\wr$by def. $\alpha'$$\wr$
$= \{\rho' \mid \exists \rho_1 \in \{\rho_1 \in \alpha'(X) \mid \rho_1 \vdash B \Rrightarrow \mathrm{tt}\} \wedge \langle\langle \mathrm{at}_P\llbracket S\rrbracket,\ \rho_1\rangle,\ \langle \mathrm{after}_P\llbracket S\rrbracket,\ \rho'\rangle\rangle \in \tau^\star\llbracket S\rrbracket\}$
$= \{\rho' \mid \exists \rho_1 \in \mathrm{Cbexp}\llbracket B\rrbracket(\alpha'(X)) \wedge \langle\langle \mathrm{at}_P\llbracket S\rrbracket,\ \rho_1\rangle,\ \langle \mathrm{after}_P\llbracket S\rrbracket,\ \rho'\rangle\rangle \in \tau^\star\llbracket S\rrbracket\}$
$= \mathrm{Pcom}\llbracket S\rrbracket(\mathrm{Cbexp}\llbracket B\rrbracket(\alpha'(X)))$

— $\alpha'(1_{\Sigma\llbracket P\rrbracket})$
$= \{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho\rangle,\ \langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle\rangle \in 1_{\Sigma\llbracket P\rrbracket}\}$ $\wr$def. $\alpha'$$\wr$
$= \{\rho' \mid \exists \rho \in R : \rho = \rho'\}$
$= R$

So $F'(X) = R \cup \mathrm{Pcom}\llbracket S\rrbracket(\mathrm{Cbexp}\llbracket B\rrbracket(X))$ and $I = \mathsf{lfp}^{\subseteq}_{\emptyset}\ F'$.

— $\mathrm{Pcom}\llbracket C\ ;\ S\rrbracket R$
$= \alpha\llbracket C\ ;\ S\rrbracket(\tau^\star\llbracket C\ ;\ S\rrbracket)R$
$= \alpha\llbracket C\ ;\ S\rrbracket(\tau^\star\llbracket C\rrbracket \circ \tau^\star\llbracket S\rrbracket)R$

$= \{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P\llbracket C\ ;\ S\rrbracket,\ \rho\rangle,\ \langle \mathrm{after}_P\llbracket C\ ;\ S\rrbracket,\ \rho'\rangle\rangle \in (\tau^\star\llbracket C\rrbracket \circ \tau^\star\llbracket S\rrbracket)\}$
$=$ $\wr \mathrm{at}_P\llbracket C\ ;\ S\rrbracket = \mathrm{at}_P\llbracket C\rrbracket$ and $\mathrm{after}_P\llbracket C\ ;\ S\rrbracket = \mathrm{after}_P\llbracket S\rrbracket\wr$
$= \{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho\rangle,\ \langle \mathrm{after}_P\llbracket S\rrbracket,\ \rho'\rangle\rangle \in (\tau^\star\llbracket C\rrbracket \circ \tau^\star\llbracket S\rrbracket)\}$
$= \{\rho' \mid \exists \rho \in R : \exists \ell'' \in \mathrm{in}_P\llbracket C\ ;\ S\rrbracket : \exists \rho'' \in \mathbb{R} : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho\rangle,\ \langle \ell'',\ \rho''\rangle\rangle \in \tau^\star\llbracket C\rrbracket \wedge \langle\langle \ell'',\ \rho''\rangle,\ \langle \mathrm{after}_P\llbracket S\rrbracket,\ \rho'\rangle\rangle \in \tau^\star\llbracket S\rrbracket)\}$
$=$ $\wr\langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho\rangle,\ \langle \ell'',\ \rho''\rangle\rangle \in \tau^\star\llbracket C\rrbracket$ so $\ell'' \in \mathrm{in}_P\llbracket C\rrbracket$ and $\langle\langle \ell'',\ \rho''\rangle,\ \langle \mathrm{after}_P\llbracket S\rrbracket,\ \rho'\rangle\rangle \in \tau^\star\llbracket S\rrbracket$ so $\ell'' \in \mathrm{in}_P\llbracket S\rrbracket$. Hence $\ell'' \in \mathrm{in}_P\llbracket C\rrbracket \cap \mathrm{in}_P\llbracket S\rrbracket$ proving $\ell'' = \mathrm{after}_P\llbracket C\rrbracket = \mathrm{at}_P\llbracket S\rrbracket\wr$
$\{\rho' \mid \exists \rho \in R : \exists \rho'' \in \mathbb{R} : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho''\rangle\rangle \in \tau^\star\llbracket C\rrbracket \wedge \langle\langle \mathrm{at}_P\llbracket S\rrbracket,\ \rho''\rangle,\ \langle \mathrm{after}_P\llbracket S\rrbracket,\ \rho'\rangle\rangle \in \tau^\star\llbracket S\rrbracket)\}$
$= \{\rho' \mid \exists \rho'' \in \mathbb{R} : \exists \rho \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho''\rangle\rangle \in \tau^\star\llbracket C\rrbracket \wedge \langle\langle \mathrm{at}_P\llbracket S\rrbracket,\ \rho''\rangle,\ \langle \mathrm{after}_P\llbracket S\rrbracket,\ \rho'\rangle\rangle \in \tau^\star\llbracket S\rrbracket)\}$
$= \{\rho' \mid \exists \rho'' \in \{\rho'' \mid \mathbb{R} : \exists \rho \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho''\rangle\rangle \in \tau^\star\llbracket C\rrbracket\} : \langle\langle \mathrm{at}_P\llbracket S\rrbracket,\ \rho''\rangle,\ \langle \mathrm{after}_P\llbracket S\rrbracket,\ \rho'\rangle\rangle \in \tau^\star\llbracket S\rrbracket)\}$
$= \mathrm{Pcom}\llbracket S\rrbracket(\{\rho'' \mid \mathbb{R} : \exists \rho \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho''\rangle\rangle \in \tau^\star\llbracket C\rrbracket\})$
$= \mathrm{Pcom}\llbracket S\rrbracket(\mathrm{Pcom}\llbracket C\rrbracket(R))$

$= (\mathrm{Pcom}\llbracket S\rrbracket \circ \mathrm{Pcom}\llbracket C\rrbracket)(R)$

— $\mathrm{Pcom}\llbracket S\ ;;\rrbracket R$
$= \alpha\llbracket S\ ;;\rrbracket(\tau^\star\llbracket S\ ;;\rrbracket)R$
$= \{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho'\rangle\rangle \in \tau^\star\llbracket S\ ;;\rrbracket\}$
$= \{\rho' \mid \exists \rho \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho'\rangle\rangle \in \tau^\star\llbracket S\rrbracket\}$
$= \mathrm{Pcom}\llbracket S\rrbracket R$

$\square$

Reminder of the small-step operational semantics of commands

Small-step operational semantics of commands and programs

Identity $C = \texttt{skip}$ $(\mathsf{at}_P[\![C]\!] = \ell \neq \ell' = \mathsf{after}_P[\![C]\!])$

$$\langle \ell,\ \rho\rangle \longmapsto\!\![\![\texttt{skip}]\!]\!\!\Longrightarrow \langle \ell',\ \rho\rangle \qquad (19)$$

Assignment $C = \texttt{X := } A$ $(\mathsf{at}_P[\![C]\!] = \ell \neq \ell' = \mathsf{after}_P[\![C]\!])$

$$\frac{\rho \vdash A \Rrightarrow i}{\langle \ell,\ \rho\rangle \longmapsto\!\![\![\texttt{X := } A]\!]\!\!\Longrightarrow \langle \ell',\ \rho[\texttt{X} := i]\rangle},\ i \in \mathbb{I} \qquad (20)$$

Conditional $C = \texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}$ $(\mathsf{at}_P[\![C]\!] = \ell \neq \ell' = \mathsf{after}_P[\![C]\!])$

$$\frac{\rho \vdash B \Rrightarrow \mathtt{tt}}{\langle \ell,\ \rho\rangle \longmapsto\!\![\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!]\!\!\Longrightarrow \langle \mathsf{at}_P[\![S_t]\!],\ \rho\rangle} \qquad (21)$$

$$\mathsf{at}_P[\![S_t]\!] \neq \mathsf{at}_P[\![C]\!]$$

$$\frac{\rho \vdash T(\neg B) \Rrightarrow \mathtt{tt}}{\langle \ell,\ \rho\rangle \longmapsto\!\![\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!]\!\!\Longrightarrow \langle \mathsf{at}_P[\![S_f]\!],\ \rho\rangle} \qquad (22)$$

$$\mathsf{at}_P[\![S_f]\!] \neq \mathsf{at}_P[\![C]\!]$$

$$\frac{\langle \ell_1,\ \rho_1\rangle \longmapsto\!\![\![S_t]\!]\!\!\Longrightarrow \langle \ell_2,\ \rho_2\rangle}{\langle \ell_1,\ \rho_1\rangle \longmapsto\!\![\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!]\!\!\Longrightarrow \langle \ell_2,\ \rho_2\rangle} \qquad (23)$$

$$\ell_2 \neq \mathsf{at}_P[\![C]\!]$$

$$\frac{\langle \ell_1,\ \rho_1\rangle \longmapsto\!\![\![S_f]\!]\!\!\Longrightarrow \langle \ell_2,\ \rho_2\rangle}{\langle \ell_1,\ \rho_1\rangle \longmapsto\!\![\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!]\!\!\Longrightarrow \langle \ell_2,\ \rho_2\rangle} \qquad (24)$$

$$\ell_2 \neq \mathsf{at}_P[\![C]\!]$$

$$\langle \mathsf{after}_P[\![S_t]\!],\ \rho\rangle \longmapsto\!\![\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!]\!\!\Longrightarrow \langle \ell',\ \rho\rangle \qquad (25)$$

$$\ell' = \mathsf{after}_P[\![C]\!] \neq \mathsf{at}_P[\![C]\!]$$

$$\langle \mathsf{after}_P[\![S_f]\!],\ \rho\rangle \longmapsto\!\![\![\texttt{if } B \texttt{ then } S_t \texttt{ else } S_f \texttt{ fi}]\!]\!\!\Longrightarrow \langle \ell',\ \rho\rangle \qquad (26)$$

$$\ell' = \mathsf{after}_P[\![C]\!] \neq \mathsf{at}_P[\![C]\!]$$

Iteration $C = \texttt{while } B \texttt{ do } S \texttt{ od}$ $(\mathsf{at}_P[\![C]\!] = \ell,$ $\mathsf{after}_P[\![C]\!] = \ell'$ and $\ell_1, \ell_2 \in \mathsf{in}_P[\![S]\!])$

$$\frac{\rho \vdash T(\neg B) \Rrightarrow \mathtt{tt}}{\langle \ell,\ \rho\rangle \longmapsto\!\![\![\texttt{while } B \texttt{ do } S \texttt{ od}]\!]\!\!\Longrightarrow \langle \ell',\ \rho\rangle} \qquad (27)$$

$$\frac{\rho \vdash B \Rrightarrow \mathtt{tt}}{\langle \ell,\ \rho\rangle \longmapsto\!\![\![\texttt{while } B \texttt{ do } S \texttt{ od}]\!]\!\!\Longrightarrow \langle \mathsf{at}_P[\![S]\!],\ \rho\rangle} \qquad (28)$$

$$\frac{\langle \ell_1,\ \rho_1\rangle \longmapsto\!\![\![S]\!]\!\!\Longrightarrow \langle \ell_2,\ \rho_2\rangle}{\langle \ell_1,\ \rho_1\rangle \longmapsto\!\![\![\texttt{while } B \texttt{ do } S \texttt{ od}]\!]\!\!\Longrightarrow \langle \ell_2,\ \rho_2\rangle} \qquad (29)$$

$$\langle \mathsf{after}_P[\![S]\!],\ \rho\rangle \longmapsto\!\![\![\texttt{while } B \texttt{ do } S \texttt{ od}]\!]\!\!\Longrightarrow \langle \ell,\ \rho\rangle \qquad (30)$$

