SAS 2019

Wednesday, October 9$^{\text{th}}$ 2019

Symposium on Formal Methods, FM'19, Porto, Portugal

Syntactic and Semantic Soundness of Structural Dataflow Analysis

Patrick Cousot

New York University, Courant Institute of Mathematics, Computer Science

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

Soundness of data flow analysis

- In what sense is data flow analysis sound?
- Classical definitions of liveness (and other data flow analyses) [Beyer, Gulwani, and Schmidt, 2018; Kildall, 1973; Schmidt, 1998; Steffen, 1991, 1993]
  - hide subtleties in the definition of soundness,
  - which may lead to incorrect semantics-based compiler optimizations.

Syntax and trace semantics of programs

Syntax

$\mathtt{x}, \mathtt{y}, \ldots \in \mathbb{V}$		variable ($\mathbb{V}$ not empty)
$\mathtt{A} \in \mathbb{A}$	$::= \mathtt{1} \mid \mathtt{x} \mid \mathtt{A}_1 - \mathtt{A}_2$	arithmetic expression
$\mathtt{B} \in \mathbb{B}$	$::= \mathtt{A}_1 < \mathtt{A}_2 \mid \mathtt{B}_1\ \mathtt{nand}\ \mathtt{B}_2$	boolean expression
$\mathtt{S} \in \mathbb{S}$	$::=$	statement
	$\mathtt{x = A ;}$	assignment
	$\mid\ \mathtt{;}$	skip
	$\mid\ \mathtt{if\ (B)\ S} \mid \mathtt{if\ (B)\ S\ else\ S}$	conditionals
	$\mid\ \mathtt{while\ (B)\ S} \mid \mathtt{break\ ;}$	iteration and break
	$\mid\ \mathtt{\{\ Sl\ \}}$	compound statement
$\mathtt{Sl} \in \mathbb{Sl}$	$::= \mathtt{Sl}\ \ \mathtt{S} \mid \epsilon$	statement list
$\mathtt{P} \in \mathbb{P}$	$::= \mathtt{Sl}$	program
$\mathtt{S} \in \mathbb{Pc}$	$\triangleq \mathbb{S} \cup \mathbb{Sl} \cup \mathbb{P}$	program component

Program labelling

Unique labelling to designate (sets of) program points:

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

Traces

- Events/action: assignment $\mathtt{x = A} = v$, true test $\mathtt{B}$, false test $\neg(\mathtt{B})$, **break**, $\mathtt{skip}$
- State: program label ℓ (next step to be executed)
- Trace: finite/infinite sequence $\pi \in \mathbb{T}^{+\infty}$ of states separated by events
- Example: $\ell_1 \xrightarrow{\mathtt{x = x + 1} = 1} \ell_2 \xrightarrow{\neg(\mathtt{x} < 0)} \ell_4$ (with implicit initialization to 0)
- Trace concatenation: $\frown$
- Value $\varrho(\pi)\mathtt{x}$ of a variable x at the end of trace π

$$
\begin{aligned}
\varrho(\ell)\mathtt{x} &\triangleq 0 && \text{implicit initialization to } 0 \\
\varrho(\pi^{\ell} \xrightarrow{\mathtt{x = A} = v} \ell')\mathtt{x} &\triangleq v \\
\varrho(\pi^{\ell} \xrightarrow{\cdots} \ell')\mathtt{x} &\triangleq \varrho(\pi^{\ell})\mathtt{x} && \text{otherwise}
\end{aligned} \tag{2}
$$

Prefix trace semantics

- Evaluation of an arithmetic expression

$$
\begin{aligned}
\mathscr{A}[\![1]\!]\rho &\triangleq 1 \\
\mathscr{A}[\![\mathtt{x}]\!]\rho &\triangleq \rho(\mathtt{x}) \\
\mathscr{A}[\![\mathtt{A}_1 - \mathtt{A}_2]\!]\rho &\triangleq \mathscr{A}[\![\mathtt{A}_1]\!]\rho - \mathscr{A}[\![\mathtt{A}_2]\!]\rho
\end{aligned} \tag{4}
$$

- Assignment $\mathtt{S} ::= {}^{\ell}\ \mathtt{x} = \mathtt{A}\ ;$ (where $\mathsf{at}[\![\mathtt{S}]\!] = \ell$)

$$
\mathcal{S}^*[\![\mathtt{S}]\!] \triangleq \{\langle \pi\ell,\ \ell\rangle, \langle \pi\ell,\ \ell \xrightarrow{\mathtt{x} = \mathtt{A} = \nu} \mathsf{after}[\![\mathtt{S}]\!]\rangle \mid \pi\ell \in \mathbb{T}^+ \wedge \nu = \mathscr{A}[\![\mathtt{A}]\!]\varrho(\pi\ell)\} \tag{3}
$$

Prefix trace semantics (cont'd)

- Break statement $\mathtt{S} ::= {}^{\ell}\ \mathtt{break}\ \mathtt{;}$ (where $\mathsf{at}[\![\mathtt{S}]\!] = \ell$)

$$\mathcal{S}^*[\![\mathtt{S}]\!] \triangleq \{\langle \pi\ell,\ \ell\rangle, \langle \pi\ell,\ \ell \xrightarrow{\mathtt{break}} \mathsf{break\text{-}to}[\![\mathtt{S}]\!]\rangle \mid \pi\ell \in \mathbb{T}^+\} \tag{5}$$

