New York University, CIMS, CS, Course CSCI-GA.3140-001, Spring 2024

"Abstract Interpretation"

Ch. 4, Syntax

Patrick Cousot

pcousot@cs.nyu.edu   cs.nyu.edu/~pcousot

Class 2, Monday, January 29th 2024, 4:55-6:55 PM, WWH, Room CIWW 202

These slides are available at
http://cs.nyu.edu/~pcousot/courses/spring24/CSCI-GA.3140-001/slides/02--2024-01-29-syntax-semantics-traces-oo/slides-04--syntax-AI.pdf

What did we learned in
Ch. 3, Syntax, Semantics, Properties, and Static Analysis of Expressions?

- To design a program analysis (e.g. signs of expressions):

(1) Define the syntax of programs (e.g. expressions)

(2) Define the semantics of programs (e.g. what expressions compute)

(3) The collecting semantics of programs is the strongest property of the program semantics

(4) Define the abstraction of semantic program properties by a Galois connection (e.g. sign abstraction of value, environment, and semantic properties)

(5) Design the abstract semantics by calculational design of an abstraction of the collecting semantics

$\Rightarrow$ the abstract semantics specifies a static analyzer which is sound by construction

- These slides consider (1) for a mini-language (a subset of C).

Objective

The objective of this chapter 4, “Syntax” is to introduce the syntax of the small subset of the C programming language used throughout the book to exemplify abstract interpretation

en.wikipedia.org/wiki/C_(programming_language)
en.wikipedia.org/wiki/Syntax_(programming_languages)

Chapter 4

Ch. 4, Syntax

Syntax of programs

Syntax of statements

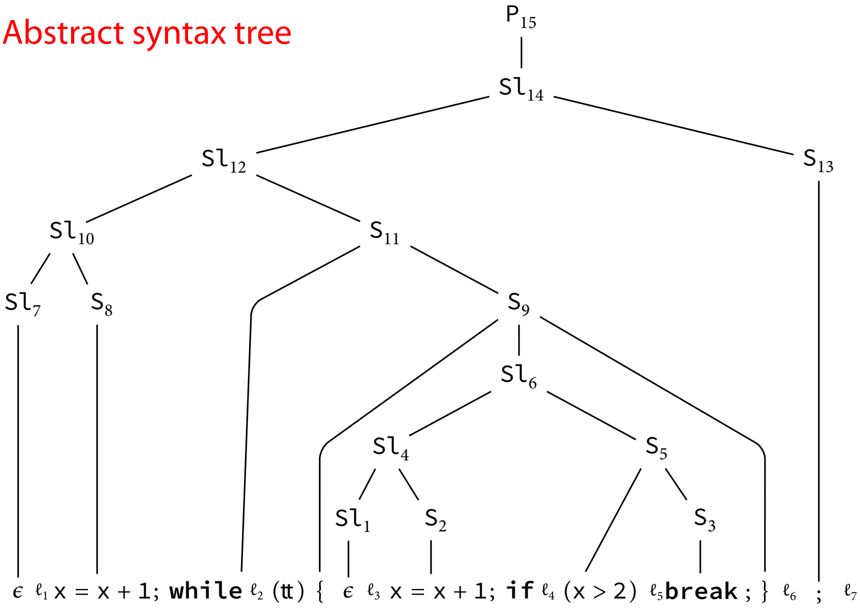
- A subset of C

$$
\begin{array}{lcll}
\mathtt{S} & ::= & & \text{statement } \mathtt{S} \in \mathbb{S} \\
& & \mathtt{x} = \mathtt{A}\ ; & \text{assignment} \\
& | & ; & \text{skip} \\
& | & \mathbf{if}\ (\mathtt{B})\ \mathtt{S} & \text{conditional} \\
& | & \mathbf{if}\ (\mathtt{B})\ \mathtt{S}\ \mathbf{else}\ \mathtt{S} & \\
& | & \mathbf{while}\ (\mathtt{B})\ \mathtt{S} & \text{iteration} \\
& | & \mathbf{break}\ ; & \text{iteration break (to inner enclosing loop exit)} \\
& | & \{\ \mathtt{Sl}\ \} & \text{compound statement} \\
\mathtt{Sl} & ::= & \mathtt{Sl}\ \mathtt{S} \mid \epsilon & \text{statement list } \mathtt{Sl} \in \mathbb{Sl}
\end{array}
$$