Sequence C_1 ; ... ; C_n, $n > 0$ ($i \in [1,n]$:$\ell_i, \ell_{i+1} \in \mathrm{in}_P[\![C_i]\!]$)

$$\frac{\langle \ell_i,\ \rho_i\rangle \Longmapsto\!\![\![C_i]\!]\!\!\Longrightarrow \langle \ell_{i+1},\ \rho_{i+1}\rangle}{\langle \ell_i,\ \rho_i\rangle \Longmapsto\!\![\![C_1\ ;\ \ldots\ ;\ C_n]\!]\!\!\Longrightarrow \langle \ell_{i+1},\ \rho_{i+1}\rangle} \tag{31}$$

$\mathrm{after}_P[\![C_i]\!] = \mathrm{at}_P[\![C_{i+1}]\!]$

Program $P = S$;;

$$\frac{\langle \ell,\ \rho\rangle \Longmapsto\!\![\![S]\!]\!\!\Longrightarrow \rho'}{\langle \ell,\ \rho\rangle \Longmapsto\!\![\![S\ ;;]\!]\!\!\Longrightarrow \langle \ell',\ \rho'\rangle} \tag{32}$$

$\ell' = \mathrm{after}_P[\![P]\!] = \mathrm{after}_P[\![S]\!] \neq \mathrm{at}_P[\![S]\!] = \mathrm{at}_P[\![P]\!]$

Transition system of a program

The transition system of a program $P = S$;; is

$$\langle \Sigma[\![P]\!],\ \tau[\![P]\!]\rangle$$

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

$$\Sigma[\![P]\!] \stackrel{\text{def}}{=} \mathrm{in}_P[\![P]\!] \times \mathrm{Env}[\![P]\!] \tag{33}$$

$$\tau[\![C]\!] \stackrel{\text{def}}{=} \{\langle\langle \ell,\ \rho\rangle,\ \langle \ell',\ \rho'\rangle\rangle \mid \langle \ell,\ \rho\rangle \Longmapsto\!\![\![C]\!]\!\!\Longrightarrow \langle \ell',\ \rho'\rangle\} \tag{34}$$

Forward reachability collecting semantics of commands

Definition of the forward reachability collecting semantics of commands

The forward reachability collecting semantics $\mathrm{Rcom}[\![C]\!]R$ of a command $C \in \mathrm{Com}$ (of a given program P) specifies the set of reachable sattes during any execution of C starting at its starting point in any of the enviornments satisfying the precondition R.

$$\mathrm{Rcom}\ \in\ \mathrm{Com} \mapsto \wp(\mathrm{Env}[\![P]\!]) \stackrel{\sqcup}{\longmapsto} (\mathrm{in}_P[\![C]\!] \mapsto \wp(\mathrm{Env}[\![P]\!]))$$

$$\mathrm{Rcom}[\![C]\!]R\ell \stackrel{\text{def}}{=} \{\rho \mid \exists \rho' \in R : \langle\langle \mathrm{at}_P[\![C]\!],\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in \tau^\star[\![C]\!]\}$$

Property of the forward reachability collecting semantics of commands

The forward reachability collecting semantics of a command is a complete join morphism (denoted by $\stackrel{\sqcup}{\longmapsto}$) that is ($\mathcal{S}$ is an arbitrary set):

$$\mathrm{Rcom}[\![C]\!]\Big(\bigcup_{k \in \mathcal{S}} R_k\Big) = \bigcup_{k \in \mathcal{S}} (\mathrm{Rcom}[\![C]\!]R_k)$$

which implies monotony, continuity and strictness:

$$\mathrm{Rcom}[\![C]\!]\emptyset = \emptyset \ .$$

Postcondition semantics as an abstraction of the forward reachability collecting semantics

We have

$$\mathrm{Pcom}[\![C]\!]R = \mathrm{Rcom}[\![C]\!]R(\mathrm{after}_P[\![C]\!]) \qquad (35)$$

which is an abstraction $\alpha[\![C]\!](\mathrm{Rcom}[\![C]\!]R)$ of a function at a point, by defining $\alpha[\![C]\!](f) = f(\mathrm{after}_P[\![C]\!])$ such that

$$\langle \mathrm{Com} \mapsto (\wp(\mathrm{Env}[\![P]\!]) \stackrel{\sqcup}{\longmapsto} \wp(\mathrm{Env}[\![P]\!])),\ \dot{\subseteq}\rangle \xrightleftharpoons[\alpha[\![C]\!]]{\gamma[\![C]\!]} \langle \wp(\mathrm{Env}[\![P]\!]) \stackrel{\sqcup}{\longmapsto} \wp(\mathrm{Env}[\![P]\!]),\ \subseteq\rangle$$

So we could have first designed $\mathrm{Rcom}[\![C]\!]$ from $\tau^\star[\![C]\!]$ and then $\mathrm{Pcom}[\![C]\!]$ from $\mathrm{Rcom}[\![C]\!]$ [2]

[2] but for pedagogical reasons, we first designed $\mathrm{Pcom}[\![C]\!]$ directly from $\tau^\star[\![C]\!]$ thinking that this would be more simple. Moreoever, $\mathrm{Rcom}[\![C]\!]$ contains fixpoint terms already computed for $\mathrm{Pcom}[\![C]\!]$ which makes the presentation more modular.

Structural definition of the forward reachability collecting semantics of commands

$\mathrm{Rcom}[\![\mathtt{skip}]\!]R\ell = R$

$\mathrm{Rcom}[\![\mathtt{X} := A]\!]R\ell = \mathsf{match}\ \ell\ \mathsf{with}$
$\mid \mathsf{at}_P[\![\mathtt{X} := A]\!] \to R$
$\mid \mathsf{after}_P[\![\mathtt{X} := A]\!] \to \{\rho[\mathtt{X} := i] \mid \rho \in R \wedge i \in (\mathrm{Faexp}[\![A]\!]\{\rho\}) \cap \mathbb{I}\}$

$\mathrm{Rcom}[\![C]\!]R\ell$ where $C = \mathtt{if}\ B\ \mathtt{then}\ S_t\ \mathtt{else}\ S_f\ \mathtt{fi} =$
$\quad \mathsf{match}\ \ell\ \mathsf{with}$
$\qquad \mid \mathsf{at}_P[\![C]\!] \to R$
$\qquad \mid \mathsf{in}_P[\![S_t]\!] \to \mathrm{Rcom}[\![S_t]\!](\mathrm{Cbexp}[\![B]\!]R)\ell$
$\qquad \mid \mathsf{in}_P[\![S_f]\!] \to \mathrm{Rcom}[\![S_f]\!](\mathrm{Cbexp}[\![T(\neg(B))]\!]R)\ell$
$\qquad \mid \mathsf{after}_P[\![C]\!] \to \mathrm{Rcom}[\![S_t]\!](\mathrm{Cbexp}[\![B]\!]R)(\mathsf{after}_P[\![S_t]\!])$
$\qquad\qquad \cup \mathrm{Rcom}[\![S_f]\!](\mathrm{Cbexp}[\![T(\neg(B))]\!]R)(\mathsf{after}_P[\![S_f]\!])$

$\mathrm{Rcom}[\![C]\!]R\ell$ where $C = \mathtt{while}\ B\ \mathtt{do}\ S\ \mathtt{od} =$
$\quad \mathsf{let}\ I = \mathsf{lfp}_{\emptyset}^{\subseteq}\ \lambda X \cdot R \cup \mathrm{Rcom}[\![S]\!](\mathrm{Cbexp}[\![B]\!]X)(\mathsf{after}_P[\![S]\!])\ \mathsf{in}$
$\qquad \mathsf{match}\ \ell\ \mathsf{with}$
$\qquad \mid \mathsf{at}_P[\![C]\!] \to I$
$\qquad \mid \mathsf{in}_P[\![S]\!] \to \mathrm{Rcom}[\![S]\!](\mathrm{Cbexp}[\![B]\!]I)(\ell)$
$\qquad \mid \mathsf{after}_P[\![C]\!] \to \mathrm{Cbexp}[\![T(\neg(B))]\!]R)I$

$\mathrm{Rcom}[\![C\ ;\ S]\!]R\ell = \mathsf{match}\ \ell\ \mathsf{with}$
$\quad \mid \mathsf{in}_P[\![C]\!] \to \mathrm{Rcom}[\![C]\!]R\ell$
$\quad \mid \mathsf{in}_P[\![S]\!] \to \mathrm{Rcom}[\![S]\!](\mathrm{Rcom}[\![C]\!]R(\mathsf{after}_P[\![C]\!]))\ell$

$\mathrm{Rcom}[\![S\ ;;]\!]R\ell = \mathrm{Rcom}[\![S]\!]R\ell$

Proof. By structural induction on the abstract syntax of programs. We first distinguish the special case of $\text{Rcom}[\![C]\!]R\ell$ where $\ell = \text{at}_P[\![C]\!]$ and $C \neq$ `while` B `do` S `od`

– $\text{Rcom}[\![C]\!]R(\text{at}_P[\![C]\!])$ when $C \neq$ `while` B `do` S `od`

$= \{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![C]\!],\ \rho'\rangle,\ \langle \text{at}_P[\![C]\!],\ \rho\rangle\rangle \in \tau^\star[\![C]\!]\}$

$= \{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![C]\!],\ \rho'\rangle,\ \langle \text{at}_P[\![C]\!],\ \rho\rangle\rangle \in \bigcup_{n\in\mathbb{N}} \tau[\![C]\!]^n\}$

$=$ ⟅From the definition of $\langle \ell,\ \rho\rangle \stackrel{}{\longmapsto\!\![\![C]\!]\!\!\Longrightarrow} \langle \ell',\ \rho'\rangle$ and that of $\langle\langle \ell,\ \rho\rangle,\ \langle \ell',\ \rho'\rangle\rangle \in \tau[\![C]\!]$, if follows that $\ell' \neq \text{at}_P[\![C]\!]$ whence whenever $n > 0$, $\langle\langle \ell,\ \rho\rangle,\ \langle \ell',\ \rho'\rangle\rangle \in \tau[\![C]\!]^n$ implies $\ell' \neq \text{at}_P[\![C]\!]$⟆

$= \{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![C]\!],\ \rho'\rangle,\ \langle \text{at}_P[\![C]\!],\ \rho\rangle\rangle \in \tau[\![C]\!]^0\}$

$= \{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![C]\!],\ \rho'\rangle,\ \langle \text{at}_P[\![C]\!],\ \rho\rangle\rangle \in 1_{\Sigma[\![P]\!]}\}$

$= \{\rho \mid \exists \rho' \in R : \rho' = \rho\}$

$= R$

We now define $\text{Rcom}[\![C]\!]R\ell$ assuming, but for the case of while loops, that $\ell \in \text{in}_P[\![C]\!] \setminus \{\text{at}_P[\![C]\!]\}$. We proceed by structural induction on C.