Prefix trace semantics (cont'd)

- Conditional statement $\mathtt{S} ::= \mathtt{if}\ \ell\ \mathtt{(B)}\ \mathtt{S}_t$ (where $\mathsf{at}[\![\mathtt{S}]\!] = \ell$)

$$\begin{aligned}\mathcal{S}^*[\![\mathtt{S}]\!] \triangleq\ & \{\langle\pi_1\ell,\ \ell\rangle \mid \pi_1\ell \in \mathbb{T}^+\} \\ & \cup \{\langle\pi_1\ell,\ \ell \xrightarrow{\neg(\mathtt{B})} \mathsf{after}[\![\mathtt{S}]\!]\rangle \mid \mathscr{B}[\![\mathtt{B}]\!]\varrho(\pi_1\ell) = \mathit{ff} \wedge \pi_1\ell \in \mathbb{T}^+\} \\ & \cup \{\langle\pi_1\ell,\ \ell \xrightarrow{\mathtt{B}} \mathsf{at}[\![\mathtt{S}_t]\!] \frown \pi_2\rangle \mid \mathscr{B}[\![\mathtt{B}]\!]\varrho(\pi_1\ell) = \mathit{tt} \wedge \langle\pi_1\ell \xrightarrow{\mathtt{B}} \mathsf{at}[\![\mathtt{S}_t]\!],\ \pi_2\rangle \in \mathcal{S}^*[\![\mathtt{S}_t]\!]\}\end{aligned} \tag{6}$$

Prefix trace semantics (cont'd)

- Statement list $\texttt{Sl} ::= \texttt{Sl}'\ \texttt{S}$ (where $\textsf{at}[\![\texttt{S}]\!] = \textsf{after}[\![\texttt{Sl}']\!]$)

$$\begin{aligned} \mathcal{S}^*[\![\texttt{Sl}]\!] \;&\triangleq\; \mathcal{S}^*[\![\texttt{Sl}']\!] \\ &\cup \{\langle\pi_1,\ \pi_2 \frown \pi_3\rangle \mid \langle\pi_1,\ \pi_2\rangle \in \mathcal{S}^*[\![\texttt{Sl}']\!] \wedge \langle\pi_1 \frown \pi_2,\ \pi_3\rangle \in \mathcal{S}^*[\![\texttt{S}]\!]\} \end{aligned} \tag{8}$$

- Empty statement list $\texttt{Sl} ::= \epsilon$ (where $\textsf{at}[\![\texttt{Sl}]\!] \triangleq \textsf{after}[\![\texttt{Sl}]\!]$)

$$\mathcal{S}^*[\![\texttt{Sl}]\!] \;\triangleq\; \{\langle\pi\textsf{at}[\![\texttt{Sl}]\!],\ \textsf{at}[\![\texttt{Sl}]\!]\rangle \mid \pi\textsf{at}[\![\texttt{Sl}]\!] \in \mathbb{T}^+\} \tag{7}$$

Prefix trace semantics (cont'd)

- Iteration statement $\texttt{S} ::= \texttt{while}\ ^{\ell}\ \texttt{(B)}\ \texttt{S}_b$ (where $\mathsf{at}[\![\texttt{S}]\!] = \ell$)

$$
\begin{aligned}
\mathcal{S}^*[\![\texttt{S}]\!] &= \mathsf{lfp}^{\subseteq}\, \mathcal{F}^*[\![\texttt{S}]\!] && (9)\\
\mathcal{F}^*[\![\texttt{while}\ ^{\ell}\ \texttt{(B)}\ \texttt{S}_b]\!](X) &\triangleq \{\langle \pi_1\ell,\ \ell\rangle \mid \pi_1\ell \in \mathbb{T}^+\} && (a)\\
&\cup \{\langle \pi_1\ell,\ \ell\pi_2\ell \xrightarrow{\neg(\texttt{B})} \mathsf{after}[\![\texttt{S}]\!]\rangle \mid \langle \pi_1\ell,\ \ell\pi_2\ell\rangle \in X \wedge \mathcal{B}[\![\texttt{B}]\!]\boldsymbol{\varrho}(\pi_1\ell\pi_2\ell) = \mathrm{ff}\} && (b)\\
&\cup \{\langle \pi_1\ell,\ \ell\pi_2\ell \xrightarrow{\texttt{B}} \mathsf{at}[\![\texttt{S}_b]\!] \frown \pi_3\rangle \mid \langle \pi_1\ell,\ \ell\pi_2\ell\rangle \in X \wedge \\
&\qquad \mathcal{B}[\![\texttt{B}]\!]\boldsymbol{\varrho}(\pi_1\ell\pi_2\ell) = \mathrm{tt} \wedge \langle \pi_1\ell\pi_2\ell \xrightarrow{\texttt{B}} \mathsf{at}[\![\texttt{S}_b]\!],\ \pi_3\rangle \in \mathcal{S}^*[\![\texttt{S}_b]\!]\} && (c)
\end{aligned}
$$