- Example:

$$
\mathtt{x} = \mathtt{x} + 1\ ; \mathbf{while}\ (\mathtt{tt})\ \{\ \mathtt{x} = \mathtt{x} + 1\ ; \mathbf{if}\ (\mathtt{x} > 2)\ \mathbf{break}\ ;\ \}; \tag{4.1}
$$

Abstract syntax tree



en.wikipedia.org/wiki/Abstract_syntax en.wikipedia.org/wiki/Abstract_syntax_tree

Syntax of programs

$$\mathtt{P} \quad ::= \quad \mathtt{Sl} \qquad \text{program } \mathtt{P} \in \mathbb{P}$$

$$\mathbb{P}c \quad \triangleq \quad \mathbb{S} \cup \mathbb{S}\mathbb{l} \cup \mathbb{P} \qquad \text{program component } \mathtt{S} \in \mathbb{P}c$$

Labels of programs

Labels

- Labels are unique and designate program points of statements
- Labels are not part of the program syntax, their syntax is free
- Example: labelling of program (4.1):

$$\begin{array}{l} \ell_1\ \mathtt{x} = \mathtt{x} + \mathtt{1}\ ; \\ \quad \mathbf{\mathtt{while}}\ \ell_2\ (\mathtt{tt})\ \{ \\ \qquad \ell_3\ \mathtt{x} = \mathtt{x} + \mathtt{1}\ ; \\ \qquad \mathbf{\mathtt{if}}\ \ell_4\ (\mathtt{x} > \mathtt{2})\ \ell_5\ \mathbf{\mathtt{break}}\ ;\}\ell_6;\ell_7 \end{array} \tag{4.5}$$

break ; statements are used to exit iteration statements only (so the **break** ; at ℓ_5 exits to ℓ_6).

en.wikipedia.org/wiki/Control_flow#Early_exit_from_loops
en.wikipedia.org/wiki/Label_(computer_science)

Labels

$\mathrm{at}[\![S]\!]$	the program point at which execution of program component S starts;
$\mathrm{after}[\![S]\!]$	the program exit point after program component S, at which execution of S is supposed to normally terminate, if ever;
$\mathrm{escape}[\![S]\!]$	a boolean indicating whether or not the program component S contains a **`break ;`** statement escaping out of that component S;
$\mathrm{break\text{-}to}[\![S]\!]$	the program point at which execution of the program component S goes to when a **`break ;`** statement escapes out of that component S;
$\mathrm{breaks\text{-}of}[\![S]\!]$	the set of labels of all **`break ;`** statements that can escape out of S
$\mathrm{in}[\![S]\!]$	the set of program points inside program component S (including $\mathrm{at}[\![S]\!]$ but excluding $\mathrm{after}[\![S]\!]$ and $\mathrm{break\text{-}to}[\![S]\!]$);
$\mathrm{labx}[\![S]\!]$	the potentially reachable program points while executing program component S either at, in, or after the program component, or resulting from a break.

Label at a statement S

- $\mathsf{at}[\![\mathtt{S}]\!]$: the program point at which execution of S starts;

$$
\begin{array}{rcll}
\mathtt{P} & ::= & \mathtt{Sl} & \mathsf{at}[\![\mathtt{P}]\!] \triangleq \mathsf{at}[\![\mathtt{Sl}]\!] \\
\mathtt{Sl} & ::= & \mathtt{Sl}'\ \mathtt{S} & \mathsf{at}[\![\mathtt{Sl}]\!] \triangleq \mathsf{at}[\![\mathtt{Sl}']\!] \\
\mathtt{Sl} & ::= & \epsilon & \mathsf{at}[\![\mathtt{Sl}]\!] \triangleq \mathsf{after}[\![\mathtt{Sl}]\!] \\
\mathtt{S} & ::= & \{\ \mathtt{Sl}\ \} & \mathsf{at}[\![\mathtt{S}]\!] \triangleq \mathsf{at}[\![\mathtt{Sl}]\!]
\end{array}
$$