– $\text{Rcom}[\![\texttt{skip}]\!]R\ell$

⟅The only case is $\ell = \text{after}_P[\![\texttt{skip}]\!]$⟆

$\text{Rcom}[\![\texttt{skip}]\!]R(\text{after}_P[\![\texttt{skip}]\!]$

$= \{\rho' \mid \exists \rho \in R : \langle\langle \text{at}_P[\![\texttt{skip}]\!],\ \rho\rangle,\ \langle \text{after}_P[\![\texttt{skip}]\!],\ \rho'\rangle\rangle \in \tau^\star[\![\texttt{skip}]\!]\}$

$= \{\rho' \mid \exists \rho \in R : \langle \text{at}_P[\![\texttt{skip}]\!],\ \rho\rangle \longmapsto\!\![\![\texttt{skip}]\!]\!\!\Longrightarrow \langle \text{after}_P[\![\texttt{skip}]\!],\ \rho'\rangle\}$

$= \{\rho' \mid \exists \rho \in R : \rho = \rho'\}$

$= R$

– $\text{Rcom}[\![\texttt{X} := A]\!]R\ell$

⟅The only case is $\ell = \text{after}_P[\![\texttt{X} := A]\!]$⟆

$\text{Rcom}[\![\texttt{X} := A]\!]R(\text{after}_P[\![\texttt{X} := A]\!]$

$= \{\rho' \mid \exists \rho \in R : \langle\langle \text{at}_P[\![\texttt{X} := A]\!],\ \rho\rangle,\ \langle \text{after}_P[\![\texttt{X} := A]\!],\ \rho'\rangle\rangle \in \tau^\star[\![\texttt{X} := A]\!]\}$

$= \{\rho' \mid \exists \rho \in R : \langle\langle \text{at}_P[\![\texttt{X} := A]\!],\ \rho\rangle,\ \langle \text{after}_P[\![\texttt{X} := A]\!],\ \rho'\rangle\rangle \in \tau[\![\texttt{X} := A]\!]\}$

$= \{\rho' \mid \exists \rho \in R : \langle \text{at}_P[\![\texttt{X} := A]\!],\ \rho\rangle \longmapsto\!\![\![\texttt{X} := A]\!]\!\!\Longrightarrow \langle \text{after}_P[\![\texttt{X} := A]\!],\ \rho'\rangle\}$

$= \{\rho[\texttt{X} := i] \mid \rho \in R \wedge i \in \mathbb{I} \wedge \rho \vdash A \Mapsto i\}$

$= \{\rho[\texttt{X} := i] \mid i \in \{v \mid \exists \rho' \in \{\rho\} : \rho' \vdash A \Mapsto i\} \cap \mathbb{I}\}$

$= \{\rho[\texttt{X} := i] \mid \rho \in R \wedge i \in (\text{Faexp}[\![A]\!]\{\rho\}) \cap \mathbb{I}\}$

– $\text{Rcom}[\![C]\!]R\ell$ where $C =$ `if` B `then` S_t `else` S_f `fi`

⟅The only cases are $\ell = \text{after}_P[\![C]\!]$, $\ell \in \text{in}_P[\![S_t]\!]$ and $\ell \in \text{in}_P[\![S_f]\!]$. The two last cases are similar and we handle only one.⟆

– $\text{Rcom}[\![C]\!]R\ell$ where $\ell \in \text{in}_P[\![S_t]\!]$

$= \{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![C]\!],\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in \tau^\star[\![C]\!] \wedge \ell \in \text{in}_P[\![S_t]\!]\}$

$= \{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![C]\!],\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in (((1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t)) \cup ((1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^f))) \wedge \ell \in \text{in}_P[\![S_t]\!]\}$

$=$ ⟅If $\langle\langle \text{at}_P[\![C]\!],\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in ((1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ \tau^\star[\![S_f]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^f))$ then

- either $\langle\langle \text{at}_P[\![C]\!],\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in ((1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ \tau^\star[\![S_f]\!] \circ \tau^f)$ in which case $\ell = \text{after}_P[\![C]\!]$;
- or $\langle\langle \text{at}_P[\![C]\!],\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in ((1_{\Sigma[\![P]\!]} \cup \tau^{\bar{B}}) \circ \tau[\![S_f]\!]^+)$ in which case $\ell \in \text{in}_P[\![S_f]\!]$;
- or $\langle\langle \text{at}_P[\![C]\!],\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in \tau^{\bar{B}}$ in which case $\ell = \text{at}_P[\![S_f]\!]$;
- or $\langle\langle \text{at}_P[\![C]\!],\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in 1_{\Sigma[\![P]\!]}$ in which case $\ell = \text{at}_P[\![C]\!]$.

In all cases, this is in contradiction with $\ell \in \text{in}_P[\![S_t]\!]$ so this case is impossible.⟆

$\{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![C]\!],\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in ((1_{\Sigma[\![P]\!]} \cup \tau^B) \circ \tau^\star[\![S_t]\!] \circ (1_{\Sigma[\![P]\!]} \cup \tau^t)) \wedge \ell \in \text{in}_P[\![S_t]\!]\}$

$= \quad \wr$If $\langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle\ell,\ \rho\rangle\rangle \in (\tau^\star\llbracket S_t\rrbracket \circ (1_{\Sigma\llbracket P\rrbracket} \cup \tau^t))$ then

- either $\langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle\ell,\ \rho\rangle\rangle \in (\tau\llbracket S_t\rrbracket^+ \circ (1_{\Sigma\llbracket P\rrbracket} \cup \tau^t))$, in which case $\mathrm{at}_P\llbracket C\rrbracket \in \mathrm{in}_P\llbracket S_t\rrbracket$, which is impossible
- or $\langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle\ell,\ \rho\rangle\rangle \in \tau^t$, in which case $\mathrm{at}_P\llbracket C\rrbracket = \mathrm{after}_P\llbracket S_t\rrbracket$, which is excluded
- or $\langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle\ell,\ \rho\rangle\rangle \in 1_{\Sigma\llbracket P\rrbracket}$ and so $\mathrm{at}_P\llbracket C\rrbracket = \ell$ is contradiction with $\mathrm{at}_P\llbracket C\rrbracket \notin \mathrm{in}_P\llbracket S_t\rrbracket$

$\wr$

$\{\rho \mid \exists\rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle\ell,\ \rho\rangle\rangle \in (\tau^B \circ \tau^\star\llbracket S_t\rrbracket \circ (1_{\Sigma\llbracket P\rrbracket} \cup \tau^t)) \wedge \ell \in \mathrm{in}_P\llbracket S_t\rrbracket\}$

$= \quad \wr$def. composition $\circ\wr$

$\{\rho \mid \exists\rho' \in R : \exists\rho_1,\rho_2 \in \mathbb{R} : \exists\ell_1,\ell_2 \in \mathrm{in}_P\llbracket C\rrbracket : \ell \in \mathrm{in}_P\llbracket S_t\rrbracket \wedge \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle\ell_1,\ \rho_1\rangle\rangle \in \tau^B \wedge \langle\langle\ell_1,\ \rho_1\rangle,\ \langle\ell_2,\ \rho_2\rangle\rangle \in \tau^\star\llbracket S_t\rrbracket \wedge \langle\langle\ell_2,\ \rho_2\rangle,\ \langle\ell,\ \rho\rangle\rangle \in (1_{\Sigma\llbracket P\rrbracket} \cup \tau^t))\}$

$= \quad \wr$def. $\tau^B \stackrel{\text{def}}{=} \{\langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho\rangle,\ \langle \mathrm{at}_P\llbracket S_t\rrbracket,\ \rho\rangle\rangle \mid \rho' \vdash B \Rrightarrow \mathtt{tt}\}\wr$

$\{\rho \mid \exists\rho' \in R : \exists\rho_2 \in \mathbb{R} : \exists\ell_2 \in \mathrm{in}_P\llbracket C\rrbracket : \ell \in \mathrm{in}_P\llbracket S_t\rrbracket \wedge \rho' \vdash B \Rrightarrow \mathtt{tt} \wedge \langle\langle \mathrm{at}_P\llbracket S_t\rrbracket,\ \rho'\rangle,\ \langle\ell_2,\ \rho_2\rangle\rangle \in \tau^\star\llbracket S_t\rrbracket \wedge \langle\langle\ell_2,\ \rho_2\rangle,\ \langle\ell,\ \rho\rangle\rangle \in (1_{\Sigma\llbracket P\rrbracket} \cup \tau^t))\}$

$= \quad \wr$y def. $\tau^t \stackrel{\text{def}}{=} \{\langle\langle \mathrm{after}_P\llbracket S_t\rrbracket,\ \rho\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho\rangle\rangle \mid \rho \in \mathrm{Env}\llbracket P\rrbracket\}$, the case $\langle\langle\ell_2,\ \rho_2\rangle,\ \langle\ell,\ \rho\rangle\rangle \in \tau^t$ implies $\ell = \mathrm{after}_P\llbracket C\rrbracket$, in contradiction with $\ell \in \mathrm{in}_P\llbracket S_t\rrbracket\wr$

$\{\rho \mid \exists\rho' \in R : \exists\rho_2 \in \mathbb{R} : \exists\ell_2 \in \mathrm{in}_P\llbracket C\rrbracket : \ell \in \mathrm{in}_P\llbracket S_t\rrbracket \wedge \rho' \vdash B \Rrightarrow \mathtt{tt} \wedge \langle\langle \mathrm{at}_P\llbracket S_t\rrbracket,\ \rho'\rangle,\ \langle\ell_2,\ \rho_2\rangle\rangle \in \tau^\star\llbracket S_t\rrbracket \wedge \langle\langle\ell_2,\ \rho_2\rangle,\ \langle\ell,\ \rho\rangle\rangle \in 1_{\Sigma\llbracket P\rrbracket}\}$

$= \{\rho \mid \exists\rho' \in R : \ell \in \mathrm{in}_P\llbracket S_t\rrbracket \wedge \rho' \vdash B \Rrightarrow \mathtt{tt} \wedge \langle\langle \mathrm{at}_P\llbracket S_t\rrbracket,\ \rho'\rangle,\ \langle\ell,\ \rho\rangle\rangle \in \tau^\star\llbracket S_t\rrbracket\}$

$= \quad \wr\langle\langle \mathrm{at}_P\llbracket S_t\rrbracket,\ \rho'\rangle,\ \langle\ell,\ \rho\rangle\rangle \in \tau^\star\llbracket S_t\rrbracket$ implies $\ell \in \mathrm{in}_P\llbracket S_t\rrbracket\wr$

$\{\rho \mid \exists\rho' \in R : \rho' \vdash B \Rrightarrow \mathtt{tt} \wedge \langle\langle \mathrm{at}_P\llbracket S_t\rrbracket,\ \rho'\rangle,\ \langle\ell,\ \rho\rangle\rangle \in \tau^\star\llbracket S_t\rrbracket\}$

$= \{\rho \mid \exists\rho' \in \{\rho' \in R \mid \rho' \vdash B \Rrightarrow \mathtt{tt}\} : \langle\langle \mathrm{at}_P\llbracket S_t\rrbracket,\ \rho'\rangle,\ \langle\ell,\ \rho\rangle\rangle \in \tau^\star\llbracket S_t\rrbracket\}$

$= \{\rho \mid \exists\rho' \in \mathrm{Cbexp}\llbracket B\rrbracket R : \langle\langle \mathrm{at}_P\llbracket S_t\rrbracket,\ \rho'\rangle,\ \langle\ell,\ \rho\rangle\rangle \in \tau^\star\llbracket S_t\rrbracket\}$ $\wr$def. $\mathrm{Cbexp}\llbracket B\rrbracket R\wr$

$= \mathrm{Rcom}\llbracket S_t\rrbracket(\mathrm{Cbexp}\llbracket B\rrbracket R)\ell$ $\wr$def. $\mathrm{Rcom}\llbracket S_t\rrbracket\wr$

– $\mathrm{Rcom}\llbracket C\rrbracket R\ell$ where $\ell \in \mathrm{in}_P\llbracket S_f\rrbracket$

$= \mathrm{Rcom}\llbracket S_f\rrbracket(\mathrm{Cbexp}\llbracket T(\neg(B))\rrbracket R)\ell$, in the same way