Maximal trace semantics

- Maximal trace semantics

$$\mathcal{S}^{+}[\![\mathsf{S}]\!] \triangleq \{\langle\pi_1,\ \pi_2\ell\rangle \in \mathcal{S}^{*}[\![\mathsf{S}]\!] \mid (\ell = \mathsf{after}[\![\mathsf{S}]\!]) \vee (\mathsf{escape}[\![\mathsf{S}]\!] \wedge \ell = \mathsf{break\text{-}to}[\![\mathsf{S}]\!])\} \quad (11)$$

$$\mathcal{S}^{\infty}[\![\mathsf{S}]\!] \triangleq \lim(\mathcal{S}^{*}[\![\mathsf{S}]\!]) \quad (12)$$

- Limit

$$\lim \mathcal{T} \triangleq \{\langle\pi,\ \pi'\rangle \mid \pi' \in \mathbb{T}^{\infty} \wedge \forall n \in \mathbb{N}\ .\ \langle\pi,\ \pi'[0..n]\rangle \in \mathcal{T}\}. \quad (13)$$

Live variables analysis [Kennedy, 1975, 1976a,b]

Parameterized live variable abstraction on a trace

$$\alpha^{l}_{use,mod}[\![\mathsf{S}]\!]\ L_b, L_e\ \langle\pi_0,\ \pi\rangle$$

- After initialization π_0, execution of component S may continue with π
- L_b live variables if S escapes with a break
- L_e live variables if S terminates
- use defining the set $use[\![a]\!]\rho$ of variables which value is used when executing action a in environment ρ;
- mod defining the set $mod[\![a]\!]\rho$ of variables which value is modified when executing action a in environment ρ.

$\alpha^{l}_{use,mod}[\![\mathsf{S}]\!]\ L_b, L_e\ \langle\pi_0,\ \pi\rangle$ is the set of live variable $\mathsf{at}[\![\mathsf{S}]\!]$ for execution π initialized by π_0

Parameterized live variable analysis

Parameterized definition of the live variable abstraction of a trace

$$
\begin{aligned}
\alpha^{l}_{use,mod}[\![\mathsf{S}]\!]\, L_b, L_e\, \langle \pi_0,\ \ell \rangle &\triangleq \{\mathsf{x} \in \mathbb{V} \mid (\ell = \mathsf{after}[\![\mathsf{S}]\!] \wedge \mathsf{x} \in L_e) \vee && \text{(a)} \quad (14)\\
&\qquad (\mathsf{escape}[\![\mathsf{S}]\!] \wedge \ell = \mathsf{break\text{-}to}[\![\mathsf{S}]\!] \wedge \mathsf{x} \in L_b)\}\\
\alpha^{l}_{use,mod}[\![\mathsf{S}]\!]\, L_b, L_e\, \langle \pi_0,\ \ell \xrightarrow{a} \ell'\pi_1 \rangle &\triangleq \{\mathsf{x} \in \mathbb{V} \mid \mathsf{x} \in use[\![a]\!]\boldsymbol{\varrho}(\pi_0) \vee && \text{(b)}\\
&\qquad (\mathsf{x} \notin mod[\![a]\!]\boldsymbol{\varrho}(\pi_0) \wedge \mathsf{x} \in \alpha^{l}_{use,mod}[\![\mathsf{S}]\!]\, L_b, L_e\, \langle \pi_0 \frown \ell \xrightarrow{a} \ell',\ \ell'\pi_1 \rangle)\}
\end{aligned}
$$

> *A variable is live at some point if it holds a value that may be needed in the future, or equivalently if its value may be read before the next time the variable is written to.*

`https://en.wikipedia.org/wiki/Live_variable_analysis`

- *may be* $\longrightarrow$ potential liveness, *is* on one trace
- *or equivalently* $\longrightarrow$ wrong

Parameterized definition of the live variable abstraction of a trace

Lemma 1 If $\pi_1 = \ell_1 \xrightarrow{a_1} \ell_2 \xrightarrow{a_2} \ldots \xrightarrow{a_{n-1}} \ell_n$ and $\langle \pi_0,\ \pi_1 \rangle \in \mathcal{S}^*[\![\mathsf{S}]\!]$ then

$$\alpha^{l}_{use,mod}[\![\mathsf{S}]\!]\ L_b, L_e\ \langle \pi_0,\ \pi_1 \rangle \quad = \quad \{\mathsf{x} \in \mathbb{V} \mid \exists i \in [1, n-1]\ .\ \forall j \in [1, i-1]\ .$$
$$\mathsf{x} \notin mod[\![a_j]\!]\varrho(\pi_0 \frown \ell_1 \xrightarrow{a_1} \ell_2 \ldots \xrightarrow{a_{j-1}} \ell_j) \wedge \mathsf{x} \in use[\![a_i]\!]\varrho(\pi_0 \frown \ell_1 \xrightarrow{a_1} \ell_2 \ldots \xrightarrow{a_{i-1}} \ell_i)\}$$
$$\cup\ (\!\!|\ \ell_n = \mathsf{after}[\![\mathsf{S}]\!]\ ?\ L_e\ \mathbin{⁏}\ \varnothing\ |\!\!)\ \cup\ (\!\!|\ \mathsf{escape}[\![\mathsf{S}]\!] \wedge \ell_n = \mathsf{break\text{-}to}[\![\mathsf{S}]\!]\ ?\ L_b\ \mathbin{⁏}\ \varnothing\ |\!\!). \qquad \square$$

