

Static analysis and all that

Martin Steffen

Autumn 2022



Chapter 2

Data flow analysis

Course “Static analysis and all that”

Martin Steffen

IN5440 / autumn 2022



Chapter 2

Learning Targets of Chapter "Data flow analysis".

various DFAs

monotone frameworks

operational semantics

foundations

special topics (SSA, context-sensitive analysis ...)



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs



Chapter 2

Outline of Chapter "Data flow analysis".

Intraprocedural analysis

- Determining the control flow graph
- Available expressions
- Reaching definitions
- Very busy expressions
- Live variable analysis

Theoretical properties and semantics

- Semantics
- Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

- Introduction
- Extending the semantics and the CFGs



Section

Intraprocedural analysis

- Determining the control flow graph
- Available expressions
- Reaching definitions
- Very busy expressions
- Live variable analysis

Chapter 2 “Data flow analysis”
Course “Static analysis and all that”
Martin Steffen
IN5440 / autumn 2022

While language and control flow graph

- starting point: while language from the intro
- *labelled* syntax (unique labels)
- labels $=$ *nodes* of the cfg
- initial and final labels
- edges of a cfg: given by function flow

3 functions (definition see script / book)

1. $\mathit{init} : \mathbf{Stmt} \to \mathbf{Lab}$
2. $\mathit{final} : \mathbf{Stmt} \to 2^{\mathbf{Lab}}$
3. $\mathit{flow} : \mathbf{Stmt} \to 2^{\mathbf{Lab} \times \mathbf{Lab}}$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Flow and reverse flow

$$labels(S) = init(S) \cup \{l \mid (l, l') \in flow(S)\} \cup \{l' \mid (l, l') \in flow(S)\}$$

- data flow analysis can be *forward* (like RD) or backward
- $flow$: for forward analyses
- for backward analyses: *reverse* flow $flow^R$, simply invert the edges



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Program of interest

- S_*: program being analysed, top-level statement
- analogously $\mathbf{Lab}_*$, $\mathbf{Var}_*$, $\mathbf{Blocks}_*$
- *trivial* expression: a single variable or constant
- $\mathbf{AExp}_*$: non-trivial arithmetic sub-expr. of S_*, analogous for $\mathbf{AExp}(a)$ and $\mathbf{AExp}(b)$.
- useful restrictions
  - *isolated entries*: $(l, init(S_*)) \notin flow(S_*)$
  - *isolated exits* $\forall l_1 \in final(S_*). \quad (l_1, l_2) \notin flow(S_*)$
  - *label consistency*

    $[B_1]^l, [B_2]^l \in blocks(S)$ then $B_1 = B_2$

"l labels *the* block B"

- even better: *unique* labelling



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Avoid recomputation: Available expressions

$$[x := a + b]^0; [y := a * b]^1; \quad \texttt{while} \quad [y > a + b]^2 \quad \texttt{do} \quad ([a := a + 1]^3; [x := a + b]^4)$$

- usage: avoid *re-computation* of expression

Avoid recomputation: Available expressions

$$[x := a + b]^0; [y := a * b]^1; \quad \texttt{while} \quad [y > a + b]^2 \quad \texttt{do} \quad ([a := a + 1]^3; [x := a + b]^4)$$

- usage: avoid *re-computation* of expression

Goal

For each program point: which expressions must have already been computed (and not later modified), on all paths to the program point.

Available expressions: general

- given as flow *equations* (not $\subseteq$-constraints, but not really crucial, as we know already)
- uniform representation of *effect of basic blocks* (= *intra-block flow*)

2 ingredients of intra-block flow

- *kill:* flow information "eliminated" passing through the basic blocks
- *generate:* flow information "generated new" passing through the basic blocks

- later analyses: presented similarly
- different analyses $\Rightarrow$ different kind of flow information + different kill- and generate-functions



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Available expressions: types

- interested in *sets of expressions*: $2^{\mathbf{AExp}_*}$
- generation and killing:

$$kill_{\mathsf{AE}}, gen_{\mathsf{AE}} : \mathbf{Blocks}_* \to 2^{\mathbf{AExp}_*}$$

- analysis: pair of functions

$$\mathsf{AE}_{entry}, \mathsf{AE}_{exit} : \mathbf{Lab}_* \to 2^{\mathbf{AExp}_*}$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Intra-block flow specification: Kill and generate

$$
\begin{aligned}
kill_{\mathsf{AE}}([x := a]^l) &= \\
kill_{\mathsf{AE}}([\mathsf{skip}]^l) &= \\
kill_{\mathsf{AE}}([b]^l) &= \\
\\
gen_{\mathsf{AE}}([x := a]^l) &= \\
gen_{\mathsf{AE}}([\mathsf{skip}]^l) &= \\
gen_{\mathsf{AE}}([b]^l) &=
\end{aligned}
$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Intra-block flow specification: Kill and generate

$$
\begin{aligned}
kill_{\mathsf{AE}}([x := a]^l) &= \{a' \in \mathbf{AExp}_* \mid x \in fv(a')\} \\
kill_{\mathsf{AE}}([\mathsf{skip}]^l) &= \emptyset \\
kill_{\mathsf{AE}}([b]^l) &= \emptyset
\end{aligned}
$$

$$
\begin{aligned}
gen_{\mathsf{AE}}([x := a]^l) &= \{a' \in \mathbf{AExp}(a) \mid x \notin fv(a')\} \\
gen_{\mathsf{AE}}([\mathsf{skip}]^l) &= \emptyset \\
gen_{\mathsf{AE}}([b]^l) &= \mathbf{AExp}(b)
\end{aligned}
$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Flow equations: $\mathsf{AE}^{=}$

split into

nodes: intra-block equations, using *kill* and *generate*

edges: inter-block equations, using *flow*

Flow equations for AE

$$\mathsf{AE}_{entry}(l) = \begin{cases} \emptyset & l = init(S_*) \\ \bigcap\{\mathsf{AE}_{exit}(l') \mid (l', l) \in flow(S_*)\} & \text{otherwise} \end{cases}$$

$$\mathsf{AE}_{exit}(l) = \mathsf{AE}_{entry}(l) \setminus kill_{\mathsf{AE}}(B^l) \cup gen_{\mathsf{AE}}(B^l)$$

where $B^l \in blocks(S_*)$

- note the *"order"* of kill and generate

Available expressions

- *forward* analysis (as RD)
- interest in *largest* solution (unlike RD)

$\Rightarrow$ must analysis (as opposed to *may*)

- expression is available: if *no path kills it*
- remember: informal description of AE: expression available on *all paths* (i.e., not killed on any)



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Example AE

$$[x := a + b]^0; [y := a * b]^1; \quad \texttt{while} \quad [y > a + b]^2 \quad \texttt{do} \quad ([a := a + 1]^3; [x := a + b]^4)$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