– $\mathrm{Rcom}\llbracket C\rrbracket R(\mathrm{after}_P\llbracket C\rrbracket)$ where $C = \mathtt{if}\ B\ \mathtt{then}\ S_t\ \mathtt{else}\ S_f\ \mathtt{fi}$

$= \{\rho \mid \exists\rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho\rangle\rangle \in \tau^\star\llbracket C\rrbracket\}$

$= \{\rho \mid \exists\rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho\rangle\rangle \in ((1_{\Sigma\llbracket P\rrbracket} \cup \tau^B) \circ \tau^\star\llbracket S_t\rrbracket \circ (1_{\Sigma\llbracket P\rrbracket} \cup \tau^t) \cup (1_{\Sigma\llbracket P\rrbracket} \cup \tau^{\bar{B}}) \circ \tau^\star\llbracket S_f\rrbracket \circ (1_{\Sigma\llbracket P\rrbracket} \cup \tau^f))\}$

$= \quad \wr$The case $\langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho\rangle\rangle \in \tau^\star\llbracket S_t\rrbracket \circ (1_{\Sigma\llbracket P\rrbracket} \cup \tau^t)$ would imply $\mathrm{at}_P\llbracket C\rrbracket \in (\mathrm{in}_P\llbracket S_t\rrbracket \cup \{\mathrm{after}_P\llbracket S_t\rrbracket\})$ by def. $\tau^\star\llbracket S_t\rrbracket$ and τ^t, or $\mathrm{at}_P\llbracket C\rrbracket = \mathrm{after}_P\llbracket C\rrbracket$ which is impossible. The same way, $\langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho\rangle\rangle \in \tau^\star\llbracket S_f\rrbracket \circ (1_{\Sigma\llbracket P\rrbracket} \cup \tau^f)$ is impossible.$\wr$

$\{\rho \mid \exists\rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho\rangle\rangle \in (\tau^B \circ \tau^\star\llbracket S_t\rrbracket \circ (1_{\Sigma\llbracket P\rrbracket} \cup \tau^t) \cup \tau^{\bar{B}} \circ \tau^\star\llbracket S_f\rrbracket \circ (1_{\Sigma\llbracket P\rrbracket} \cup \tau^f))\}$

$= \quad \wr$If $\langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho\rangle\rangle \in \tau^B \circ \tau^\star\llbracket S_t\rrbracket$ then $\mathrm{after}_P\llbracket C\rrbracket \in \mathrm{in}_P\llbracket S_t\rrbracket$ by def. $\tau\llbracket S_t\rrbracket^+$ or $\mathrm{after}_P\llbracket C\rrbracket = \mathrm{at}_P\llbracket S_t\rrbracket$, which is impossible. The same way, $\langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho\rangle\rangle \in \tau^{\bar{B}} \circ \tau^\star\llbracket S_f\rrbracket$ is impossible.$\wr$

$\{\rho \mid \exists\rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho\rangle\rangle \in (\tau^B \circ \tau^\star\llbracket S_t\rrbracket \circ \tau^t \cup \tau^{\bar{B}} \circ \tau^\star\llbracket S_f\rrbracket \circ \tau^f)\}$

$= \{\rho \mid \exists\rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho\rangle\rangle \in \tau^B \circ \tau^\star\llbracket S_t\rrbracket \circ \tau^t\}$
$\cup \{\rho \mid \exists\rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho\rangle\rangle \in \tau^{\bar{B}} \circ \tau^\star\llbracket S_f\rrbracket \circ \tau^f\}$

Since both cases are identical, we handle the first one.

$\{\rho \mid \exists\rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho\rangle\rangle \in \tau^B \circ \tau^\star\llbracket S_t\rrbracket \circ \tau^t\}$

$= \{\rho \mid \exists\rho' \in R : \exists\rho_1,\rho_2 \in \mathrm{Env}\llbracket P\rrbracket : \exists : \ell_1,\ell_2 \in \mathrm{in}_P\llbracket C\rrbracket : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle\ell_1,\ \rho_1\rangle\rangle \in \tau^B \wedge \langle\langle\ell_1,\ \rho_1\rangle,\ \langle\ell_2,\ \rho_2\rangle\rangle \in \tau^\star\llbracket S_t\rrbracket \wedge \langle\langle\ell_2,\ \rho_2\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho\rangle\rangle \in \tau^t\}$ $\wr$by def. $\circ\wr$

$= \quad \wr$By def. $\tau^B \stackrel{\text{def}}{=} \{\langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho\rangle,\ \langle \mathrm{at}_P\llbracket S_t\rrbracket,\ \rho\rangle\rangle \mid \rho \vdash B \Rrightarrow \mathtt{tt}\}$, $\rho_1 = \rho'$, $\ell_1 = \mathrm{at}_P\llbracket S_t\rrbracket$ and $\rho' \vdash B \Rrightarrow \mathtt{tt}\wr$

$\{\rho \mid \exists\rho' \in R : \exists\rho_2 \in \mathrm{Env}\llbracket P\rrbracket : \exists : \ell_2 \in \mathrm{in}_P\llbracket C\rrbracket : \rho' \vdash B \Rrightarrow \mathtt{tt} \wedge \langle\langle \mathrm{at}_P\llbracket S_t\rrbracket,\ \rho'\rangle,\ \langle\ell_2,\ \rho_2\rangle\rangle \in \tau^\star\llbracket S_t\rrbracket \wedge \langle\langle\ell_2,\ \rho_2\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho\rangle\rangle \in \tau^t\}$

$= \quad \wr$By def. $\tau^t \stackrel{\text{def}}{=} \{\langle\langle \mathrm{after}_P\llbracket S_t\rrbracket,\ \rho\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho\rangle\rangle \mid \rho \in \mathrm{Env}\llbracket P\rrbracket\}$, we have $\ell_2 = \mathrm{after}_P\llbracket S_t\rrbracket$ and $\rho_2 = \rho\wr$

$\{\rho \mid \exists\rho' \in R : \rho' \vdash B \Rrightarrow \mathtt{tt} \wedge \langle\langle \mathrm{at}_P\llbracket S_t\rrbracket,\ \rho'\rangle,\ \langle \mathrm{after}_P\llbracket S_t\rrbracket,\ \rho\rangle\rangle \in \tau^\star\llbracket S_t\rrbracket\}$

$$= \{\rho \mid \exists \rho' \in \{\rho' \in\in R : \rho' \vdash B \Rrightarrow tt\} : \langle\langle \mathrm{at}_P\llbracket S_t\rrbracket,\ \rho'\rangle,\ \langle \mathrm{after}_P\llbracket S_t\rrbracket,\ \rho\rangle\rangle \in \tau^\star\llbracket S_t\rrbracket\}$$
$$= \{\rho \mid \exists \rho' \in \mathrm{Cbexp}\llbracket B\rrbracket R : \langle\langle \mathrm{at}_P\llbracket S_t\rrbracket,\ \rho'\rangle,\ \langle \mathrm{after}_P\llbracket S_t\rrbracket,\ \rho\rangle\rangle \in \tau^\star\llbracket S_t\rrbracket\}$$
$$= \mathrm{Rcom}\llbracket S_t\rrbracket(\mathrm{Cbexp}\llbracket B\rrbracket R)(\mathrm{after}_P\llbracket S_t\rrbracket)$$

The same way, we have

$$\{\rho \mid \exists \rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \mathrm{after}_P\llbracket C\rrbracket,\ \rho\rangle\rangle \in \tau^{\bar{B}} \circ \tau^\star\llbracket S_f\rrbracket \circ \tau^f\}$$
$$= \mathrm{Rcom}\llbracket S_f\rrbracket(\mathrm{Cbexp}\llbracket T(\neg(B))\rrbracket R)(\mathrm{after}_P\llbracket S_f\rrbracket)$$

— $\mathrm{Rcom}\llbracket C\rrbracket R\ell$ where $C = \texttt{while } B \texttt{ do } S \texttt{ od}$

$$= \{\rho \mid \exists \rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in \tau^\star\llbracket C\rrbracket\}$$
$$= \{\rho \mid \exists \rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in (((1_{\Sigma\llbracket P\rrbracket} \cup \tau^\star\llbracket S\rrbracket \circ \tau^R) \circ (\tau^B \circ \tau^\star\llbracket S\rrbracket \circ \tau^R)^\star \circ (1_{\Sigma\llbracket P\rrbracket} \cup \tau^B \circ \tau^\star\llbracket S\rrbracket \cup \tau^{\bar{B}})) \cup \tau\llbracket S\rrbracket^\star)\}$$
$= \quad \wr$The case $\langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in \tau\llbracket S\rrbracket^\star$ implies $\mathrm{at}_P\llbracket C\rrbracket \in \mathrm{in}_P\llbracket S\rrbracket$, which is impossible$\wr$

$$\{\rho \mid \exists \rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in ((1_{\Sigma\llbracket P\rrbracket} \cup \tau^\star\llbracket S\rrbracket \circ \tau^R) \circ (\tau^B \circ \tau^\star\llbracket S\rrbracket \circ \tau^R)^\star \circ (1_{\Sigma\llbracket P\rrbracket} \cup \tau^B \circ \tau^\star\llbracket S\rrbracket \cup \tau^{\bar{B}}))\}$$
$= \quad \wr$The same way, $\langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in ((\tau^\star\llbracket S\rrbracket \circ \tau^R) \circ (\tau^B \circ \tau^\star\llbracket S\rrbracket \circ \tau^R)^\star \circ (1_{\Sigma\llbracket P\rrbracket} \cup \tau^B \circ \tau^\star\llbracket S\rrbracket \cup \tau^{\bar{B}}))$ implies either $\mathrm{at}_P\llbracket C\rrbracket \in \mathrm{in}_P\llbracket S\rrbracket$ by def. $\tau\llbracket S\rrbracket$ or $\mathrm{at}_P\llbracket C\rrbracket = \mathrm{after}_P\llbracket S\rrbracket$, by def. τ^R and both cases are impossible$\wr$

$$\{\rho \mid \exists \rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in ((\tau^B \circ \tau^\star\llbracket S\rrbracket \circ \tau^R)^\star \circ (1_{\Sigma\llbracket P\rrbracket} \cup \tau^B \circ \tau^\star\llbracket S\rrbracket \cup \tau^{\bar{B}}))\}$$
$$= \{\rho \mid \exists \rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in (\tau^B \circ \tau^\star\llbracket S\rrbracket \circ \tau^R)^\star\}$$
$$\cup\ \{\rho \mid \exists \rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in (\tau^B \circ \tau^\star\llbracket S\rrbracket \circ \tau^R)^\star \circ \tau^B \circ \tau^\star\llbracket S\rrbracket\}$$
$$\cup\ \{\rho \mid \exists \rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in (\tau^B \circ \tau^\star\llbracket S\rrbracket \circ \tau^R)^\star \circ \tau^{\bar{B}}\}$$

We study all three cases separately.

- $\{\rho \mid \exists \rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in (\tau^B \circ \tau^\star\llbracket S\rrbracket \circ \tau^R)^\star\}$

$= \quad \wr$
– in case the reflexive transitive closure is the identity, then obviously $\ell = \mathrm{at}_P\llbracket C\rrbracket$;
– in case the reflexive transitive closure is not the identity, then, by definition of $\tau^B \stackrel{\text{def}}{=} \{\langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho\rangle,\ \langle \mathrm{at}_P\llbracket S\rrbracket,\ \rho\rangle\rangle \mid \rho \vdash B \Rrightarrow tt\}$, $\ell = \mathrm{at}_P\llbracket C\rrbracket$.