Label after a statement S

- after⟦S⟧: the program point at which execution will continue upon termination of the statement S (unless there is a **break** ;);

P	::=	Sl	$\mathsf{after}[\![\mathtt{P}]\!] \triangleq \mathsf{after}[\![\mathtt{Sl}]\!]$
Sl	::=	Sl′ S	$\mathsf{after}[\![\mathtt{Sl'}]\!] \triangleq \mathsf{at}[\![\mathtt{S}]\!]$, $\quad \mathsf{after}[\![\mathtt{S}]\!] \triangleq \mathsf{after}[\![\mathtt{Sl}]\!]$
S	::=	**if** (B) $\mathtt{S}_t$	$\mathsf{after}[\![\mathtt{S}_t]\!] \triangleq \mathsf{after}[\![\mathtt{S}]\!]$
S	::=	**if** (B) $\mathtt{S}_t$ **else** $\mathtt{S}_f$	$\mathsf{after}[\![\mathtt{S}_t]\!] \triangleq \mathsf{after}[\![\mathtt{S}_f]\!] \triangleq \mathsf{after}[\![\mathtt{S}]\!]$
S	::=	**while** (B) $\mathtt{S}_b$	$\mathsf{after}[\![\mathtt{S}_b]\!] \triangleq \mathsf{at}[\![\mathtt{S}]\!]$
S	::=	{ Sl }	$\mathsf{after}[\![\mathtt{Sl}]\!] \triangleq \mathsf{after}[\![\mathtt{S}]\!]$

Explicit labelling

- When explicitly decorating programs with labels, we should have

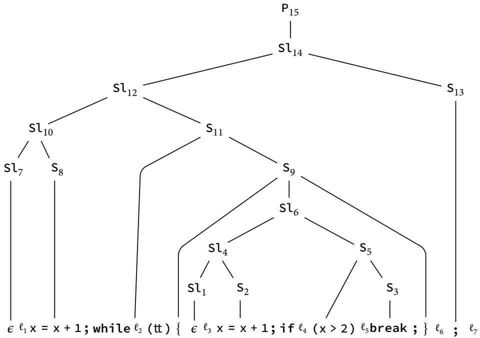
$$
\begin{array}{lcll}
\mathtt{S} & ::= & \ell\ \mathtt{x} = \mathtt{A}\ ; & \mathsf{at}[\![\mathtt{S}]\!] \triangleq \ell \\
\mathtt{S} & ::= & \ell ; & \mathsf{at}[\![\mathtt{S}]\!] \triangleq \ell \\
\mathtt{S} & ::= & \mathbf{if}\ \ell\ (\mathtt{B})\ \mathtt{S}_t & \mathsf{at}[\![\mathtt{S}]\!] \triangleq \ell \\
\mathtt{S} & ::= & \mathbf{if}\ \ell\ (\mathtt{B})\ \mathtt{S}_t\ \mathbf{else}\ \mathtt{S}_f & \mathsf{at}[\![\mathtt{S}]\!] \triangleq \ell \\
\mathtt{S} & ::= & \mathbf{while}\ \ell\ (\mathtt{B})\ \mathtt{S}_b & \mathsf{at}[\![\mathtt{S}]\!] \triangleq \ell \\
\mathtt{S} & ::= & \ell\ \mathbf{break}\ ; & \mathsf{at}[\![\mathtt{S}]\!] \triangleq \ell \\
\mathtt{P} & ::= & \mathtt{Sl}\ \ell & \mathsf{after}[\![\mathtt{P}]\!] \triangleq \mathsf{after}[\![\mathtt{Sl}]\!] \triangleq \ell
\end{array}
$$

- "In ℓ_1 $\mathtt{x} = \mathtt{A}\ ;\ell_2$, execution starts at ℓ_1 and continues at ℓ_2" is a shorthand for 'In $\mathtt{x} = \mathtt{A}\ ;$, execution starts at $\mathsf{at}[\![\mathtt{x} = \mathtt{A}\ ;]\!]$ and continues at $\mathsf{after}[\![\mathtt{x} = \mathtt{A}\ ;]\!]$"