Parameterized definition of the live variable abstraction of a trace semantics

- Liveness

$$\alpha^{\exists l}_{use,mod}[\![\mathsf{S}]\!]\ \mathcal{S}\ L_b, L_e \ = \bigcup_{\langle\pi_0,\,\pi\rangle\,\in\,\mathcal{S}} \alpha^{l}_{use,mod}[\![\mathsf{S}]\!]\ L_b, L_e\ \langle\pi_0,\ \pi\rangle \qquad \text{potential liveness} \quad (15)$$

$$\alpha^{\forall l}_{use,mod}[\![\mathsf{S}]\!]\ \mathcal{S}\ L_b, L_e \ = \bigcap_{\langle\pi_0,\,\pi\rangle\,\in\,\mathcal{S}} \alpha^{l}_{use,mod}[\![\mathsf{S}]\!]\ L_b, L_e\ \langle\pi_0,\ \pi\rangle \qquad \text{definite liveness} \quad (16)$$

- Deadness is defined dually

$$\alpha^{\exists d}_{use,mod}[\![\mathsf{S}]\!]\ \mathcal{S}\ D_b, D_e \ = \neg\alpha^{\forall l}_{use,mod}[\![\mathsf{S}]\!]\ \mathcal{S}\ \neg D_b, \neg D_e \qquad \text{potential deadness} \quad (1)$$

$$\alpha^{\forall d}_{use,mod}[\![\mathsf{S}]\!]\ \mathcal{S}\ D_b, D_e \ = \neg\alpha^{\exists l}_{use,mod}[\![\mathsf{S}]\!]\ \mathcal{S}\ \neg D_b, \neg D_e \qquad \text{definite deadness} \quad (2)$$

Parameterized definition of the live variable abstraction of a trace semantics

Lemma 2 $\alpha^{\exists l}_{use,mod}[\![\mathtt{S}]\!]\,(\mathcal{S}^{+\infty}[\![\mathtt{S}]\!]) = \alpha^{\exists l}_{use,mod}[\![\mathtt{S}]\!]\,(\mathcal{S}^{*}[\![\mathtt{S}]\!])$. $\square$

Instances of the parameterized live variable analysis

Semantic liveness/deadness

- *use*

$$\begin{aligned}
\mathsf{use}[\![\mathsf{skip}]\!]\,\rho &\triangleq \emptyset \\
\mathsf{use}[\![\mathsf{x}\ \texttt{=}\ \mathsf{A}]\!]\,\rho &\triangleq \{\mathsf{y} \mid \exists \nu \in \mathbb{V}\ .\ \mathscr{A}[\![\mathsf{A}]\!]\,\rho \neq \mathscr{A}[\![\mathsf{A}]\!]\,\rho[\mathsf{y} \leftarrow \nu] \wedge \rho(\mathsf{x}) \neq \mathscr{A}[\![\mathsf{A}]\!]\,\rho\} \\
\mathsf{use}[\![a]\!]\,\rho &\triangleq \{\mathsf{y} \mid \exists \nu \in \mathbb{V}\ .\ \mathscr{B}[\![a]\!]\,\rho \neq \mathscr{B}[\![a]\!]\,\rho[\mathsf{y} \leftarrow \nu]\} \qquad a \in \{\mathsf{B}, \neg(\mathsf{B})\}
\end{aligned} \tag{19}$$

- *mod*

$$\mathsf{mod}[\![a]\!]\,\rho \triangleq \{\mathsf{x} \mid a = (\mathsf{x}\ \texttt{=}\ \mathsf{A}) \wedge (\rho(\mathsf{x}) \neq \mathscr{A}[\![\mathsf{A}]\!]\,\rho)\}$$

- Semantic potential liveness

$$\mathcal{S}^{\exists l}[\![\mathsf{S}]\!] \triangleq \alpha^{\exists l}_{\mathsf{use},\mathsf{mod}}[\![\mathsf{S}]\!]\,(\mathcal{S}^{+\infty}[\![\mathsf{S}]\!]) \tag{20}$$

Classical syntactic liveness/deadness

- *use*

$$\begin{aligned} \mathbb{use}[\![x = A]\!]\,\rho &\triangleq \mathbb{vars}[\![A]\!] \\ \mathbb{use}[\![skip]\!]\,\rho &\triangleq \emptyset \\ \mathbb{use}[\![B]\!]\,\rho &\triangleq \mathbb{use}[\![\neg(B)]\!]\,\rho \quad\triangleq\quad \mathbb{vars}[\![B]\!] \end{aligned} \tag{21}$$

(ρ is useless)

- *mod*

$$\begin{aligned} \mathbb{mod}[\![x = A]\!]\,\rho &\triangleq \{x\} \\ \mathbb{mod}[\![skip]\!]\,\rho &\triangleq \emptyset \\ \mathbb{mod}[\![B]\!]\,\rho &\triangleq \mathbb{mod}[\![\neg(B)]\!]\,\rho \quad\triangleq\quad \emptyset \end{aligned}$$