So when $\ell \neq \mathrm{at}_P\llbracket C\rrbracket$, the expression is empty.$\wr$

match ℓ with
$\mid \mathrm{at}_P\llbracket C\rrbracket \to \underbrace{\{\rho \mid \exists \rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho\rangle\rangle \in (\tau^B \circ \tau^\star\llbracket S\rrbracket \circ \tau^R)^\star\}}_{I}$
$\mid\ \_ \to \emptyset$

$= \quad \wr$as already computed on pages 27 to 29$\wr$

match ℓ with
$\mid \mathrm{at}_P\llbracket C\rrbracket \to \mathbf{lfp}_{\emptyset}^{\subseteq}\ \lambda X \cdot R \cup \mathrm{Pcom}\llbracket S\rrbracket(\mathrm{Cbexp}\llbracket B\rrbracket(X))$
$\mid\ \_ \to \emptyset$

= match ℓ with
$\mid \mathrm{at}_P\llbracket C\rrbracket \to \underbrace{\mathbf{lfp}_{\emptyset}^{\subseteq}\ \lambda X \cdot R \cup \mathrm{Rcom}\llbracket S\rrbracket\ \mathrm{Cbexp}\llbracket B\rrbracket(X)(\mathrm{after}_P\llbracket S\rrbracket)}_{I}$
$\mid\ \_ \to \emptyset$ $\quad\quad \wr$by (35)$\wr$

- $\{\rho \mid \exists \rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in (\tau^B \circ \tau^\star\llbracket S\rrbracket \circ \tau^R)^\star \circ \tau^B \circ \tau^\star\llbracket S\rrbracket\}$

$= \quad \wr$def. composition $\circ\wr$

$$\{\rho \mid \exists \rho' \in R : \exists \rho_1, \rho_2 \in \mathbb{R} : \exists \ell_1, \ell_2 \in \mathrm{in}_P\llbracket C\rrbracket : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \ell_1,\ \rho_1\rangle\rangle \in (\tau^B \circ \tau^\star\llbracket S\rrbracket \circ \tau^R)^\star \wedge \langle\langle \ell_1,\ \rho_1\rangle,\ \langle \ell_2,\ \rho_2\rangle\rangle \in \tau^B \wedge \langle\langle \ell_2,\ \rho_2\rangle,\ \langle \ell,\ \rho\rangle\rangle \in \tau^\star\llbracket S\rrbracket\}$$

$= \quad \wr$By def. $\tau^B \stackrel{\text{def}}{=} \{\langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho\rangle,\ \langle \mathrm{at}_P\llbracket S\rrbracket,\ \rho\rangle\rangle \mid \rho \vdash B \Rrightarrow tt\}$, we have $\ell_1 = \mathrm{at}_P\llbracket C\rrbracket$, $\ell_2 = \mathrm{at}_P\llbracket S\rrbracket$, $\rho_1 = \rho_2$ and $\rho_1 \vdash B \Rrightarrow tt\wr$

$$\{\rho \mid \exists \rho' \in R : \exists \rho_1 \in \mathbb{R} : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho_1\rangle\rangle \in (\tau^B \circ \tau^\star\llbracket S\rrbracket \circ \tau^R)^\star \wedge \rho_1 \vdash B \Rrightarrow tt \wedge \langle\langle \mathrm{at}_P\llbracket S\rrbracket,\ \rho_1\rangle,\ \langle \ell,\ \rho\rangle\rangle \in \tau^\star\llbracket S\rrbracket\}$$

$= \quad \wr$By def. $I = \{\rho \mid \exists \rho' \in R : \langle\langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho'\rangle,\ \langle \mathrm{at}_P\llbracket C\rrbracket,\ \rho\rangle\rangle \in (\tau^B \circ \tau^\star\llbracket S\rrbracket \circ \tau^R)^\star\}\wr$

$$\{\rho \mid \exists \rho_1 \in \{\rho_1 \in I \mid \rho_1 \vdash B \Rrightarrow tt\} : \langle\langle \mathrm{at}_P\llbracket S\rrbracket,\ \rho_1\rangle,\ \langle \ell,\ \rho\rangle\rangle \in \tau^\star\llbracket S\rrbracket\}$$
$$= \{\rho \mid \exists \rho_1 \in \mathrm{Cbexp}\llbracket B\rrbracket I : \langle\langle \mathrm{at}_P\llbracket S\rrbracket,\ \rho_1\rangle,\ \langle \ell,\ \rho\rangle\rangle \in \tau^\star\llbracket S\rrbracket\}$$

$= \; (\!|\, \ell \in \text{in}_P[\![S]\!] \;?\; \text{Rcom}[\![S]\!](\text{Cbexp}[\![B]\!]I)\ell \;\grave{\imath}\; \emptyset \,|\!)$

- $\{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![C]\!],\, \rho'\rangle,\, \langle \ell,\, \rho\rangle\rangle \in (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \circ \tau^{\bar{B}}\}$

$=$ $\wr$def. composition $\circ\int$

$\{\rho \mid \exists \rho' \in R : \exists \rho_1 \in \mathbb{R} : \exists \ell_1 \in \text{in}_P[\![C]\!] : \langle\langle \text{at}_P[\![C]\!],\, \rho'\rangle,\, \langle \ell_1,\, \rho_1\rangle\rangle \in (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \wedge \langle\langle \ell_1,\, \rho_1\rangle,\, \langle \ell,\, \rho\rangle\rangle \in \tau^{\bar{B}}\}$

$=$ $\wr$def. $\tau^{\bar{B}} \stackrel{\text{def}}{=} \{\langle\langle \text{at}_P[\![C]\!],\, \rho\rangle,\, \langle \text{after}_P[\![C]\!],\, \rho\rangle\rangle \mid \rho \vdash T(\neg B) \Rightarrow \text{tt}\}$ so that $\ell_1 = \text{at}_P[\![C]\!]$, $\rho_1 = \rho$, $\ell = \text{after}_P[\![C]\!]$ and $\rho \vdash T(\neg B) \Rightarrow \text{tt}\int$

$\{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![C]\!],\, \rho'\rangle,\, \langle \text{at}_P[\![C]\!],\, \rho\rangle\rangle \in (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \wedge \rho \vdash T(\neg B) \Rightarrow \text{tt} \wedge \ell = \text{after}_P[\![C]\!]\}$

$=$ match ℓ with
| $\text{after}_P[\![C]\!] \to \{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![C]\!],\, \rho'\rangle,\, \langle \text{at}_P[\![C]\!],\, \rho\rangle\rangle \in (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star \wedge \rho \vdash T(\neg B) \Rightarrow \text{tt}\}$
| $\_ \to \emptyset$

$=$ match ℓ with
| $\text{after}_P[\![C]\!] \to \{\rho \in \{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![C]\!],\, \rho'\rangle,\, \langle \text{at}_P[\![C]\!],\, \rho\rangle\rangle \in (\tau^B \circ \tau^\star[\![S]\!] \circ \tau^R)^\star\} \mid \rho \vdash T(\neg B) \Rightarrow \text{tt}\}$
| $\_ \to \emptyset$

$=$ match ℓ with
| $\text{after}_P[\![C]\!] \to \{\rho \in I \mid \rho \vdash T(\neg B) \Rightarrow \text{tt}\}$
| $\_ \to \emptyset$

$=$ match ℓ with
| $\text{after}_P[\![C]\!] \to \text{Cbexp}[\![T(\neg(B))]\!]I$
| $\_ \to \emptyset$

- Grouping all cases of the $\cup$ together, we get:

$\text{Rcom}[\![C]\!]R\ell$ where $C = \texttt{while}\ B\ \texttt{do}\ S\ \texttt{od}$

$=$ let $I = \textsf{lfp}^{\subseteq}_{\emptyset}\; \lambda X \cdot E \cup \text{Rcom}[\![S]\!](\text{Cbexp}[\![B]\!]X)(\text{after}_P[\![S]\!])$ in
match ℓ with
| $\text{at}_P[\![C]\!] \to I$
| $\text{in}_P[\![S]\!] \to \text{Rcom}[\![S]\!](\text{Cbexp}[\![B]\!]I)(\ell)$
| $\text{after}_P[\![C]\!] \to \text{Cbexp}[\![T(\neg(B))]\!]R)I$

– $\text{Rcom}[\![C\ ;\ S]\!]R\ell$

$= \{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![C]\!],\, \rho'\rangle,\, \langle \ell,\, \rho\rangle\rangle \in \tau^\star[\![C\ ;\ S]\!]\}$

$= \{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![C]\!],\, \rho'\rangle,\, \langle \ell,\, \rho\rangle\rangle \in \tau^\star[\![C]\!] \circ \tau^\star[\![S]\!]\}$

$= \{\rho \mid \exists \rho' \in R : \exists \rho_1 \in \mathbb{R} : \exists \ell_1 \in \text{in}_P[\![C]\!] : \langle\langle \text{at}_P[\![C]\!],\, \rho'\rangle,\, \langle \ell_1,\, \rho_1\rangle\rangle \in \tau^\star[\![C]\!] \wedge \langle\langle \ell_1,\, \rho_1\rangle,\, \langle \ell,\, \rho\rangle\rangle \in \tau^\star[\![S]\!]\}$

$=$ $\wr$Distinguishing the cases when $\ell\,\text{in}_P[\![C]\!]$ (when $\tau^\star[\![S]\!]$ is the identity) or $\ell\,\text{in}_P[\![S]\!]$ and noting in this second case that $\ell_1 = \text{after}_P[\![C]\!] = \text{at}_P[\![S]\!] \in \text{in}_P[\![C]\!] \cap \text{in}_P[\![S]\!]\int$

match ℓ with
| $\text{in}_P[\![C]\!] \to \{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![C]\!],\, \rho'\rangle,\, \langle \ell,\, \rho\rangle\rangle \in \tau^\star[\![C]\!]\}$
| $\text{in}_P[\![S]\!] \to \{\rho \mid \exists \rho' \in R : \exists \rho_1 \in \mathbb{R} : \langle\langle \text{at}_P[\![C]\!],\, \rho'\rangle,\, \langle \text{after}_P[\![C]\!],\, \rho_1\rangle\rangle \in \tau^\star[\![C]\!] \wedge \langle\langle \text{at}_P[\![S]\!],\, \rho_1\rangle,\, \langle \ell,\, \rho\rangle\rangle \in \tau^\star[\![S]\!]\}$
| $\_ \to \emptyset$

$=$ $\wr$The last case is indeed impossible since $\ell \in \text{in}_P[\![C\ ;\ S]\!] = \text{in}_P[\![C]\!] \cup \text{in}_P[\![S]\!]\int$

match ℓ with
| $\text{in}_P[\![C]\!] \to \{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![C]\!],\, \rho'\rangle,\, \langle \ell,\, \rho\rangle\rangle \in \tau^\star[\![C]\!]\}$
| $\text{in}_P[\![S]\!] \to \{\rho \mid \exists \rho_1 \in \{\rho_1 \in \mathbb{R} \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![C]\!],\, \rho'\rangle,\, \langle \text{after}_P[\![C]\!],\, \rho_1\rangle\rangle \in \tau^\star[\![C]\!]\} : \langle\langle \text{at}_P[\![S]\!],\, \rho_1\rangle,\, \langle \ell,\, \rho\rangle\rangle \in \tau^\star[\![S]\!]\}$

$=$ $\wr$def. $\text{Rcom}[\![C]\!]R\ell \stackrel{\text{def}}{=} \{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![C]\!],\, \rho'\rangle,\, \langle \ell,\, \rho\rangle\rangle \in \tau^\star[\![C]\!]\}\int$

match ℓ with
| $\text{in}_P[\![C]\!] \to \text{Rcom}[\![C]\!]R\ell$
| $\text{in}_P[\![S]\!] \to \{\rho \mid \exists \rho_1 \in \text{Rcom}[\![C]\!]R(\text{after}_P[\![C]\!]) : \langle\langle \text{at}_P[\![S]\!],\, \rho_1\rangle,\, \langle \ell,\, \rho\rangle\rangle \in \tau^\star[\![S]\!]\}$