- The program labelling is arbitrary and can change and be generated automatically

Another choice for labels

- A label can be a path in the abstract syntax tree
- Example:



$$\ell_3 \quad = \quad \mathtt{P}_{15}\,\mathtt{Sl}_{14}\,\mathtt{Sl}_{12}\,\mathtt{S}_{11}\,\mathtt{S}_{9}\,\mathtt{Sl}_{6}\,\mathtt{Sl}_{4}\,\mathtt{S}_{2}$$

Escaping a statement S

- $\mathsf{escape}[\![\mathsf{S}]\!]$: true (tt) if and only if the statement S contains a **break** ; that can escape out of that S;

$$
\begin{array}{rcllr}
\mathsf{P} & ::= & \mathtt{Sl} & \mathsf{escape}[\![\mathsf{P}]\!] \triangleq \mathsf{escape}[\![\mathtt{Sl}]\!] & \mathsf{escape}[\![\mathsf{P}]\!] = \mathit{ff} \\
\mathtt{Sl} & ::= & \mathtt{Sl}'\ \mathsf{S} & \mathsf{escape}[\![\mathtt{Sl}]\!] \triangleq \mathsf{escape}[\![\mathtt{Sl}']\!] \vee \mathsf{escape}[\![\mathsf{S}]\!] & \\
\mathtt{Sl} & ::= & \epsilon & \mathsf{escape}[\![\mathtt{Sl}]\!] \triangleq \mathit{ff} & \\
\mathsf{S} & ::= & \mathsf{x} = \mathsf{A}\ ; & \mathsf{escape}[\![\mathsf{S}]\!] \triangleq \mathit{ff} & \\
\mathsf{S} & ::= & ; & \mathsf{escape}[\![\mathsf{S}]\!] \triangleq \mathit{ff} & \\
\mathsf{S} & ::= & \mathtt{if}\ (\mathsf{B})\ \mathsf{S}_t & \mathsf{escape}[\![\mathsf{S}]\!] \triangleq \mathsf{escape}[\![\mathsf{S}_t]\!] & \\
\mathsf{S} & ::= & \mathtt{if}\ (\mathsf{B})\ \mathsf{S}_t\ \mathtt{else}\ \mathsf{S}_f & \mathsf{escape}[\![\mathsf{S}]\!] \triangleq \mathsf{escape}[\![\mathsf{S}_t]\!] \vee \mathsf{escape}[\![\mathsf{S}_f]\!] & \\
\mathsf{S} & ::= & \mathtt{while}\ (\mathsf{B})\ \mathsf{S}_b & \mathsf{escape}[\![\mathsf{S}]\!] \triangleq \mathit{ff} & \\
\mathsf{S} & ::= & \mathtt{break}\ ; & \mathsf{escape}[\![\mathsf{S}]\!] \triangleq \mathit{tt} & \\
\mathsf{S} & ::= & \{\ \mathtt{Sl}\ \} & \mathsf{escape}[\![\mathsf{S}]\!] \triangleq \mathsf{escape}[\![\mathtt{Sl}]\!] &
\end{array}
$$

Break label of a statement S

- break-to⟦S⟧: the program point at which execution of S goes to when a **break ;** statement is executed while executing S;

$$
\begin{array}{rcl@{\qquad}l}
\mathtt{Sl} & ::= & \mathtt{Sl}'\ \mathtt{S} & \text{break-to}[\![\mathtt{Sl}']\!] \triangleq \text{break-to}[\![\mathtt{S}]\!] \triangleq \text{break-to}[\![\mathtt{Sl}]\!] \\
\mathtt{S} & ::= & \mathbf{if}\ (\mathtt{B})\ \mathtt{S}_t & \text{break-to}[\![\mathtt{S}_t]\!] \triangleq \text{break-to}[\![\mathtt{S}]\!] \\
\mathtt{S} & ::= & \mathbf{if}\ (\mathtt{B})\ \mathtt{S}_t\ \mathbf{else}\ \mathtt{S}_f & \text{break-to}[\![\mathtt{S}_t]\!] \triangleq \text{break-to}[\![\mathtt{S}_f]\!] \triangleq \text{break-to}[\![\mathtt{S}]\!] \\
\mathtt{S} & ::= & \mathbf{while}\ (\mathtt{B})\ \mathtt{S}_b & \text{break-to}[\![\mathtt{S}_b]\!] \triangleq \text{after}[\![\mathtt{S}]\!] \\
\mathtt{S} & ::= & \{\ \mathtt{Sl}\ \} & \text{break-to}[\![\mathtt{Sl}]\!] \triangleq \text{break-to}[\![\mathtt{S}]\!]
\end{array}
$$