- Classical syntactic potential liveness

$$\mathcal{S}^{\exists l}[\![S]\!] \triangleq \alpha^{\exists l}_{\mathbb{use},\mathbb{mod}}[\![S]\!]\,(\mathcal{S}^{+\infty}[\![S]\!]) \tag{22}$$

Soundness expectation

- Soundness of potential liveness

$$\mathcal{S}^{\exists \mathsf{l}}[\![\mathsf{s}]\!] \subseteq \mathcal{S}^{\exists \mathbb{l}}[\![\mathsf{s}]\!]$$

Soundness expectation

- Soundness of potential liveness

$$\mathcal{S}^{\exists\mathsf{l}}[\![\mathsf{S}]\!] \subseteq \mathcal{S}^{\exists\mathbb{l}}[\![\mathsf{S}]\!]$$

- THIS IS <u>NOT</u> TRUE!

Soundness expectation

- Soundness of potential liveness

$$\mathcal{S}^{\exists l}[\![\mathtt{s}]\!] \subseteq \mathcal{S}^{\exists \mathbb{l}}[\![\mathtt{s}]\!]$$

- THIS IS <u>NOT</u> TRUE!
- Problem

$$\exists a \mathrel{.} \exists \rho \in \mathbb{E}\mathbb{v} \mathrel{.} \mathtt{x} \in \mathbb{mod}[\![a]\!]\,\rho \wedge \mathtt{x} \notin \mathsf{mod}[\![a]\!]\,\rho$$

Soundness expectation

- Soundness of potential liveness

$$\mathcal{S}^{\exists l}[\![\mathsf{s}]\!] \subseteq \mathcal{S}^{\mathbb{\exists l}}[\![\mathsf{s}]\!]$$

- THIS IS <u>NOT</u> TRUE!
- Problem

$$\exists a \,.\, \exists \rho \in \mathbb{E}\mathbb{v} \,.\, \mathsf{x} \in \mathbb{mod}[\![a]\!]\,\rho \wedge \mathsf{x} \notin \mathsf{mod}[\![a]\!]\,\rho$$

- Counter-example

```
x = x;
```

If the compiler eliminates that assignment, this changes syntactic liveness (but not semantic liveness). For soundness, the syntactic liveness analysis must be redone after useless assignment elimination.

How to fix the problem?

- Change the live variable algorithm to be sound with respect to the semantic definition
  - $\Rightarrow$ this becomes a liveness/eventuality problem, requires variant functions, *etc*.
  - $\Rightarrow$ too complicated for compilers!
- Keep the live variable algorithm, but change the notion of soundness
  - $\Rightarrow$ this limits compiler optimizations, or
    requires a recomputation of the live variable information after the program transformation
  - $\Rightarrow$ less complicated for compilers (which may even be incorrect if the live variable analysis is not redone after program transformation)

Restating soundness

- Define

$$\mathcal{S}^{\exists l}[\![\mathsf{s}]\!] \triangleq \alpha^{\exists l}_{\mathsf{use},\mathbb{mod}}(\mathcal{S}^{+\infty}[\![\mathsf{s}]\!]) \tag{24}$$

Theorem 1 If $\alpha^{\exists l}_{\mathbb{use},\mathbb{mod}}[\![\mathsf{s}]\!](\mathcal{S}^{+\infty}[\![\mathsf{s}]\!]) \dot{\subseteq} \mathcal{S}^{\exists l}[\![\mathsf{s}]\!]$ then $\mathcal{S}^{\exists l}[\![\mathsf{s}]\!] \dot{\subseteq} \mathcal{S}^{\exists l}[\![\mathsf{s}]\!]$.

- Follows from

$$\exists \rho \in \mathbb{Ev} \,.\, \mathsf{y} \in \mathsf{use}[\![a]\!]\rho \Rightarrow \forall \rho \in \mathbb{Ev} \,.\, \mathsf{y} \in \mathbb{use}[\![a]\!]\rho \tag{23}$$

- Intuition

  A variable is live at some point if it holds a value that may be necessarily used before the next time the variable is *<u>assigned</u>* *to.*

Calculational design of the structural syntactic potential liveness static analysis

Calculational design

- Based on the soundness definition

$$\alpha^{\exists l}_{\mathtt{use,mod}}[\![\mathsf{S}]\!]\,(\mathcal{S}^*[\![\mathsf{S}]\!]) \mathrel{\dot{\subseteq}} \mathcal{S}^{\exists \mathrm{l}}[\![\mathsf{S}]\!]$$

- Method
  - by structural induction on program components S
  - develop $\alpha^{\exists l}_{\mathtt{use,mod}}[\![\mathsf{S}]\!]\,(\mathcal{S}^*[\![\mathsf{S}]\!])$ to eliminate the abstraction $\alpha^{\exists l}_{\mathtt{use,mod}}[\![\mathsf{S}]\!]$
  - over-approximate to eliminate all concrete computations (*e.g.* value of a test with dead branch)