$=$ $\wr$def. $\text{Rcom}[\![C]\!]R\ell \stackrel{\text{def}}{=} \{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![C]\!],\, \rho'\rangle,\, \langle \ell,\, \rho\rangle\rangle \in \tau^\star[\![C]\!]\}\int$

match ℓ with
| $\text{in}_P[\![C]\!] \rightarrow \text{Rcom}[\![C]\!]R\ell$
| $\text{in}_P[\![S]\!] \rightarrow \text{Rcom}[\![S]\!](\text{Rcom}[\![C]\!]R(\text{after}_P[\![C]\!]))\ell$

$$
\begin{aligned}
&- \text{Rcom}[\![S\ ;;]\!]R\ell \\
&= \{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![S\ ;;]\!],\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in \tau^\star[\![S\ ;;]\!]\} \\
&= \quad \wr \text{at}_P[\![S\ ;;]\!] = \text{at}_P[\![S]\!] \text{ and } \tau^\star[\![S\ ;;]\!] = \tau^\star[\![S]\!] \wr \\
&\quad \{\rho \mid \exists \rho' \in R : \langle\langle \text{at}_P[\![S]\!],\ \rho'\rangle,\ \langle \ell,\ \rho\rangle\rangle \in \tau^\star[\![S]\!]\} \\
&= \text{Rcom}[\![S]\!]R\ell
\end{aligned}
$$

□

Implementation of the forward reachability collecting

Implementability of the forward reachability collecting semantics

- The foward reachability collecting semantics collects all values and environments by simulation of all possible executions.
- The main problem is random input (?) because of the very large number of possible values (e.g. between `min_int=-1.073.741.834` and `max_int=-1.073.741.833`) which makes exhaustive simulation impossible in practice
- We will limit the possible values of the random input (?) to 3 instead of 2.147.483.648

- Apart from this restriction, the implementation is faithful to the definition of the forward reachability collecting semantics in that it simulates all possible executions of the program (for 3 of the possible random values at each random expression computation)
- Despite this restriction, the forward reachability collecting semantics is rapidly subject to a combinatorial explosion of the states and therefore is useless in practice.

Example: trace of the fixpoint computation with two nested loops

I — no iteration within the loops

```
% cat ../Examples/example12-1.sil
% example12-1.sil %
n := 1;
x := 1;
while (x < n) do
        x := x + 1;
        a := ?;
        y := 1;
        while (y < n) do
                y := y + 1;
                b := ?
        od
od;;
```



```
% ./a.out ../Examples/example12-1.sil
...

iterate 0 = { }
iterate 0 = { }
fixpoint = { }
iterate 1 = { [ n = 1; x = 1; a = _O_(i); y = _O_(i); b = _O_(i); ] }
iterate 0 = { }
fixpoint = { }
fixpoint = { [ n = 1; x = 1; a = _O_(i); y = _O_(i); b = _O_(i); ] }
{ [ n = 1; x = 1; a = _O_(i); y = _O_(i); b = _O_(i); ] }
% ^Dexit

Script done on Mon Apr  4 12:37:52 2005
```


II — 10 iterations within the loops

Page 1:

```
Script started on Mon Apr  4 12:45:41 2005
% make trace

ocamlyacc parser.mly
ocamllex lexer.mll
62 states, 3001 transitions, table size 12376 bytes
ocamlc symbol_Table.mli symbol_Table.ml variables.mli variables.ml
abstract_Syntax.ml concrete_To_Abstract_Syntax.mli
concrete_To_Abstract_Syntax.ml labels.mli labels.ml parser.mli
parser.ml lexer.ml program_To_Abstract_Syntax.mli
program_To_Abstract_Syntax.ml pretty_Print.mli pretty_Print.ml
values.mli values.ml cvalues.mli cvalues.ml env.mli env.ml cenv.mli
cenv.ml caexp.mli caexp.ml cbexp.mli cbexp.ml fixpoint.mli fixpoint.ml
ccom.mli ccom.ml main.ml
fixpoint tracing mode
```



```
% cat ../Examples/example12-10.sil

% example12-10.sil %
n := 10;
x := 1;
while (x < n) do
x := x + 1;
a := ?;
y := 1;
while (y < n) do
y := y + 1;
b := ?
od
od;;
```

```
% ./a.out ../Examples/example12-10.sil
...
iterate 0 = { }
iterate 0 = { }
fixpoint = { }
iterate 1 = { [ n = 10; x = 1; a = _O_(i); y = _O_(i); b = _O_(i); ] }
iterate 0 = { }
iterate 1 = { [ n = 10; x = 2; a = -819618235; y = 1; b = _O_(i); ]
              [ n = 10; x = 2; a = -361540549; y = 1; b = _O_(i); ]
              [ n = 10; x = 2; a = 625724514; y = 1; b = _O_(i); ] }
iterate 2 = { [ n = 10; x = 2; a = -819618235; y = 1; b = _O_(i); ]
              [ n = 10; x = 2; a = -819618235; y = 2; b = -819618235; ]
              [ n = 10; x = 2; a = -819618235; y = 2; b = -361540549; ]
              [ n = 10; x = 2; a = -819618235; y = 2; b = 625724514; ]
              [ n = 10; x = 2; a = -361540549; y = 1; b = _O_(i); ]
              [ n = 10; x = 2; a = -361540549; y = 2; b = -819618235; ]
```


Page 471 (showing the invariant on loop exit):

```
{ [ n = 10; x = 10; a = -819618235; y = 10; b = -819618235; ]
  [ n = 10; x = 10; a = -819618235; y = 10; b = -361540549; ]
  [ n = 10; x = 10; a = -819618235; y = 10; b = 625724514; ]
  [ n = 10; x = 10; a = -361540549; y = 10; b = -819618235; ]
  [ n = 10; x = 10; a = -361540549; y = 10; b = -361540549; ]
  [ n = 10; x = 10; a = -361540549; y = 10; b = 625724514; ]
  [ n = 10; x = 10; a = 625724514; y = 10; b = -819618235; ]
  [ n = 10; x = 10; a = 625724514; y = 10; b = -361540549; ]
  [ n = 10; x = 10; a = 625724514; y = 10; b = 625724514; ] }
```

An exhaustive simulation would involve 2.147.483.648 $\times$ 2.147.483.648 cases, instead of the above $3 \times 3 = 9$. Note that choosing *different* random values at each random assignment would ultimately cover all machine integers whence explode combinatorially.

Totally ordered types in OCaml

From `http://caml.inria.fr/pub/docs/manual-ocaml/libref/Set.OrderedType.html`:

`module type OrderedType = sig .. end`

Input signature of the functor `Set.Make`

`type t`

The type of the set elements.

`val compare : t -> t -> int`

A total ordering function over the set elements. This is a two-argument function `f` such that `f e1 e2` is zero if the elements `e1` and `e2` are equal, `f e1 e2` is strictly negative if `e1` is smaller than `e2`, and `f e1 e2` is strictly positive if `e1` is greater than `e2`.

Implementation: sets in OCaml

From `http://caml.inria.fr/pub/docs/manual-ocaml/libref/Set.html`:

`module Set: sig .. end`

Sets over ordered types implemented using balanced binary trees, no side-effects.

`module type S = sig .. end`

Output signature of the functor `Set.Make`

`module Make: functor (Ord : OrderedType) -> S with type elt = Ord.t`

Functor building an implementation of the set structure given a totally ordered type.

Example: ordered set of machine integers

```
(* ordered set of machine integers *)
include Set.Make
   (struct
      type t = machine_int
      (* order on values *)
      let compare v1 v2 = match v1, v2 with
         | (ERROR_NAT INITIALIZATION), (ERROR_NAT INITIALIZATION) -> 0
         | (ERROR_NAT INITIALIZATION), (ERROR_NAT ARITHMETIC) -> -1
         | (ERROR_NAT ARITHMETIC), (ERROR_NAT INITIALIZATION) -> 1
         | (ERROR_NAT ARITHMETIC), (ERROR_NAT ARITHMETIC) -> 0
         | (ERROR_NAT e), (NAT i) -> -1
         | (NAT i), (ERROR_NAT e) -> 1
         | (NAT i), (NAT j) -> if (i<j) then -1 else
                                  if (i=j) then 0 else 1
   end)
```


Signatures

From `http://caml.inria.fr/pub/docs/manual-ocaml/libref/type_Set.html`:

```
module Make :
  functor (Ord : OrderedType) ->
    sig
      type elt = Ord.t
      type t
      val add : elt -> t -> t
      val elements : t -> elt list
      val empty : t
      val equal : t -> t -> bool
      val filter : (elt -> bool) -> t -> t
      val fold : (elt -> 'a -> 'a) -> t -> 'a -> 'a
      val for_all : (elt -> bool) -> t -> bool
      val inter : t -> t -> t
      val is_empty : t -> bool
      val iter : (elt -> unit) -> t -> unit
      val singleton : elt -> t
      val subset : t -> t -> bool
      val union : t -> t -> t
    end
```


Specification of set operations

- `add` $e\ s \stackrel{\text{def}}{=} s \cup \{e\}$
- `elements` $\emptyset \stackrel{\text{def}}{=}$ `[]`
- `elements` $\{a_1, \ldots, a_n\} \stackrel{\text{def}}{=} [a_1; \ldots; a_n]$
- `empty` $\stackrel{\text{def}}{=} \emptyset$
- `equal` $s_1\ s_2 \stackrel{\text{def}}{=} (s_1 = s_2\ ?\ \mathrm{tt}\ ¿\ \mathrm{ff})$
- `filter` $p\ s \stackrel{\text{def}}{=} \{x \in s \mid p(x)\}$
- `fold` $f\ \emptyset\ b \stackrel{\text{def}}{=} b$
- `fold` $f\ \{a_1, \ldots, a_n\}\ b \stackrel{\text{def}}{=} f\ a_1\ (f a_2\ (\ldots (f\ a_n\ b) \ldots))$

- `inter` $s_1\ s_2 \stackrel{\text{def}}{=} s_1 \cap s_2$
- `is_empty` $s \stackrel{\text{def}}{=} (s = \emptyset\ ?\ \mathrm{tt}\ ¿\ \mathrm{ff})$
- `iter` $f\ \emptyset = ()$
- `iter` $f\ \{a_1, \ldots, a_n\} = f\ a_1;\ f\ a_2;\ \ldots;\ f\ a_n$
- `singleton` $e \stackrel{\text{def}}{=} \{e\}$
- `subset` $s_1\ s_2 \stackrel{\text{def}}{=} (s_1 \subseteq s_2\ ?\ \mathrm{tt}\ ¿\ \mathrm{ff})$
- `union` $s_1\ s_2 \stackrel{\text{def}}{=} s_1 \cup s_2$