Break label of a statement S

- $\text{breaks-of}[\![\mathsf{S}]\!]$: collects the labels of all **break** ; statements that can escape out of S (so excluding **break** ; statements inside an iteration statement within S).
- The definition checks that **break** ; statements can only appear within loops;

$$
\begin{array}{lcll}
\mathsf{P} & ::= & \mathtt{Sl} & \text{breaks-of}[\![\mathsf{P}]\!] \triangleq \text{breaks-of}[\![\mathtt{Sl}]\!] \qquad \text{breaks-of}[\![\mathsf{P}]\!] = \varnothing \\
\mathtt{Sl} & ::= & \mathtt{Sl}'\ \mathsf{S} & \text{breaks-of}[\![\mathtt{Sl}]\!] \triangleq \text{breaks-of}[\![\mathtt{Sl}']\!] \cup \text{breaks-of}[\![\mathsf{S}]\!] \\
\mathtt{Sl} & ::= & \epsilon & \text{breaks-of}[\![\mathtt{Sl}]\!] \triangleq \varnothing \\
\mathsf{S} & ::= & \mathsf{x} = \mathsf{A}\ ; & \text{breaks-of}[\![\mathsf{S}]\!] \triangleq \varnothing \\
\mathsf{S} & ::= & ; & \text{breaks-of}[\![\mathsf{S}]\!] \triangleq \varnothing \\
\mathsf{S} & ::= & \mathbf{if}\ (\mathsf{B})\ \mathsf{S}_t & \text{breaks-of}[\![\mathsf{S}]\!] \triangleq \text{breaks-of}[\![\mathsf{S}_t]\!] \\
\mathsf{S} & ::= & \mathbf{if}\ (\mathsf{B})\ \mathsf{S}_t\ \mathbf{else}\ \mathsf{S}_f & \text{breaks-of}[\![\mathsf{S}]\!] \triangleq \text{breaks-of}[\![\mathsf{S}_t]\!] \cup \text{breaks-of}[\![\mathsf{S}_f]\!] \\
\mathsf{S} & ::= & \mathbf{while}\ (\mathsf{B})\ \mathsf{S}_b & \text{breaks-of}[\![\mathsf{S}]\!] \triangleq \varnothing \\
\mathsf{S} & ::= & \ell\ \mathbf{break}\ ; & \text{breaks-of}[\![\mathsf{S}]\!] \triangleq \{\text{at}[\![\mathsf{S}]\!]\} \\
\mathsf{S} & ::= & \{\ \mathtt{Sl}\ \} & \text{breaks-of}[\![\mathsf{S}]\!] \triangleq \text{breaks-of}[\![\mathtt{Sl}]\!]
\end{array}
$$

Labels in a statement S

- in⟦S⟧: the program points inside S (including at⟦S⟧ but excluding after⟦S⟧ and break-to⟦S⟧)
- The definition checks that program labels are unique.