Assignment $\texttt{S} ::= \ell\ \texttt{x = A ;}$

$$
\begin{aligned}
& \mathcal{S}^{\exists l}[\![\texttt{S}]\!]\, L_b, L_e \\
= \ & \alpha^{\exists l}_{\mathrm{use,mod}}[\![\texttt{S}]\!](\mathcal{S}^*[\![\texttt{S}]\!])\, L_b, L_e && \text{⦃(22) and Lemma 2⦄} \\
= \ & \bigcup\{\alpha^{l}_{\mathrm{use,mod}}[\![\texttt{S}]\!]\, L_b, L_e\, \langle\pi_0,\ \pi_1\rangle \mid \langle\pi_0,\ \pi_1\rangle \in \widehat{\mathcal{S}}^*[\![\texttt{S}]\!]\} && \text{⦃def. (15) of } \alpha^{\exists l}_{\mathrm{use,mod}}[\![\texttt{S}]\!]\text{⦄} \\
= \ & \bigcup\{\alpha^{l}_{\mathrm{use,mod}}[\![\texttt{S}]\!]\, L_b, L_e \langle\pi_0 \mathsf{at}[\![\texttt{S}]\!], \mathsf{at}[\![\texttt{S}]\!]\rangle\} \cup \bigcup\{\alpha^{l}_{\mathrm{use,mod}}[\![\texttt{S}]\!]\, L_b, L_e \langle\pi_0 \mathsf{at}[\![\texttt{S}]\!], \mathsf{at}[\![\texttt{S}]\!] \xrightarrow{\texttt{x = A} = \mathcal{A}[\![\texttt{A}]\!]\varrho(\pi_0 \mathsf{at}[\![\texttt{S}]\!])} \mathsf{after}[\![\texttt{S}]\!]\rangle\} \\
& && \text{⦃def. (3) of } \mathcal{S}^*[\![\texttt{S}]\!]\text{⦄} \\
= \ & \bigcup\{\alpha^{l}_{\mathrm{use,mod}}[\![\texttt{S}]\!]\, L_b, L_e\, \langle\pi_0 \mathsf{at}[\![\texttt{S}]\!],\ \mathsf{at}[\![\texttt{S}]\!] \xrightarrow{\texttt{x = A} = \mathcal{A}[\![\texttt{A}]\!]\varrho(\pi_0 \mathsf{at}[\![\texttt{S}]\!])} \mathsf{after}[\![\texttt{S}]\!]\rangle\} \\
& && \text{⦃def. (14.a) of } \alpha^{l}_{\mathrm{use,mod}}[\![\texttt{S}]\!]\, L_b, L_e\, \langle\pi_0 \mathsf{at}[\![\texttt{S}]\!],\ \mathsf{at}[\![\texttt{S}]\!]\rangle = \varnothing\text{⦄} \\
= \ & \bigcup\{\texttt{y} \in \mathbb{V} \mid \texttt{y} \in \mathsf{use}[\![\texttt{x = A}]\!]\varrho(\pi_0 \mathsf{at}[\![\texttt{S}]\!]) \vee (\texttt{y} \notin \mathsf{mod}[\![\texttt{x = A}]\!]\varrho(\pi_0 \mathsf{at}[\![\texttt{S}]\!]) \wedge \texttt{y} \in \alpha^{l}_{\mathrm{use,mod}}[\![\texttt{S}]\!]\, L_b, L_e\, \langle\pi_0 \mathsf{at}[\![\texttt{S}]\!] \\
& \quad \mathsf{at}[\![\texttt{S}]\!] \xrightarrow{\texttt{x = A} = \mathcal{A}[\![\texttt{A}]\!]\varrho(\pi_0 \mathsf{at}[\![\texttt{S}]\!])} \mathsf{after}[\![\texttt{S}]\!],\ \mathsf{after}[\![\texttt{S}]\!]\rangle)\} \\
& && \text{⦃def. (14.b) of } \alpha^{l}_{\mathrm{use,mod}}\, L_b, L_e\, \langle\pi_0 \mathsf{at}[\![\texttt{S}]\!],\ \mathsf{at}[\![\texttt{S}]\!] \xrightarrow{\texttt{x = A} = \mathcal{A}[\![\texttt{A}]\!]\varrho(\pi_0 \mathsf{at}[\![\texttt{S}]\!])} \mathsf{after}[\![\texttt{S}]\!]\rangle\text{⦄} \\
= \ & \{\texttt{y} \in \mathbb{V} \mid \texttt{y} \in \mathsf{use}[\![\texttt{x = A}]\!] \vee (\texttt{y} \notin \mathsf{mod}[\![\texttt{x = A}]\!] \wedge \texttt{y} \in L_e)\} \\
& \quad \text{⦃def. (14.a) of } \alpha^{l}_{\mathrm{use,mod}}[\![\texttt{S}]\!]\, L_b, L_e\, \langle\pi_0,\ \mathsf{after}[\![\texttt{S}]\!]\rangle \triangleq \{\texttt{x} \in \mathbb{V} \mid \texttt{x} \in L_e\} = L_e \text{ since } \mathsf{escape}[\![\texttt{S}]\!] = \mathsf{ff} \text{ and} \\
& \quad \text{omitting the useless parameters of } \mathsf{use} \text{ and } \mathsf{mod}\text{⦄} \\
= \ & \mathsf{use}[\![\texttt{x = A}]\!] \cup (L_e \setminus \mathsf{mod}[\![\texttt{x = A}]\!]) && \text{⦃def. } \in\text{⦄} \\
\triangleq \ & \widehat{\mathcal{S}}^{\exists l}[\![\texttt{x = A ;}]\!]\, L_b, L_e && \text{⦃ Id Est Ratione (without approximation!)⦄}
\end{aligned}
$$