Implementation: sets of values

```
(* cvalues.mli *)
open Values
(* set of machine integers *)
type elt = machine_int
and t
val add : elt -> t -> t
val singleton : elt -> t
val fold : (elt -> 'a -> 'a) -> t -> 'a -> 'a
val iter : (elt -> unit) -> t -> unit
val bot : unit -> t
val isbotempty : unit -> bool
val initerr : unit -> t
val top : unit -> 'a
val join : t -> t -> t
val meet : t -> t -> t
```



```
val leq : t -> t -> bool
val eq : t -> t -> bool
val in_errors : t -> bool
val print : t -> unit
(* forward collecting semantics of arithmetic expressions *)
val f_NAT : string -> t
val f_RANDOM : unit -> t
val f_UMINUS : t -> t
val f_UPLUS : t -> t
val f_PLUS : t -> t -> t
val f_MINUS : t -> t -> t
val f_TIMES : t -> t -> t
val f_DIV : t -> t -> t
val f_MOD : t -> t -> t
```



```
(* cvalues.ml *)
open Values
(* ordered set of machine integers *)
include Set.Make
 (struct
   type t = machine_int
   (* order on values *)
   let compare v1 v2 = match v1, v2 with
   | (ERROR_NAT INITIALIZATION), (ERROR_NAT INITIALIZATION) -> 0
   | (ERROR_NAT INITIALIZATION), (ERROR_NAT ARITHMETIC) -> -1
   | (ERROR_NAT ARITHMETIC), (ERROR_NAT INITIALIZATION) -> 1
   | (ERROR_NAT ARITHMETIC), (ERROR_NAT ARITHMETIC) -> 0
   | (ERROR_NAT e), (NAT i) -> -1
   | (NAT i), (ERROR_NAT e) -> 1
   | (NAT i), (NAT j) -> if (i<j) then -1 else if (i=j) then 0 else 1
  end)
(* infimum *)
```



```
let bot () = empty
(* bottom is emptyset? *)
let isbotempty () = true
(* uninitialization *)
let initerr () = singleton (ERROR_NAT INITIALIZATION)
(* supremum *)
exception ErrorCvalues of string
let top () = raise (ErrorCvalues "top not implemented")
(* least upper bound *)
let join = union
(* greatest lower bound *)
let meet = inter
(* approximation ordering *)
let leq = subset
(* equality *)
let eq = equal
(* included in errors? *)
let in_errors v =
```

```
  let iserror i = match i with
    | (ERROR_NAT e) -> true
    | (NAT j) -> false
  in for_all iserror v
(* printing *)
let print v =
   let printelement e =
      print_machine_int e;
      print_string " "
   in
      print_string "{ ";
     iter printelement v;
      print_string " }"
(* image u s = { u(x) | x in s } *)
let image u s =
    let f e s' = add (u e) s' in
      fold  f s empty
(* set_bin b s1 s2 = { b(x,y) | x in s1 /\ y in s2 } *)
```



```
let set_bin b s1 s2 =
    let f a2 s =
      let g a1 s = add (b a1 a2) s
      in  fold g s1 empty
    in
       fold f s2 empty
(* forward collecting semantics of arithmetic expressions *)
let f_NAT s =
   singleton (machine_int_of_string s)
let r1 = (machine_unary_random ())
let r2 = (machine_unary_random ())
let r3 = (machine_unary_random ())
let f_RANDOM () =
  (* should be the set of all possible values! *)
  add r1 (add r2 (singleton r3))
let f_UMINUS a =
   image machine_unary_minus a
let f_UPLUS a =
```



```
   image machine_unary_plus a
let f_PLUS a1 a2 = set_bin machine_binary_plus a1 a2
let f_MINUS a1 a2 = set_bin machine_binary_minus a1 a2
let f_TIMES a1 a2 = set_bin machine_binary_times a1 a2
let f_DIV a1 a2 = set_bin machine_binary_div a1 a2
let f_MOD a1 a2 = set_bin machine_binary_mod a1 a2
```


Implementation: sets of environments

```
(* cenv.mli *)
open Abstract_Syntax
open Cvalues
open Env
(* set of environments *)
type elt = Env.env
and t
(* infimum *)
val bot : unit -> t
(* check for infimum *)
val is_bot : t -> bool
(* uninitialization *)
val initerr : unit -> t
(* supremum *)
val top : unit -> 'a
```

```
(* copy *)
val copy : t -> t
(* least upper bound *)
val join : t -> t -> t
(* greatest lower bound *)
val meet : t -> t -> t
(* approximation ordering *)
val leq : t -> t -> bool
(* equality *)
val eq : t -> t -> bool
(* printing *)
val print : t -> unit
(* r(X) = {e(X) | X in r} *)
val get : t -> variable -> Cvalues.t
(* r[X <- i] = {e[X <- i] | e in r } *)
val set_elem : t -> variable -> Values.machine_int -> t
(* r[X <- v] = {e[X <- i] | e in r /\ i in v}    *)
val set : t -> variable -> Cvalues.t -> t
```



```
(* collecting semantics of assignment                              *)
(* f_ASSIGN x f r =  {e[x <- i] | e in r /\ i in f({e}) cap I }   *)
val f_ASSIGN : variable -> (t -> Cvalues.t) -> t -> t
(* collecting semantics of boolean expressions                     *)
(* f_EQ f g r =                                                    *)
(*   {e in r | exists v1 in f({e}) cap I: exists v2 in g({e}) cap *)
(*             I: v1 = v2 }                                        *)
val f_EQ : (t -> Cvalues.t) -> (t -> Cvalues.t) -> t -> t
(* f_LT f g r =                                                    *)
(*   {e in R | exists v1 in f({e}) cap I: exists v2 in g({e}) cap *)
(*             I: v1 < v2 }                                        *)
val f_LT : (t -> Cvalues.t) -> (t -> Cvalues.t) -> t -> t
```



```
(* cenv.ml *)
open Variables
open Values
open Cvalues
open Env
(* order on values *)
let compare_values v1 v2 = match v1, v2 with
  | (ERROR_NAT INITIALIZATION), (ERROR_NAT INITIALIZATION) -> 0
  | (ERROR_NAT INITIALIZATION), (ERROR_NAT ARITHMETIC) -> -1
  | (ERROR_NAT ARITHMETIC), (ERROR_NAT INITIALIZATION) -> 1
  | (ERROR_NAT ARITHMETIC), (ERROR_NAT ARITHMETIC) -> 0
  | (ERROR_NAT e), (NAT i) -> -1
  | (NAT i), (ERROR_NAT e) -> 1
  | (NAT i), (NAT j) -> if (i<j) then -1 else if (i=j) then 0 else 1
(* ordered set of environments *)
exception Found of int
include Set.Make
```



```
   (struct
      type t = env
      (* order on environments *)
      let compare r1 r2 =
        try
          for i = 0 to ((number_of_variables ()) - 1) do
            let c = compare_values (get r1 i) (get r2 i) in
              if c != 0 then   raise (Found c)
          done;
          0
        with Found c -> c
   end)
(* infimum *)
let bot () = empty
(* check for infimum *)
let is_bot = is_empty
(* uninitialization *)
let initerr () = singleton (Env.initerr ())
```

```
(* supremum *)
exception ErrorCenv of string
let top () = raise (ErrorCenv "top not implemented")
(* copy *)
let copy s = s (* implementation without side-effects *)
(* least upper bound *)
let join = union
(* greatest lower bound *)
let meet = inter
(* approximation ordering *)
let leq = subset
(* equality *)
let eq = equal
(* printing *)
let print r =
   print_string "{ ";
   let pe e = (print_string "[ ";print_env e;print_string "] ") in
   iter  pe r;
```



```
   print_string "}"
(* r(X) = {e(X) | e in r}               *)
(* val get : t -> variable -> Cvalues.t *)
let get r x =
  let f e s = Cvalues.add (Env.get e x) s in
    fold f r (Cvalues.bot ())
(* r[X <- i] = {e[X <- i] | e in r }                        *)
(* val set_elem : t -> variable -> Values.machine_int -> t *)
let set_elem r x i  =
  let f e s = add (let e' = (Env.copy e) in Env.set e' x i; e') s in
    fold f r empty
(* r[X <- v] = {e[X <- i] | e in r /\ i in v}    *)
(* val set : t -> variable -> Cvalues.t -> t *)
let set r x v =
    let f i s = union (set_elem r x i) s in
      Cvalues.fold f v empty
(* f_ASSIGN x f r =  {e[x <- i] | e in r /\ i in f({e}) cap I } *)
let f_ASSIGN x f r =
```



```
 let assign e s =
   let a i s' =
     match i with
     | ERROR_NAT _ -> s'
     | NAT _ -> add (let e' = (Env.copy e) in (Env.set e' x i; e')) s'
   in  Cvalues.fold a (f (singleton e)) s
 in fold assign r empty
(* cmp c f g r =                                              *)
(*   {e in r | exists v1 in f({e}) cap I: exists v2 in g({e}) *)
(*             cap I: v1 c v2 }                               *)
(* val cmp : (elt -> elt -> Values.machine_bool) -> (t -> t)  *)
(*                                    -> (t -> t) -> t -> t *)
exception Found
let cmp c f g r =
  let isFound i j =
    match (c i j) with
    | ERROR_BOOL _  -> ()
    | BOOLEAN false -> ()
```



```
    | BOOLEAN true  -> raise Found
  in let ok e =
    let s1 = (f (singleton e))  and s2 = (g (singleton e))
      in (try
        let tests2 j =
          (let tests1 i = isFound i j in Cvalues.iter tests1 s1)
        in Cvalues.iter tests2 s2;
           false
        with Found -> true)
    in filter ok r
(* f_EQ f g r =                                               *)
(*   {e in R | exists v1 in f({e}) cap I: exists v2 in g({e}) *)
(*                                       cap I: v1 = v2 } *)
let f_EQ f g r = cmp machine_eq f g r
(* f_LT f g r =                                               *)
(*   {e in R | exists v1 in f({e}) cap I: exists v2 in g({e}) *)
(*                                       cap I: v1 < v2 } *)
let f_LT f g r = cmp machine_lt f g r
```

Implementation: Forward collecting semantics of arithmetic expressions

```
(* caexp.mli *)
open Abstract_Syntax
open Cvalues
open Cenv
(* evaluation of arithmetic operations *)
val c_aexp : aexp -> Cenv.t -> Cvalues.t
```



```
(* caexp.ml *)
open Abstract_Syntax
(* evaluation of arithmetic operations *)
let rec c_aexp a r = match a with
  | (Abstract_Syntax.NAT i) -> (Cvalues.f_NAT i)
  | (VAR v)       -> (Cenv.get r v)
  | RANDOM        -> Cvalues.f_RANDOM ()
  | (UPLUS a1)    -> (Cvalues.f_UPLUS (c_aexp a1 r))
  | (UMINUS a1)   -> (Cvalues.f_UMINUS (c_aexp a1 r))
  | (PLUS (a1, a2))  -> (Cvalues.f_PLUS (c_aexp a1 r) (c_aexp a2 r))
  | (MINUS (a1, a2)) -> (Cvalues.f_MINUS (c_aexp a1 r) (c_aexp a2 r))
  | (TIMES (a1, a2)) -> (Cvalues.f_TIMES (c_aexp a1 r) (c_aexp a2 r))
  | (DIV (a1, a2))   -> (Cvalues.f_DIV (c_aexp a1 r) (c_aexp a2 r))
  | (MOD (a1, a2))   -> (Cvalues.f_MOD (c_aexp a1 r) (c_aexp a2 r))
```


Implementation: Forward collecting semantics of boolean expressions

```
(* cbexp.mli *)
open Abstract_Syntax
open Cvalues
open Cenv
(* evaluation of boolean operations *)
val c_bexp : bexp -> Cenv.t -> Cenv.t
```



```
(* cbexp.ml *)
open Abstract_Syntax
open Cvalues
open Cenv
open Caexp
(* evaluation of boolean operations *)
let rec c_bexp b r =
  match b with
  | TRUE           -> r
  | FALSE          -> (Cenv.bot ())
  | (EQ (a1, a2))  -> f_EQ (c_aexp a1) (c_aexp a2) r
  | (LT (a1, a2))  -> f_LT (c_aexp a1) (c_aexp a2) r
  | (AND (b1, b2)) -> Cenv.meet (c_bexp b1 r) (c_bexp b2 r)
  | (OR (b1, b2))  -> Cenv.join (c_bexp b1 r) (c_bexp b2 r)
```

We have made an oversimplification in the last alternative ignoring the case when `b1` holds and `b2` yields an error.