P	::=	$\mathtt{Sl}$	$\mathrm{in}[\![\mathtt{P}]\!] \triangleq \mathrm{in}[\![\mathtt{Sl}]\!]$	$\mathrm{after}[\![\mathtt{Sl}]\!] \notin \mathrm{in}[\![\mathtt{Sl}]\!]$
Sl	::=	$\mathtt{Sl}'\ \mathtt{S}$	$\mathrm{in}[\![\mathtt{Sl}]\!] \triangleq \mathrm{in}[\![\mathtt{Sl}']\!] \cup \mathrm{in}[\![\mathtt{S}]\!]$	$\mathrm{in}[\![\mathtt{Sl}']\!] \cap \mathrm{in}[\![\mathtt{S}]\!] = \varnothing$ when $\mathtt{Sl}' \neq \{\ \ldots\{\ \epsilon\ \}\ldots\ \}$
Sl	::=	ϵ	$\mathrm{in}[\![\mathtt{Sl}]\!] \triangleq \{\mathrm{at}[\![\mathtt{Sl}]\!]\}$	
S	::=	$\mathtt{x} = \mathtt{A}\ ;$	$\mathrm{in}[\![\mathtt{S}]\!] \triangleq \{\mathrm{at}[\![\mathtt{S}]\!]\}$	
S	::=	;	$\mathrm{in}[\![\mathtt{S}]\!] \triangleq \{\mathrm{at}[\![\mathtt{S}]\!]\}$	
S	::=	$\mathtt{if}\ (\mathtt{B})\ \mathtt{S}_t$	$\mathrm{in}[\![\mathtt{S}]\!] \triangleq \{\mathrm{at}[\![\mathtt{S}]\!]\} \cup \mathrm{in}[\![\mathtt{S}_t]\!]$	$\mathrm{at}[\![\mathtt{S}]\!] \notin \mathrm{in}[\![\mathtt{S}_t]\!]$
S	::=	$\mathtt{if}\ (\mathtt{B})\ \mathtt{S}_t\ \mathtt{else}\ \mathtt{S}_f$	$\mathrm{in}[\![\mathtt{S}]\!] \triangleq \{\mathrm{at}[\![\mathtt{S}]\!]\} \cup \mathrm{in}[\![\mathtt{S}_t]\!] \cup \mathrm{in}[\![\mathtt{S}_f]\!]$	$\mathrm{at}[\![\mathtt{S}]\!] \notin \mathrm{in}[\![\mathtt{S}_t]\!] \cup \mathrm{in}[\![\mathtt{S}_f]\!]$ $\mathrm{in}[\![\mathtt{S}_t]\!] \cap \mathrm{in}[\![\mathtt{S}_f]\!] = \varnothing$
S	::=	$\mathtt{while}\ (\mathtt{B})\ \mathtt{S}_b$	$\mathrm{in}[\![\mathtt{S}]\!] \triangleq \{\mathrm{at}[\![\mathtt{S}]\!]\} \cup \mathrm{in}[\![\mathtt{S}_b]\!]$	$\mathrm{at}[\![\mathtt{S}]\!] \notin \mathrm{in}[\![\mathtt{S}_b]\!]$
S	::=	$\mathtt{break}\ ;$	$\mathrm{in}[\![\mathtt{S}]\!] \triangleq \{\mathrm{at}[\![\mathtt{S}]\!]\}$	
S	::=	$\{\ \mathtt{Sl}\ \}$	$\mathrm{in}[\![\mathtt{S}]\!] \triangleq \mathrm{in}[\![\mathtt{Sl}]\!]$	

Labels of a statement S

- labs$[\![S]\!]$: the potentially reachable program points while executing S either in or after the statement;
- labx$[\![S]\!]$: the potentially reachable program points while executing S either in or after the statement, or resulting from a break.

$$
\begin{array}{lcl}
\mathrm{labs}[\![S]\!] & \triangleq & \mathrm{in}[\![S]\!] \cup \{\mathrm{after}[\![S]\!]\} \\
\mathrm{labx}[\![S]\!] & \triangleq & \mathrm{labs}[\![S]\!] \cup (\!\!\lvert\, \mathrm{escape}[\![S]\!] \;?\; \{\mathrm{break\text{-}to}[\![S]\!]\} \;:\; \varnothing \,\rvert\!\!)
\end{array}
$$