Potentially live variables

Structural syntactic potential liveness analysis

$$\begin{aligned}
\widehat{\mathcal{S}}^{\exists l}[\![\mathtt{Sl}\ \ell]\!]\ L_e &\triangleq \widehat{\mathcal{S}}^{\exists l}[\![\mathtt{Sl}\ \ell]\!]\ \varnothing, L_e && (25)\\
\widehat{\mathcal{S}}^{\exists l}[\![\mathtt{x = A ;}]\!]\ L_b, L_e &\triangleq \mathbb{use}[\![\mathtt{x = A}]\!] \cup (L_e \setminus \mathbb{mod}[\![\mathtt{x = A}]\!])\\
\widehat{\mathcal{S}}^{\exists l}[\![\mathtt{;}]\!]\ L_b, L_e &\triangleq L_e\\
\widehat{\mathcal{S}}^{\exists l}[\![\mathtt{Sl}'\ \mathtt{S}]\!]\ L_b, L_e &\triangleq \widehat{\mathcal{S}}^{\exists l}[\![\mathtt{Sl}']\!]\ L_b, (\widehat{\mathcal{S}}^{\exists l}[\![\mathtt{S}]\!]\ L_b, L_e)\\
\widehat{\mathcal{S}}^{\exists l}[\![\ \epsilon\ ]\!]\ L_b, L_e &\triangleq L_e\\
\widehat{\mathcal{S}}^{\exists l}[\![\mathtt{if\ (B)\ S}_t]\!]\ L_b, L_e &\triangleq \mathbb{use}[\![\mathtt{B}]\!] \cup L_e \cup \widehat{\mathcal{S}}^{\exists l}[\![\mathtt{S}_t]\!]\ L_b, L_e\\
\widehat{\mathcal{S}}^{\exists l}[\![\mathtt{if\ (B)\ S}_t\ \mathtt{else}\ \mathtt{S}_f]\!]\ L_b, L_e &\triangleq \mathbb{use}[\![\mathtt{B}]\!] \cup \widehat{\mathcal{S}}^{\exists l}[\![\mathtt{S}_t]\!]\ L_b, L_e \cup \widehat{\mathcal{S}}^{\exists l}[\![\mathtt{S}_f]\!]\ L_b, L_e\\
\widehat{\mathcal{S}}^{\exists l}[\![\mathtt{while\ (B)\ S}_b]\!]\ L_b, L_e &\triangleq \mathbb{use}[\![\mathtt{B}]\!] \cup L_e \cup \widehat{\mathcal{S}}^{\exists l}[\![\mathtt{S}_b]\!]\ L_b, L_e\\
\widehat{\mathcal{S}}^{\exists l}[\![\mathtt{break\ ;}]\!]\ L_b, L_e &\triangleq L_b\\
\widehat{\mathcal{S}}^{\exists l}[\![\mathtt{\{\ Sl\ \}}]\!]\ L_b, L_e &\triangleq \widehat{\mathcal{S}}^{\exists l}[\![\mathtt{Sl}]\!]\ L_b, L_e && \square
\end{aligned}$$

A surprise

The fixpoint in the structural syntactic potential liveness analysis of the iteration **`while (B)`** $\mathtt{S}_b$ is a constant.