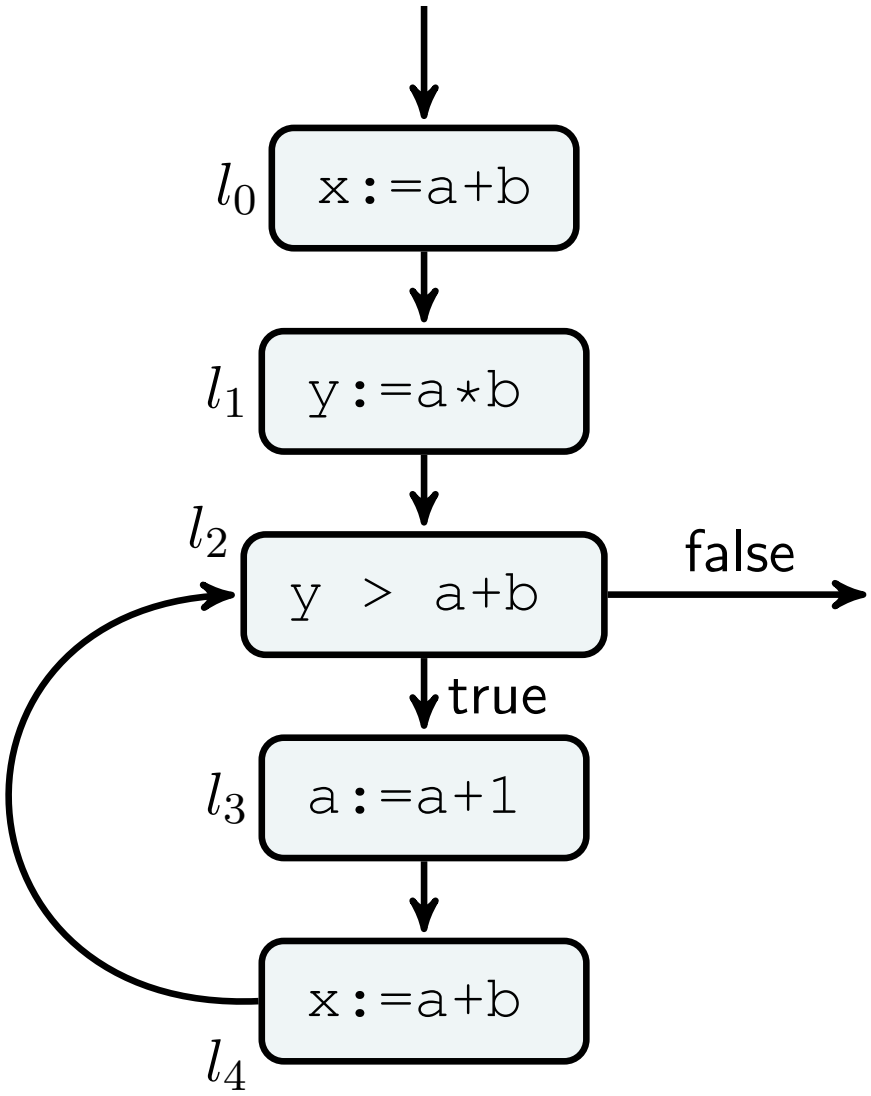
Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Example AE





Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Reaching definitions

- remember the intro
- here: the *same* analysis, but based on the new definitions: kill, generate, flow ...

$$[x := 5]^0; [y := 1]^1; \texttt{while}[x > 1]^2 \texttt{do}([y := x*y]^3; [x := x-1]^4)$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

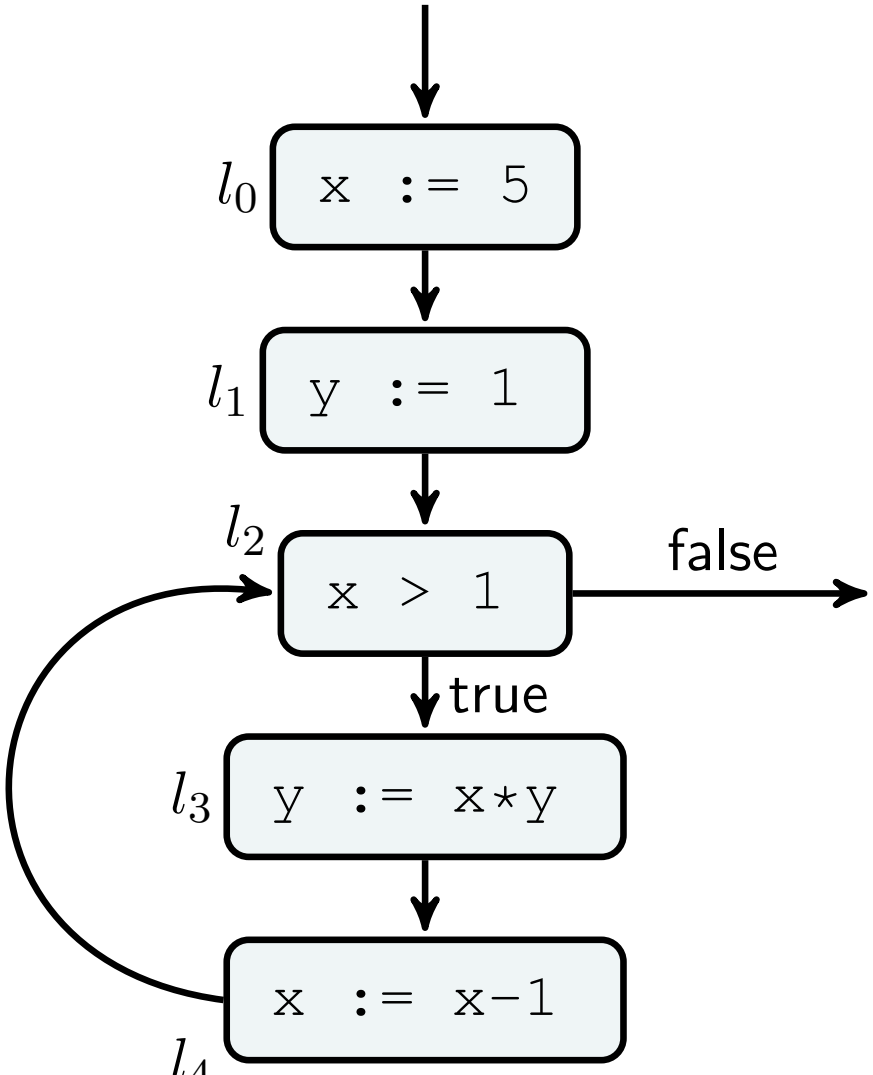
Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Reaching definitions

- remember the intro
- here: the *same* analysis, but based on the new definitions: kill, generate, flow ...