Implementation: fixpoints

```
(* fixpoint.mli *)
open Cenv
(* lfp x c f : iterative computation of the c-least fixpoint of f *)
(* c-greater than or equal to the prefixpoint x (f(x) >= x)       *)
val lfp : t -> (t -> t -> bool) -> (t -> t) -> t
```



```
(* fixpoint.ml *)
open Cenv
(* lfp x c f : iterative computation of the c-least fixpoint of f *)
(* c-greater than or equal to the prefixpoint x (f(x) >= x)       *)
(* x0 = x; ...; xn+1 = xn U f(xn);... ; xl where xl c xl U f(xl)  *)
let rec lfp x c f =
  let x' = (join x (f (copy x))) in
    if (c x' x) then x'
      else lfp x' c f
```


Note on the iterates: if the random choice picks 3 (may be) different values at each choice then f may not be monotonic. It follows that the iteration sequence

- $x^0 = x$
- $x^{n+1} = f(x^n)$ if $f(x^n) \not\sqsubseteq x^n$
- $x^{n+1} = x_n$ if $f(x^n) = x^n$

may not converge, even if $x \sqsubseteq f(x)$. So we compute instead

- $x^{n+1} = \bigcup_{k \leq n} f(x^k)$ if $f(x^n) \not\sqsubseteq x^n$

which is extensive and ultimately convergent to $\bigcup_{n\in\mathbb{N}} f(x^n)$. The two iterations are the same when initially x is a prefixpoint and f is monotonic. Convergence is assumed to follow from other considerations (otherwise the computation may not terminate properly).

To trace the fixpoint iterates:

```
(* fixpoint.ml *)
open Cenv
(* lfp x c f : iterative computation of the c-least fixpoint of f *)
(* c-greater than or equal to the prefixpoint x (f(x) >= x)       *)
(* x0 = x; ...; xn+1 = xn U f(xn);... ; xl where xl c xl U f(xl)  *)
let lfp x c f =
  let rec iterate n x =
     print_string "iterate ";print_int n;print_string " = ";print x;
     print_newline ();
     let x' = (join x (f (copy x))) in
        (if (c x' x) then (print_string "fixpoint = ";print x';
                            print_newline ();x')
          else iterate (n + 1) x')
   in iterate 0 x
```

Implementation: forward reachability collecting semantics of commands

```
(* ccom.mli *)
open Abstract_Syntax
open Labels
open Cenv
(* forward collecting semantics of commands *)
val ccom : com -> Cenv.t -> label -> Cenv.t

(* ccom.ml *)
open Abstract_Syntax
open Labels
open Cenv
open Caexp
open Cbexp
open Fixpoint
```



```
(* collecting semantics of commands *)

exception Error of string
let rec ccom c r l =
  match c with
  | (SKIP (l', l'')) ->
      if (l = l') then r
      else if (l = l'') then r
      else (raise (Error "SKIP incoherence"))
  | (ASSIGN (l',x,a,l'')) ->
      if (l = l') then r
      else if (l = l'') then
           f_ASSIGN x (c_aexp a) r
      else (raise (Error "ASSIGN incoherence"))
  | (SEQ (l', s, l'')) ->
      (ccomseq s r l)
  | (IF (l', b, nb, t, f, l'')) ->
      (if (l = l') then r
```



```
       else if (incom l t) then
          (ccom t (c_bexp b r) l)
       else if (incom l f) then
          (ccom f (c_bexp nb r) l)
       else if (l = l'') then
          (join (ccom t (c_bexp b r) (after t))
                (ccom f (c_bexp nb r) (after f)))
       else (raise (Error "IF incoherence")))
  | (WHILE (l', b, nb, c', l'')) ->
     let f x = join r (ccom c' (c_bexp b x) (after c'))
     in let i = lfp (bot ()) leq f in
       (if (l = l') then i
        else if (incom l c') then (ccom c' (c_bexp b i) l)
        else if (l = l'') then (c_bexp nb i)
        else (raise (Error "WHILE incoherence")))
and ccomseq s r l = match s with
  | []   -> raise (Error "empty SEQ incoherence")
  | [c]  -> if (incom l c) then (ccom c r l)
```



```
            else (raise (Error "SEQ incoherence"))
  | h::t -> if (incom l h) then (ccom h r l)
            else (ccomseq t (ccom h r (after h)) l)
```

Implementation: forward reachability collecting interpretor

```
(* main.ml *)
open Program_To_Abstract_Syntax
open Labels
open Pretty_Print
open Cenv
open Ccom
let _ =
  let arg = if (Array.length  Sys.argv) = 1 then ""
            else Sys.argv.(1) in
     Random.self_init ();
    let p = (abstract_syntax_of_program arg) in
      (print (initerr ());
       pretty_print p;
       print (ccom p (initerr ()) (after p));
       print_newline ())
```


Implementation: makefile

```
# makefile

SOURCES = \
symbol_Table.mli \
symbol_Table.ml \
variables.mli \
variables.ml \
abstract_Syntax.ml \
concrete_To_Abstract_Syntax.mli \
concrete_To_Abstract_Syntax.ml \
labels.mli \
labels.ml \
parser.mli \
parser.ml \
lexer.ml \
```



```
program_To_Abstract_Syntax.mli \
program_To_Abstract_Syntax.ml \
pretty_Print.mli \
pretty_Print.ml \
values.mli \
values.ml \
cvalues.mli \
cvalues.ml \
env.mli \
env.ml \
cenv.mli \
cenv.ml \
caexp.mli \
caexp.ml \
cbexp.mli \
cbexp.ml \
fixpoint.mli \
fixpoint.ml \
```



```
ccom.mli \
ccom.ml \
main.ml

.PHONY : help
help :
   @echo ""
   @echo "make help          : this help"
   @echo "make trace         : trace fixpoint iterates"
   @echo "make untrace       : don't trace fixpoint iterates"
   @echo "make compile       : compile"
   @echo "./a.out filename : execute"
   @echo "make examples      : execute the examples"
   @echo "make errors        : execute the examples with runtime errors"
   @echo "make clean         : remove auxilairy files"
   @echo ""

.PHONY : trace preparetrace
```

```
trace: preparetrace compile
   @echo "fixpoint tracing mode"
preparetrace:
   @/bin/rm -f fixpoint.ml
   @ln -s fixpoint_printing_iterates.ml fixpoint.ml

.PHONY : untrace prepareuntrace
untrace: prepareuntrace compile
   @echo "no fixpoint tracing, recompile!"
prepareuntrace:
   @/bin/rm -f fixpoint.ml
   @ln -s fixpoint_no_printing.ml fixpoint.ml

.PHONY : compile
compile:
   ocamlyacc parser.mly
   ocamllex lexer.mll
#   ocamlc -i $(SOURCES) # to print types
```



```
   ocamlc $(SOURCES)

.PHONY : examples
examples :
   ./a.out ../Examples/example00.sil
   ./a.out ../Examples/example01.sil
   ./a.out ../Examples/example02.sil
   ./a.out ../Examples/example03.sil
   ./a.out ../Examples/example04.sil
   ./a.out ../Examples/example05.sil
   ./a.out ../Examples/example07.sil

.PHONY : errors
errors :
   ./a.out ../Examples/example06.sil
   ./a.out ../Examples/example08.sil
   ./a.out ../Examples/example09.sil
   ./a.out ../Examples/example10.sil
```



```
   ./a.out ../Examples/example11.sil

.PHONY :
clean :
   /bin/rm -f *.cmi *.cmo *~ a.out lexer.ml parser.mli parser.ml
```


Implementation: Examples

```
Script started on Mon Apr  4 15:22:39 2005
% make clean
...
% make untrace
...
% make compile
...
% make examples

./a.out ../Examples/example0.sil
{ [ ] }
:
  skip
:

```

```
{ [ ] }
./a.out ../Examples/example1.sil
{ [ x = _O_(i); ] }
:
  x := 1;
:
  while (x < 100) do
    2:
      x := (x + 1)
    3:
  od {((100 < x) | (x = 100))}
:

{ [ x = 100; ] }
./a.out ../Examples/example2.sil
{ [ x = _O_(i); y = _O_(i); ] }
:
  x := (-1073741823 - 1);
```



```
:
  y := (x - 1)
:

{ }
./a.out ../Examples/example3.sil
{ [ x = _O_(i); y = _O_(i); ] }
:
  x := 0;
:
  y := 1
:

{ [ x = 0; y = 1; ] }
./a.out ../Examples/example4.sil
{ [ x = _O_(i); ] }
```



```
:
  if true then
    1:
      x := 1
    2:
  else {false}
    3:
      x := 0
    4:
  fi
:

{ [ x = 1; ] }
./a.out ../Examples/example5.sil
{ [ x = _O_(i); ] }
:
  if false then
    1:
```



```
      x := 1
    2:
  else {true}
    3:
      x := 0
    4:
  fi
:

{ [ x = 0; ] }
./a.out ../Examples/example7.sil
{ [ x = _O_(i); ] }
:
  x := 1;
:
  while ((x < 10) | (x = 10)) do
    2:
      x := (x + 1)
```

```
    3:
  od {(10 < x)}
:

{ [ x = 11; ] }
% ^Dexit

Script done on Mon Apr  4 15:23:08 2005
```


Implementation: Examples of runtime errors than stop execution

```
Script started on Mon Apr  4 15:23:25 2005
% make errors

./a.out ../Examples/example6.sil
{ [ x = _O_(i); ] }
:
  x := -1073741824
:

{ }
./a.out ../Examples/example8.sil
{ [ x = _O_(i); ] }
:
```



```
  x := 1073741823
:

{ [ x = 1073741823; ] }
./a.out ../Examples/example9.sil
{ [ x = _O_(i); y = _O_(i); z = _O_(i); t = _O_(i); ] }
:
  x := (-536870912 * 2);
:
  y := (536870912 * 2);
:
  z := ((-1073741823 - 1) * 1);
:
  t := ((-1073741823 - 1) * 1073741823)
:

{ }
./a.out ../Examples/example10.sil
```



```
{ [ x = _O_(i); ] }
:
  x := ?;
:
  if (x < (-1073741823 - 1)) then
    2:
      x := 1
    3:
  else {(((-1073741823 - 1) < x) | (x = (-1073741823 - 1)))}
    4:
      x := 0
    5:
  fi
:

{ [ x = 0; ] }
./a.out ../Examples/example11.sil
{ [ x = _O_(i); ] }
```

```
:
  x := 1;
:
  while (0 < 1073741824) do
    2:
      x := (x + 1)
    3:
  od {((1073741824 < 0) | (1073741824 = 0))}
:

{ }
% ^Dexit

Script done on Mon Apr  4 15:23:36 2005
```


Conclusion

– We have examplified the calculational design of program static analyzers by abstract interpretation of a formal semantics;
– This provides a thorough understanding of the abstraction process allowing for the later development of useful large scale analyzers;
– Scales up manually by small parts;
– One can hope that in the future one can extract analyzers from correctness proofs;

– Since such extracted will probably be inefficient, one can consider:
  - The use of manually constructed static analyzers to compute inductive fixpoint approximations (which involve iterations with convergence acceleration)
  - The use of static analyzers extracted from the correctness proof to check that the previous fixpoint approximations are indeed inductive (which involves no iteration)

Bibliography

[1] Patrick Cousot.
The Calculational Design of a Generic Abstract Interpreter. In *Calculational System Design*, M. Broy & R. Steinbrüggen, editors, NATO ASI Series F. IOS Press, Amsterdam, 1999. http://www.dmi.ens.fr/~cousot/COUSOTpapers/Marktoberdorf98.shtml, November 1998.

[2] Patrick Cousot.
The Marktoberdorf'98 generic abstract interpreter. http://www.dmi.ens.fr/~cousot/Marktoberdorf98.shtml, November 1998.

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/.