Definitely dead variables

Structural syntactic definite deadness analysis

$$
\begin{aligned}
\widehat{\mathcal{S}}^{\forall d}[\![\mathtt{Sl}\ \ell]\!]\ D_e &= \widehat{\mathcal{S}}^{\forall d}[\![\mathtt{Sl}\ \ell]\!]\ \mathbb{V}, D_e \qquad (26)\\
\widehat{\mathcal{S}}^{\forall d}[\![\mathtt{x = A ;}]\!]\ D_b, D_e &= \neg\,\mathrm{use}[\![\mathtt{x = A}]\!] \cap (D_e \cup \mathrm{mod}[\![\mathtt{x = A}]\!])\\
\widehat{\mathcal{S}}^{\forall d}[\![\mathtt{;}]\!]\ D_b, D_e &= D_e\\
\widehat{\mathcal{S}}^{\forall d}[\![\mathtt{Sl}'\ \mathtt{S}]\!]\ D_b, D_e &= \widehat{\mathcal{S}}^{\forall d}[\![\mathtt{Sl}']\!]\ D_b, (\widehat{\mathcal{S}}^{\forall d}[\![\mathtt{S}]\!]\ D_b, D_e)\\
\widehat{\mathcal{S}}^{\forall d}[\![\epsilon]\!]\ D_b, D_e &= D_e\\
\widehat{\mathcal{S}}^{\forall d}[\![\mathtt{if\ (B)\ S}_t]\!]\ D_b, D_e &= \neg\,\mathrm{use}[\![\mathtt{B}]\!] \cap D_e \cap \widehat{\mathcal{S}}^{\forall d}[\![\mathtt{S}_t]\!]\ D_b, D_e\\
\widehat{\mathcal{S}}^{\forall d}[\![\mathtt{if\ (B)\ S}_t\ \mathtt{else\ S}_f]\!]\ D_b, D_e &= \neg\,\mathrm{use}[\![\mathtt{B}]\!] \cap \widehat{\mathcal{S}}^{\forall d}[\![\mathtt{S}_t]\!]\ D_b, D_e \cap \widehat{\mathcal{S}}^{\forall d}[\![\mathtt{S}_f]\!]\ D_b, D_e\\
\widehat{\mathcal{S}}^{\forall d}[\![\mathtt{while\ (B)\ S}_b]\!]\ D_b, D_e &= \neg\,\mathrm{use}[\![\mathtt{B}]\!] \cap D_e \cap \widehat{\mathcal{S}}^{\forall d}[\![\mathtt{S}_b]\!]\ D_b, D_e\\
\widehat{\mathcal{S}}^{\forall d}[\![\mathtt{break ;}]\!]\ D_b, D_e &= D_b\\
\widehat{\mathcal{S}}^{\forall d}[\![\mathtt{\{\ Sl\ \}}]\!]\ D_b, D_e &= \widehat{\mathcal{S}}^{\forall d}[\![\mathtt{Sl}]\!]\ D_b, D_e \qquad \square
\end{aligned}
$$

Conclusion

Conclusion

- Classical definitions of the soundness of data flow analyses [Beyer, Gulwani, and Schmidt, 2018; Kildall, 1973; Schmidt, 1998; Steffen, 1991, 1993] are specified with respect to an abstraction of the semantics not the semantics itself, which is confusing
- Transition systems forget about the program structure[1] so lead to iterations that may be useless

[1] see however Patrick Cousot & Radhia Cousot. "À la Floyd" induction principles for proving inevitability properties of programs. In «Algebraic methods in semantics», M. Nivat & J. Reynolds (Eds.), Cambridge University Press, Cambridge, UK, pp. 277—312, December 1985.

Conclusion

- Classical definitions of the soundness of data flow analyses [Beyer, Gulwani, and Schmidt, 2018; Kildall, 1973; Schmidt, 1998; Steffen, 1991, 1993] are specified with respect to an abstraction of the semantics not the semantics itself, which is confusing
- Transition systems forget about the program structure[1] so lead to iterations that may be useless
- Why CompCert get it right?
  - does simultaneously the liveness analysis and the program transformation based on this analysis
  - returns the result of the liveness analysis valid after the transformation
  - justifies by dependency: a variable is dead if nothing later depends on its value

[1] see however Patrick Cousot & Radhia Cousot. "À la Floyd" induction principles for proving inevitability properties of programs. In «Algebraic methods in semantics», M. Nivat & J. Reynolds (Eds.), Cambridge University Press, Cambridge, UK, pp. 277—312, December 1985.

Conclusion

- Anonymous reviewer

  "It is an old story that the dataflow analysis framework ("syntactic" dataflow analysis in paper's characterization) is way too weak. For modern programming languages, control flow is not syntactic but a part of semantics. Dataflow analysis assumes the control flow to be available before the analysis hence a stalemate for modern languages with higher order functions, dynamic bindings, or dynamic gotos; dataflow analysis has neither a systematic guide to prove the correctness of an analysis nor systematic approach to manage the precision of the analysis. On the other hand, the semantics-based design theory (abstract interpretation) is general enough to handle any kind of source languages and powerful enough to prove the correctness and to manage its precision."

Bibliography

References I

Beyer, Dirk, Sumit Gulwani, and David A. Schmidt (2018). "Combining Model Checking and Data-Flow Analysis". In: *Handbook of Model Checking*. Springer, pp. 493–540 (2, 36, 37).

Kennedy, Ken (1975). "Node Listings Applied to Data Flow Analysis". In: *POPL*. ACM Press, pp. 10–21 (13).

– (Mar. 1976a). "A Comparison of Two Algorithms for Global Data Flow Analysis". *SIAM J. Comput.* 5.1, pp. 158–180 (13).

– (1976b). "A Comparison of Two Algorithms for Global Data Flow Analysis". *Int. J. of Comp. Math.* Section A, Volume 3, pp. 5–15 (13).

Kildall, Gary A. (1973). "A Unified Approach to Global Program Optimization". In: *POPL*. ACM Press, pp. 194–206 (2, 36, 37).

Schmidt, David A. (1998). "Data Flow Analysis is Model Checking of Abstract Interpretations". In: *POPL*. ACM, pp. 38–48 (2, 36, 37).

References II

Steffen, Bernhard (1991). "Data Flow Analysis as Model Checking". In: *TACS*. Vol. 526. Lecture Notes in Computer Science. Springer, pp. 346–365 (2, 36, 37).

– (1993). "Generating Data Flow Analysis Algorithms from Modal Specifications". *Sci. Comput. Program.* 21.2, pp. 115–139 (2, 36, 37).

The End, Thank you