Reaching definitions: types

- interest in *sets of tuples of var's and program points i.e., labels*:

$$2^{\mathbf{Var}_* \times \mathbf{Lab}_*^?} \quad \text{where} \quad \mathbf{Lab}_*^? = \mathbf{Lab}_* + \{?\}$$

- generation and killing:

$$kill_{\mathsf{RD}}, gen_{\mathsf{RD}} : \mathbf{Blocks}_* \to 2^{\mathbf{Var}_* \times \mathbf{Lab}_*^?}$$

- analysis: pair of mappings

$$\mathsf{RD}_{entry}, \mathsf{RD}_{exit} : \mathbf{Lab}_* \to 2^{\mathbf{Var}_* \times \mathbf{Lab}_*^?}$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Reaching defs: kill and generate

$$
\begin{aligned}
kill_{\mathsf{RD}}([x := a]^l) &= \\
kill_{\mathsf{RD}}([\mathsf{skip}]^l) &= \\
kill_{\mathsf{RD}}([b]^l) &= \\
\\
gen_{\mathsf{RD}}([x := a]^l) &= \\
gen_{\mathsf{RD}}([\mathsf{skip}]^l) &= \\
gen_{\mathsf{RD}}([b]^l) &=
\end{aligned}
$$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Reaching defs: kill and generate

$$
\begin{aligned}
kill_{\mathsf{RD}}([x := a]^l) &= \{(x, ?)\} \cup \\
&\quad \bigcup \{(x, l') \mid B^{l'} \text{ is assgm. to } x \text{ in } S_*\} \\
kill_{\mathsf{RD}}([\mathsf{skip}]^l) &= \emptyset \\
kill_{\mathsf{RD}}([b]^l) &= \emptyset
\end{aligned}
$$

$$
\begin{aligned}
gen_{\mathsf{RD}}([x := a]^l) &= \{(x, l)\} \\
gen_{\mathsf{RD}}([\mathsf{skip}]^l) &= \emptyset \\
gen_{\mathsf{RD}}([b]^l) &= \emptyset
\end{aligned}
$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Flow equations: $\mathsf{RD}^{=}$

split into

- *intra*-block equations, using *kill* and *generate*
- *inter*-block equations, using *flow*

Flow equations for RD

$$\mathsf{RD}_{entry}(l) = \;$$

$$\mathsf{RD}_{exit}(l) = \mathsf{RD}_{entry}(l) \setminus kill_{\mathsf{RD}}(B^l) \cup gen_{\mathsf{RD}}(B^l)$$

where $B^l \in blocks(S_*)$

- same order of kill/generate

Flow equations: $\mathsf{RD}^{=}$

split into

- *intra*-block equations, using *kill* and *generate*
- *inter*-block equations, using *flow*

Flow equations for RD

$$\mathsf{RD}_{entry}(l) = \begin{cases} \{(x, ?) \mid x \in fv(S_*)\} & l = init(S_*) \\ \bigcup\{\mathsf{RD}_{exit}(l') \mid (l', l) \in flow(S_*)\} & \text{otherwise} \end{cases}$$

$$\mathsf{RD}_{exit}(l) = \mathsf{RD}_{entry}(l) \setminus kill_{\mathsf{RD}}(B^l) \cup gen_{\mathsf{RD}}(B^l)$$

where $B^l \in blocks(S_*)$

- same order of kill/generate

Very busy expressions

$$
\begin{array}{ll}
\texttt{if} & [a > b]^1 \\
\texttt{then} & [x := b - a]^2; [y := a - b]^3 \\
\texttt{else} & [a := b - a]^4; [x := a - b]^5
\end{array}
$$

Definition (Very busy expression)

An expression is *very busy* at the exit of a label, if for "all" paths from that label, the expression is used before any of its variables are "redefined" (= overwritten).

- usage: expression "hoisting"

Goal

For each program point, which expressions are very busy at the *exit* of that point.



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Very busy expressions: types

- interested in: *sets of expressions*: $2^{\mathbf{AExp}_*}$
- generation and killing:

$$kill_{\mathsf{VB}}, gen_{\mathsf{VB}} : \mathbf{Blocks}_* \to 2^{\mathbf{AExp}_*}$$

- analysis: pair of mappings

$$\mathsf{VB}_{entry}, \mathsf{VB}_{exit} : \mathbf{Lab}_* \to 2^{\mathbf{AExp}_*}$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Very busy expr.: kill and generate

core of the intra-block flow specification

$$
\begin{aligned}
kill_{\mathsf{VB}}([x := a]^l) &= \\
kill_{\mathsf{VB}}([\mathsf{skip}]^l) &= \\
kill_{\mathsf{VB}}([b]^l) &= \\
\\
gen_{\mathsf{VB}}([x := a]^l) &= \\
gen_{\mathsf{VB}}([\mathsf{skip}]^l) &= \\
gen_{\mathsf{VB}}([b]^l) &=
\end{aligned}
$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Very busy expr.: kill and generate

core of the intra-block flow specification

$$
\begin{aligned}
kill_{\mathsf{VB}}([x := a]^l) &= \{a' \in \mathbf{AExp}_* \mid x \in fv(a')\} \\
kill_{\mathsf{VB}}([\mathsf{skip}]^l) &= \emptyset \\
kill_{\mathsf{VB}}([b]^l) &= \emptyset
\end{aligned}
$$

$$
\begin{aligned}
gen_{\mathsf{VB}}([x := a]^l) &= \mathbf{AExp}(a) \\
gen_{\mathsf{VB}}([\mathsf{skip}]^l) &= \emptyset \\
gen_{\mathsf{VB}}([b]^l) &= \mathbf{AExp}(b)
\end{aligned}
$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Flow equations.: $\mathsf{VB}^{=}$

split into

- intra-block equations, using kill/generate
- inter-block equations, using *flow*

however: everything works backwards now

Flow equations: VB

$$\mathsf{VB}_{exit}(l) \quad =$$

$$\mathsf{VB}_{entry}(l) \quad =$$

where $B^l \in blocks(S_*)$

Flow equations.: $\mathsf{VB}^{=}$

split into

- intra-block equations, using kill/generate
- inter-block equations, using *flow*