Informal illustration of the definitions

$\mathtt{S}'$

$\mathsf{labx}\llbracket\mathtt{S}\rrbracket$ $\ldots$

$\mathsf{labs}\llbracket\mathtt{S}\rrbracket$

$\mathsf{in}\llbracket\mathtt{S}\rrbracket$

$\mathsf{break\text{-}to}\llbracket\mathtt{S}\rrbracket$

$\mathtt{\mathbf{while}\ (B)}\ \{\{\mathsf{at}\llbracket\mathtt{S}\rrbracket \ldots \ell_1\mathtt{\mathbf{break}}\ \mathtt{;} \ldots \ell_2\mathtt{\mathbf{break}}\ \mathtt{;} \ldots\}\ \mathsf{after}\llbracket\mathtt{S}\rrbracket\ \ldots\}\ \mathsf{after}\llbracket\mathtt{S}'\rrbracket$

$\mathsf{breaks\text{-}of}\llbracket\mathtt{S}\rrbracket$

Example of program labelling (home work)

The program P = $\texttt{while}\ \ell_1\ (\texttt{tt})\ \ell_2\ \texttt{break}\ ;\ \ell_3\ \texttt{x} = 7\ ;\ell_4$ has grammatical structure

- P
  - $\texttt{Sl}_1$
    - $\texttt{Sl}_2$
      - $\texttt{Sl}_3$
        - ϵ
      - $\texttt{S}_4$
        - $\texttt{while}\ \ell_1\ (\texttt{tt})$
        - $\texttt{S}_5$
          - $\ell_2\ \texttt{break}\ ;$
    - $\texttt{S}_6$
      - $\ell_3\ \texttt{x} = 7\ ;$
  - ℓ_4

The labelling is

S	at⟦S⟧	in⟦S⟧	escape⟦S⟧	break-to⟦S⟧	after⟦S⟧	labs⟦S⟧	labx⟦S⟧
P	ℓ_1	ℓ_1, ℓ_2, ℓ_3	ff	—	ℓ_4	$\ell_1, \ell_2, \ell_3, \ell_4$	$\ell_1, \ell_2, \ell_3, \ell_4$
$\texttt{Sl}_1$	ℓ_1	ℓ_1, ℓ_2, ℓ_3	ff	—	ℓ_4	$\ell_1, \ell_2, \ell_3, \ell_4$	$\ell_1, \ell_2, \ell_3, \ell_4$
$\texttt{Sl}_2$	ℓ_1	ℓ_1, ℓ_2	ff	—	ℓ_3	ℓ_1, ℓ_2, ℓ_3	ℓ_1, ℓ_2, ℓ_3
$\texttt{Sl}_3$	ℓ_1	ℓ_1	ff	—	ℓ_1	ℓ_1	ℓ_1
$\texttt{S}_4$	ℓ_1	ℓ_1, ℓ_2	ff	—	ℓ_3	ℓ_1, ℓ_2, ℓ_3	ℓ_1, ℓ_2, ℓ_3
$\texttt{S}_5$	ℓ_2	ℓ_2	tt	ℓ_3	ℓ_1	ℓ_1, ℓ_2	ℓ_1, ℓ_2, ℓ_3
$\texttt{S}_6$	ℓ_3	ℓ_3	ff	—	ℓ_4	ℓ_3, ℓ_4	ℓ_3, ℓ_4

Properties of the program labelling

Lemma (4.15) For all program components S of a program P, $\mathsf{at}[\![\mathsf{S}]\!] \in \mathsf{in}[\![\mathsf{S}]\!]$.

Lemma (4.16) For all program non-empty components $\mathsf{S} \neq \{ \ldots \{ \ \epsilon \ \} \ldots \}$ of a program P, $\mathsf{after}[\![\mathsf{S}]\!] \notin \mathsf{in}[\![\mathsf{S}]\!]$.

Lemma (4.17) For all program components S of a program P, $\mathsf{escape}[\![\mathsf{S}]\!] \Rightarrow (\mathsf{break\text{-}to}[\![\mathsf{S}]\!] \notin \mathsf{in}[\![\mathsf{S}]\!])$.

Lemma (4.18) For all program components S of a program P, $\mathsf{escape}[\![\mathsf{S}]\!] \Rightarrow (\mathsf{break\text{-}to}[\![\mathsf{S}]\!] \neq \mathsf{after}[\![\mathsf{S}]\!])$.

Proofs in the book.

The End, Thank you