however: everything works backwards now

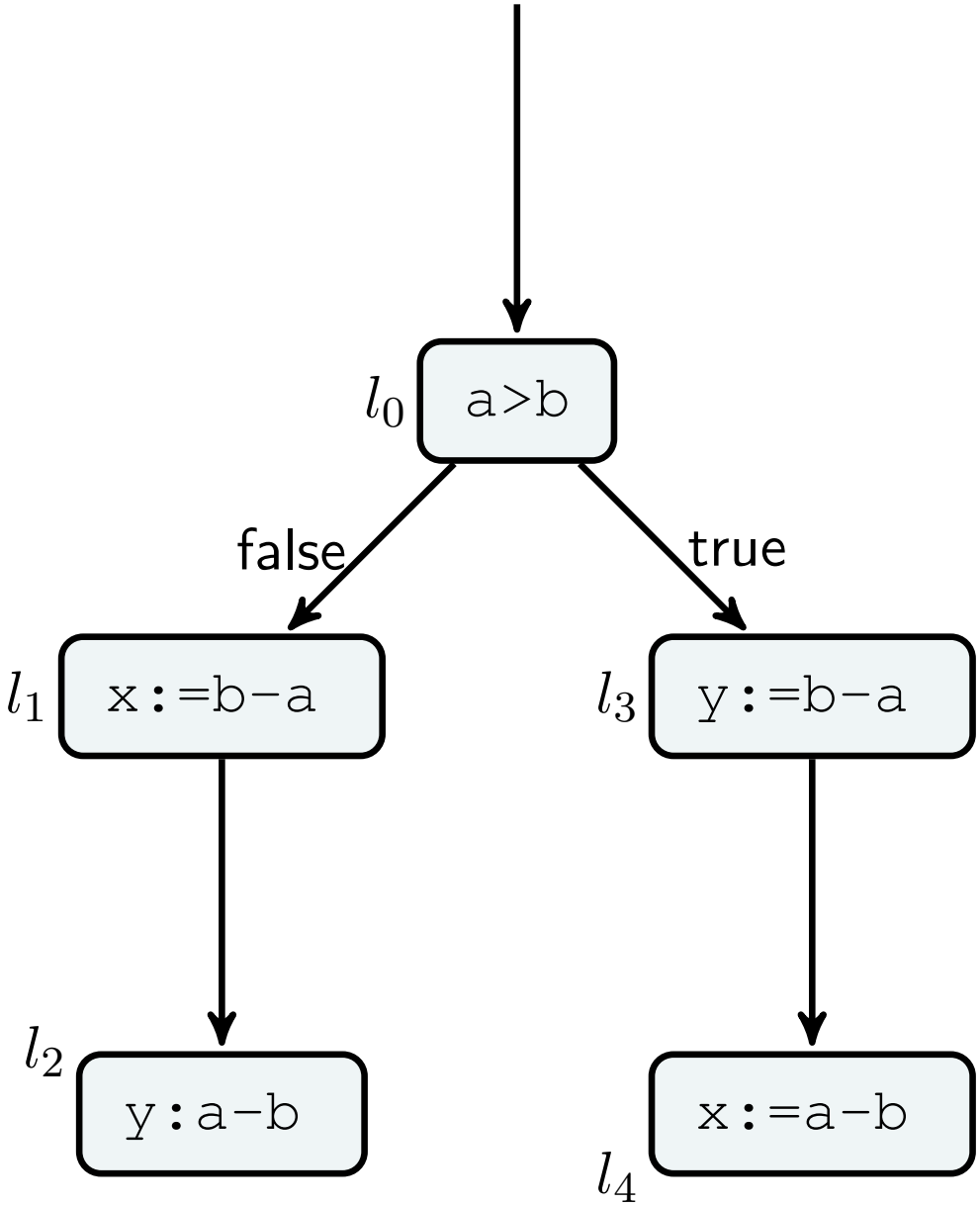
Flow equations: VB

$$\mathsf{VB}_{exit}(l) = \begin{cases} \emptyset & l \in final(S_*) \\ \bigcap\{\mathsf{VB}_{entry}(l') \mid (l', l) \in flow^R(S_*)\} & \text{otherwise} \end{cases}$$

$$\mathsf{VB}_{entry}(l) = \mathsf{VB}_{exit}(l) \setminus kill_{\mathsf{VB}}(B^l) \cup gen_{\mathsf{VB}}(B^l)$$

where $B^l \in blocks(S_*)$

Example



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

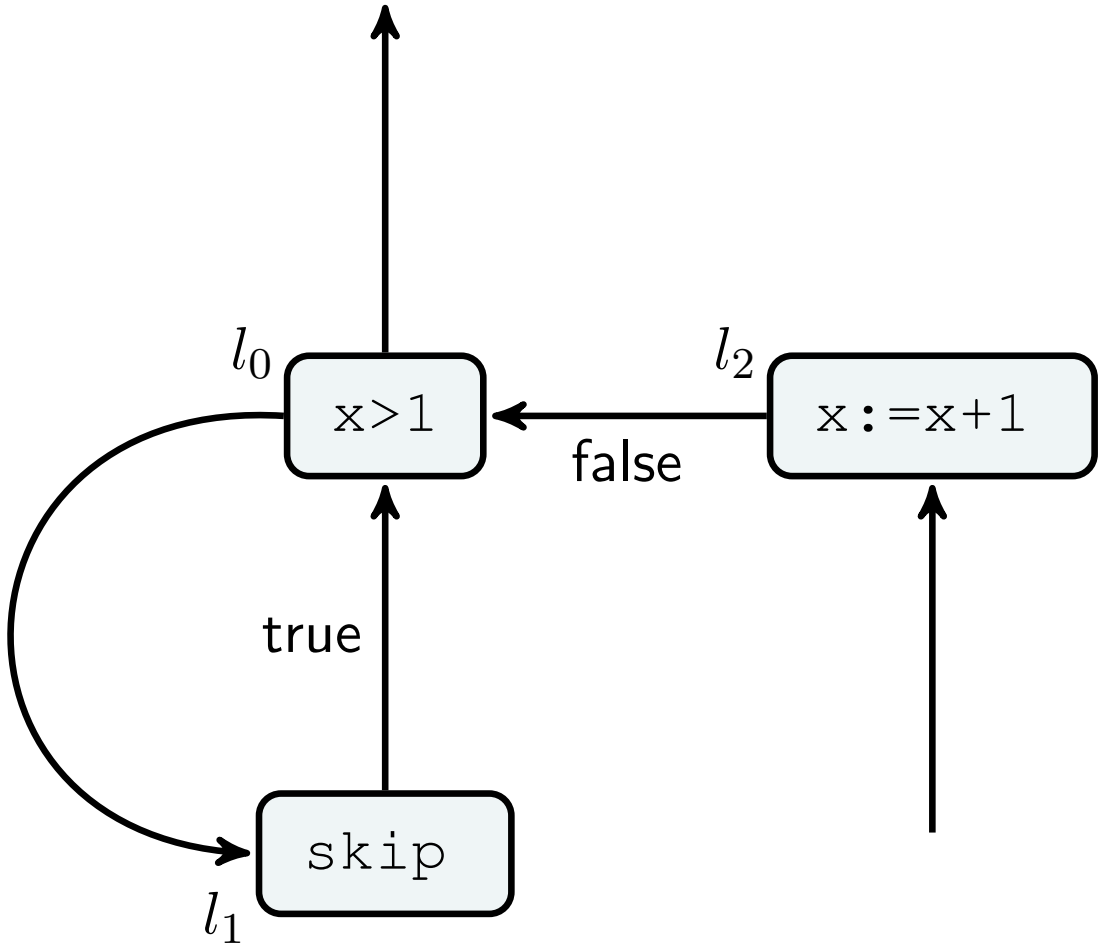
Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Example: infinite loop





Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

When can var's be "recycled": Live variable analysis

$$[x := 2]^0; [y := 4]^1; [x := 1]^2;$$
$$(\texttt{if}\, [y > x]^3\, \texttt{then}\, [z := y]^4\, \texttt{else}\, [z := y * y]^5); [x := z]^6$$

Goal therefore

for each program point: which variables may be live at the exit of that point.

- usage: *register allocation*



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

When can var's be "recycled": Live variable analysis

$$[x := 2]^0; [y := 4]^1; [x := 1]^2;$$
$$(\texttt{if}\, [y > x]^3\, \texttt{then}\, [z := y]^4\, \texttt{else}\, [z := y * y]^5); [x := z]^6$$

Live variable

A variable is live (at the exit of a label) if there *exists* a path from the mentioned exit to the *use* of that variable which does not assign to the variable (i.e., redefines its value)

Goal therefore

for each program point: which variables may be live at the exit of that point.

- usage: *register allocation*



Targets & Outline

Intraprocedural analysis
- Determining the control flow graph
- Available expressions
- Reaching definitions
- Very busy expressions
- Live variable analysis

Theoretical properties and semantics
- Semantics
- Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis
- Introduction
- Extending the semantics and the CFGs

Live variables: types

- interested in sets of variables $2^{\mathbf{Var}_*}$
- generation and killing:

$$kill_{\mathsf{LV}}, gen_{\mathsf{LV}} : \mathbf{Blocks}_* \to 2^{\mathbf{Var}_*}$$

- analysis: pair of functions

$$\mathsf{LV}_{entry}, \mathsf{LV}_{exit} : \mathbf{Lab}_* \to 2^{\mathbf{Var}_*}$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Live variables: kill and generate

$$
\begin{aligned}
kill_{\mathsf{AE}}([x := a]^l) &= \\
kill_{\mathsf{LV}}([\mathsf{skip}]^l) &= \\
kill_{\mathsf{LV}}([b]^l) &= \\
\\
gen_{\mathsf{LV}}([x := a]^l) &= \\
gen_{\mathsf{LV}}([\mathsf{skip}]^l) &= \\
gen_{\mathsf{LV}}([b]^l) &=
\end{aligned}
$$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Live variables: kill and generate

$$
\begin{aligned}
kill_{\mathsf{AE}}([x := a]^l) &= \{x\} \\
kill_{\mathsf{LV}}([\mathsf{skip}]^l) &= \emptyset \\
kill_{\mathsf{LV}}([b]^l) &= \emptyset
\end{aligned}
$$

$$
\begin{aligned}
gen_{\mathsf{LV}}([x := a]^l) &= fv(a) \\
gen_{\mathsf{LV}}([\mathsf{skip}]^l) &= \emptyset \\
gen_{\mathsf{LV}}([b]^l) &= fv(b)
\end{aligned}
$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Flow equations $\mathsf{LV}^{=}$

split into

- *intra*-block equations, using kill/generate
- inter-block equations, using flow

however: everything works backwards now

Flow equations LV

$$\mathsf{LV}_{exit}(l) \quad =$$

$$\mathsf{LV}_{entry}(l) \quad =$$

where $B^l \in blocks(S_*)$

Flow equations $\mathsf{LV}^=$

split into

- *intra*-block equations, using kill/generate
- inter-block equations, using flow

however: everything works backwards now

Flow equations LV

$$\mathsf{LV}_{exit}(l) = \begin{cases} \emptyset & l \in final(S_*) \\ \bigcup\{\mathsf{LV}_{entry}(l') \mid (l', l) \in flow^R(S_*)\} & \text{otherwise} \end{cases}$$

$$\mathsf{LV}_{entry}(l) = \mathsf{LV}_{exit}(l) \setminus kill_{\mathsf{LV}}(B^l) \cup gen_{\mathsf{LV}}(B^l)$$

where $B^l \in blocks(S_*)$

Example

$$(\texttt{while}\ [x > 1]^{l_0}\ \texttt{do}\ [\mathsf{skip}]^{l_1}); [x := x + 1]^{l_2}$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

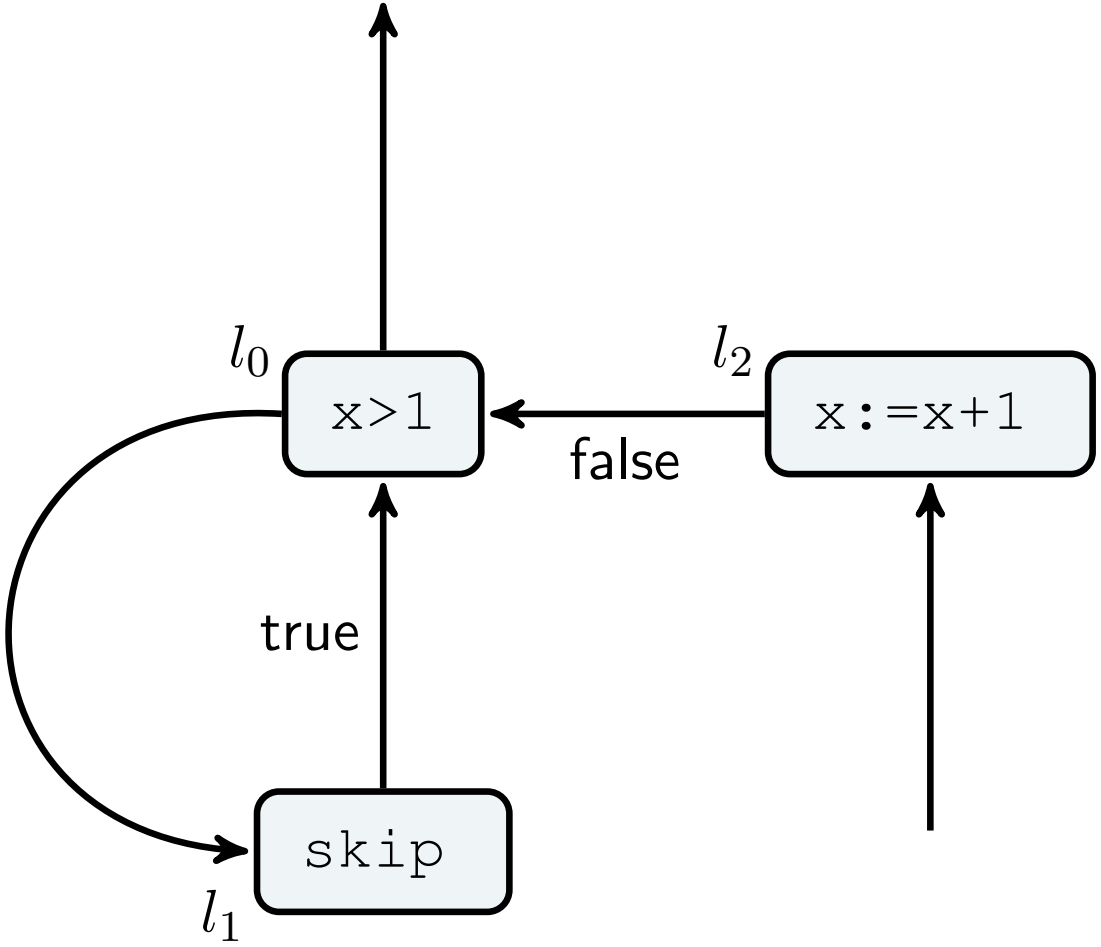
Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Looping example





Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs



Section

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Chapter 2 “Data flow analysis”
Course “Static analysis and all that”
Martin Steffen
IN5440 / autumn 2022

Relating programs with analyses

- analyses
  - intended as (static) *abstraction* or overapprox. of real program behavior
  - so far: *without real connection* to programs
- soundness of the analysis: safe analysis
- but: *behavior* or *semantics* of programs not yet defined
- here: "easiest" semantics: *operational*
- more precisely: *small-step SOS* (structural operational semantics)



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

States, configs, and transitions

fixing some data types

- *state* $\sigma : \mathbf{State} = \mathbf{Var} \to \mathbf{Z}$
- *configuration:* pair of *statement* $\times$ *state* or (terminal) just a *state*

Transitions

$$\langle S, \sigma \rangle \to \acute{\sigma} \quad \text{or} \quad \langle S, \sigma \rangle \to \langle \acute{S}, \acute{\sigma} \rangle$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Semantics of expressions

$$[\ _\ ]^{\mathcal{A}}_{_} : \mathbf{AExp} \to (\mathbf{State} \to \mathbb{Z})$$
$$[\ _\ ]^{\mathcal{B}}_{_} : \mathbf{BExp} \to (\mathbf{State} \to \mathbf{B})$$

simplifying assumption: no errors

$$\begin{aligned}
[\![x]\!]^{\mathcal{A}}_{\sigma} &= \sigma(x) \\
[\![n]\!]^{\mathcal{A}}_{\sigma} &= \mathcal{N}(n) \\
[\![a_1\ \mathrm{op}_a\ a_2]\!]^{\mathcal{A}}_{\sigma} &= [\![a_1]\!]^{\mathcal{A}}_{\sigma}\ \mathbf{op}_a\ [\![a_2]\!]^{\mathcal{A}}_{\sigma}
\end{aligned}$$

$$\begin{aligned}
[\![\mathrm{not}\ b]\!]^{\mathcal{B}}_{\sigma} &= \neg [\![b]\!]^{\mathcal{B}}_{\sigma} \\
[\![b_1\ \mathrm{op}_b\ b_2]\!]^{\mathcal{B}}_{\sigma} &= [\![b_1]\!]^{\mathcal{B}}_{\sigma}\ \mathbf{op}_b\ [\![b_2]\!]^{\mathcal{B}}_{\sigma} \\
[\![a_1\ \mathrm{op}_r\ a_2]\!]^{\mathcal{B}}_{\sigma} &= [\![a_1]\!]^{\mathcal{A}}_{\sigma}\ \mathbf{op}_r\ [\![a_2]\!]^{\mathcal{A}}_{\sigma}
\end{aligned}$$

clearly:

$$\forall x \in fv(a).\ \sigma_1(x) = \sigma_2(x) \text{ then } [\![a]\!]^{\mathcal{A}}_{\sigma_1} = [\![a]\!]^{\mathcal{A}}_{\sigma_2}$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

SOS

$$\langle [x := a]^l, \sigma\rangle \to \sigma[x \mapsto [\![a]\!]^{\mathcal{A}}_{\sigma}] \quad \textsc{Ass} \qquad \langle [\mathsf{skip}]^l, \sigma\rangle \to \sigma \quad \textsc{Skip}$$

$$\frac{\langle S_1, \sigma\rangle \to \langle \acute{S}_1, \acute{\sigma}\rangle}{\langle S_1; S_2, \sigma\rangle \to \langle \acute{S}_1; S_2, \acute{\sigma}\rangle}\ \textsc{Seq}_1 \qquad \frac{\langle S_1, \sigma\rangle \to \acute{\sigma}}{\langle S_1; S_2, \sigma\rangle \to \langle S_2, \acute{\sigma}\rangle}\ \textsc{Seq}_2$$

$$\frac{[\![b]\!]^{\mathcal{B}}_{\sigma} = \top}{\langle \texttt{if } [b]^l \texttt{ then } S_1 \texttt{ else } S_2, \sigma\rangle \to \langle S_1, \sigma\rangle}\ \textsc{If}_1$$

$$\frac{[\![b]\!]^{\mathcal{B}}_{\sigma} = \top}{\langle \texttt{while } [b]^l \texttt{ do } S, \sigma\rangle \to \langle S; \texttt{while} [b]^l \texttt{ do } S, \sigma\rangle}\ \textsc{While}_1$$

$$\frac{[\![b]\!]^{\mathcal{B}}_{\sigma} = \bot}{\langle \texttt{while } [b]^l \texttt{ do } S, \sigma\rangle \to \sigma}\ \textsc{While}_2$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Derivation sequences

- derivation sequence: "completed" execution:
  - finite sequence: $\langle S_1, \sigma_1 \rangle, \ldots, \langle S_n, \sigma_n \rangle, \sigma_{n+1}$
  - infinite sequence: $\langle S_1, \sigma_1 \rangle, \ldots, \langle S_i, \sigma_i \rangle, \ldots$
- note: labels do *not* influence the semantics
- CFG for the "rest" of the program only gets "smaller" when running:

Lemma (Steps and CFG)

1. $\langle S, \sigma \rangle \to \sigma'$, *then* $final(S) = \{init(S)\}$
2. *Assume* $\langle S, \sigma \rangle \to \langle \acute{S}, \acute{\sigma} \rangle$, *then*
   - 2.1 $final(S) \supseteq \{final(\acute{S})\}$
   - 2.2 $flow(S) \supseteq \{flow(\acute{S})\}$
   - 2.3 $blocks(S) \supseteq blocks(\acute{S})$; *if* S *is label consistent, then so is* $\acute{S}$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Correctness of live analysis

- LV as example
- given as *constraint system* (not as equational system)

LV constraint system

$$\mathsf{LV}_{exit}(l) \supseteq \begin{cases} \emptyset & l \in final(S_*) \\ \bigcup\{\mathsf{LV}_{entry}(l') \mid (l', l) \in flow^R(S_*)\} & \text{otherwise} \end{cases}$$

$$\mathsf{LV}_{entry}(l) \supseteq \mathsf{LV}_{exit}(l) \setminus kill_{\mathsf{LV}}(B^l) \cup gen_{\mathsf{LV}}(B^l)$$

$$live_{entry}, live_{exit} : \mathbf{Lab}_* \to 2^{\mathbf{Var}_*}$$

"$live$ solves constraint system $\mathsf{LV}^{\subseteq}(S)$"

$$live \models \mathsf{LV}^{\subseteq}(S)$$

(analogously for equations $\mathsf{LV}^{=}(S)$)

Equational vs. constraint analysis

Lemma (Equational and inclusion constraints)

1. *If* $live \models \mathsf{LV}^{=}$, *then* $live \models \mathsf{LV}^{\subseteq}$
2. *The least solutions of* $live \models \mathsf{LV}^{=}$ *and* $live \models \mathsf{LV}^{\subseteq}$ *coincide.*



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Intermezzo: orders, lattices etc.

as a reminder:

- partial order $(L, \sqsubseteq)$
- *upper bound* l of $Y \subseteq L$:
- *least* upper bound (lub): $\bigsqcup Y$ (or *join*)
- dually: lower bounds and greatest lower bounds: $\bigsqcap Y$ (or *meet*)
- complete lattice $L = (L, \sqsubseteq) = (L, \sqsubseteq, \bigsqcap, \bigsqcup, \bot, \top)$: a partially ordered set where meets and joins exist for *all subsets*, furthermore $\top = \bigsqcap \emptyset$ and $\bot = \bigsqcup \emptyset$.



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Fixpoints

given: complete lattice L and monotone $f : L \to L$.

- fixpoint: $f(a) = a$

$$Fix(a) = \{a \mid f(a) = a\}$$

- f *reductive* at a, a is a pre-fixpoint of f: $f(a) \sqsubseteq a$:

$$Red(f) = \{a \mid f(a) \sqsubseteq a\}$$

- f *extensive* at a, a is a post-fixpoint of f: $f(a) \sqsupseteq a$:

$$Ext(f) = \{a \mid f(a) \sqsupseteq a\}$$

Define "lfp" / "gfp"

$$lfp(f) \triangleq \bigsqcap Fix(f) \quad \text{and} \quad gfp(f) \triangleq \bigsqcup Fix(f) \qquad (1)$$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Tarski's theorem

Core

Perhaps core insight of the whole lattice/fixpoint business: not only does the $\bigsqcap$ of all pre-fixpoints uniquely exist (that's what the lattice is for), but —and that's the trick— *it's a pre-fixpoint* itself (ultimately due to montonicity of f).

Theorem

L: complete lattice, $f : L \to L$ monotone.

$$\begin{array}{lcll} lfp(f) & \triangleq & \bigsqcap Red(f) & \in \quad Fix(f) \\ gfp(f) & \triangleq & \bigsqcup Ext(f) & \in \quad Fix(f) \end{array} \tag{2}$$

- Note: lfp (despite the name) is *defined* as glb of all pre-fixpoints
- The theorem (more or less directly) implies lfp is the *least* fixpoint



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Fixpoint iteration

- often: iterate, approximate least fixed point from below $(f^n(\bot))_n$:

$$\bot \sqsubseteq f(\bot) \sqsubseteq f^2(\bot) \sqsubseteq \dots$$

- not assured that we "reach" the fixpoint ("within" ω)

$$\bot \sqsubseteq f^n(\bot) \sqsubseteq \bigsqcup_n f^n(\bot) \quad \sqsubseteq \quad lfp(f)$$
$$gfp(f) \quad \sqsubseteq \bigsqcap_n f^n(\top) \sqsubseteq f^n(\top) \sqsubseteq (\top)$$

- additional requirement: continuity on f for all ascending chains $(l_n)_n$

$$f(\bigsqcup_n (l_n)) = \bigsqcup (f(l_n))$$

- *ascending chain condition* ("stabilization"): $f^n(\bot) = f^{n+1}(\bot)$, i.e., $lfp(f) = f^n(\bot)$
- *descending* chain condition: dually

Basic preservation results

Lemma ("Smaller" graph $\rightarrow$ less constraints)

Assume $live \models \mathsf{LV}^{\subseteq}(S_1)$. *If* $flow(S_1) \supseteq flow(S_2)$ *and* $blocks(S_1) \supseteq blocks(S_2)$, *then* $live \models \mathsf{LV}^{\subseteq}(S_2)$.

Corollary ("subject reduction")

If $live \models \mathsf{LV}^{\subseteq}(S)$ *and* $\langle S, \sigma\rangle \rightarrow \langle \acute{S}, \acute{\sigma}\rangle$, *then* $live \models \mathsf{LV}^{\subseteq}(\acute{S})$

Lemma (Inter-block flow)

Assume $live \models \mathsf{LV}^{\subseteq}(S)$. *If* $l \rightarrow_{flow} l'$, *then* $live_{exit}(l) \supseteq live_{entry}(l')$.



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Correctness relation

- basic intuitition: only live variables influence the program
- proof by *induction*

$\Rightarrow$

Correctness relation on states:

Given V = set of variables:

$$\sigma_1 \sim_V \sigma_2 \text{ iff } \forall x \in V.\sigma_1(x) = \sigma_2(x) \tag{3}$$

$$\begin{array}{ccccccccc}
\langle S, \sigma_1 \rangle & \longrightarrow & \langle S', \sigma_1' \rangle & \longrightarrow & \ldots & \longrightarrow & \langle S'', \sigma_1'' \rangle & \longrightarrow & \sigma_1''' \\
\Big| \sim_V & & \Big| \sim_{V'} & & & & \Big| \sim_{V''} & & \Big| \sim_{X(l)} \\
\langle S, \sigma_2 \rangle & \longrightarrow & \langle S', \sigma_2' \rangle & \longrightarrow & \ldots & \longrightarrow & \langle S'', \sigma_2'' \rangle & \longrightarrow & \sigma_2'''
\end{array}$$

Notation: $N(l) = live_{entry}(l)$, $X(l) = live_{exit}(l)$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Correctness (1)

Lemma (Preservation inter-block flow)

Assume $live \models \mathsf{LV}^{\subseteq}$. *If* $\sigma_1 \sim_{X(l)} \sigma_2$ *and* $l \rightarrow_{flow} l'$, *then* $\sigma_1 \sim_{N(l')} \sigma_2$.



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Correctness

Theorem (Correctness)

Assume $live \models \mathsf{LV}^{\subseteq}(S)$.

- *If* $\langle S, \sigma_1\rangle \to \langle \acute{S}, \acute{\sigma}_1\rangle$ *and* $\sigma_1 \sim_{N(init(S))} \sigma_2$, *then there exists* $\acute{\sigma}_2$ *s.t.* $\langle S, \sigma_2\rangle \to \langle \acute{S}, \acute{\sigma}_2\rangle$ *and* $\acute{\sigma}_1 \sim_{N(init(\acute{S}))} \acute{\sigma}_2$.
- *If* $\langle S, \sigma_1\rangle \to \acute{\sigma}_1$ *and* $\sigma_1 \sim_{N(init(S))} \sigma_2$, *then there exists* $\acute{\sigma}_2$ *s.t.* $\langle S, \sigma_2\rangle \to \acute{\sigma}_2$ *and* $\acute{\sigma}_1 \sim_{X(init(S))} \acute{\sigma}_2$.

$$\begin{array}{ccc} \langle S, \sigma_1\rangle & \longrightarrow & \langle \acute{S}, \acute{\sigma}_1\rangle \\ \Big| \sim_{N(init(S))} & & \vdots \sim_{N(init(\acute{S}))} \\ \langle S, \sigma_2\rangle & \cdots\cdots\to & \langle \acute{S}, \acute{\sigma}_2\rangle \end{array} \qquad \begin{array}{ccc} \langle S, \sigma_1\rangle & \longrightarrow & \acute{\sigma}_1 \\ \Big| \sim_{N(init(S))} & & \vdots \sim_{X(init(S))} \\ \langle S, \sigma_2\rangle & \cdots\cdots\to & \acute{\sigma}_2 \end{array}$$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Correctness (many steps)

Assume $live \models \mathsf{LV}^{\subseteq}(S)$

- If $\langle S, \sigma_1 \rangle \rightarrow^* \langle \acute{S}, \acute{\sigma}_1 \rangle$ and $\sigma_1 \sim_{N(init(S))} \sigma_2$, then there exists $\acute{\sigma}_2$ s.t. $\langle S, \sigma_2 \rangle \rightarrow^* \langle \acute{S}, \acute{\sigma}_2 \rangle$ and $\acute{\sigma}_1 \sim_{N(init(\acute{S}))} \acute{\sigma}_2$.
- If $\langle S, \sigma_1 \rangle \rightarrow^* \acute{\sigma}_1$ and $\sigma_1 \sim_{N(init(S))} \sigma_2$, then there exists $\acute{\sigma}_2$ s.t. $\langle S, \sigma_2 \rangle \rightarrow^* \acute{\sigma}_2$ and $\acute{\sigma}_1 \sim_{X(l)} \acute{\sigma}_2$ for some $l \in \mathit{final}(S)$.



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs



Section

Monotone frameworks

Chapter 2 “Data flow analysis”
Course “Static analysis and all that”
Martin Steffen
IN5440 / autumn 2022

Monotone framework: general pattern

$$
\begin{aligned}
Analysis_\circ(l) &= \begin{cases} \iota & \text{if } l \in E \\ \bigsqcup\{Analysis_\bullet(l') \mid (l', l) \in F\} & \text{otherwise} \end{cases} \\
Analysis_\bullet(l) &= f_l(Analysis_\circ(l))
\end{aligned}
\tag{4}
$$

- $\bigsqcup$: either $\bigcup$ or $\bigcap$
- F: either $flow(S_*)$ or $flow^R(S_*)$.
- E: either $\{init(S_*)\}$ or $final(S_*)$
- ι: either the initial or final information
- f_l: transfer function for $[B]^l \in blocks(S_*)$.



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Monotone frameworks

direction of flow:

- forward analysis:
  - $F = \mathit{flow}(S_*)$
  - $Analysis_\circ$ for entry and $Analysis_\bullet$ for exits
  - assumption: isolated entries
- backward analysis: dually
  - $F = \mathit{flow}^R(S_*)$
  - $Analysis_\circ$ for exit and $Analysis_\bullet$ for entry
  - assumption: isolated exits

sort of solution

- may analysis
  - properties for *some* path
  - *smallest* solution
- must analysis
  - properties of /all paths
  - *greatest* solution



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Basic definitions: property space

- *property space* L, often *complete lattice*
- *combination* operator: $\bigsqcup : 2^L \to L$, $\sqcup$: binary case
- $\bot = \bigsqcup \emptyset$
- often: ascending chain condition (stabilization)



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Transfer functions

$$f_l : L \to L$$

with $l \in \mathbf{Lab}_*$

- associated with the *blocks*
- requirement: *monotone*
- $\mathcal{F}$: monotone functions over L:
  - containing all *transfer functions*
  - containing *identity*
  - *closed under composition*



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Summary

- complete lattice L, ascending chain condition
- $\mathcal{F}$ monotone functions, closed as stated
- distributive framework

$$f(l_1 \sqcup l_2) = f(l_1) \sqcup f(l_2)$$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

The 4 classical examples

- for a label consistent program S_*, all are *instances* of a monotone, distributive, framework:
- conditions:
  - lattice of properties: immediate (subset/superset)
  - ascending chain condition: *finite* set of syntactic entities
  - *closure* conditions on $\mathcal{F}$
    - monotone
    - closure under identity and composition
  - *distributivity*: assured by using the kill- and generate-formulation



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Overview over the 4 examples

	avail. epxr.	reach. def's	very busy expr.	live var's
L	$2^{\mathbf{AExp}_*}$	$2^{\mathbf{Var}_* \times \mathbf{Lab}_*^?}$	$2^{\mathbf{AExp}_*}$	$2^{\mathbf{Var}_*}$
$\sqsubseteq$	$\supseteq$	$\subseteq$	$\supseteq$	$\subseteq$
$\bigsqcup$	$\bigcap$	$\bigcup$	$\bigcap$	$\bigcup$
$\bot$	$\mathbf{AExp}_*$	$\emptyset$	$\mathbf{AExp}_*$	$\emptyset$
ι	$\emptyset$	$\{(x, ?) \mid x \in fv(S_*)\}$	$\emptyset$	$\emptyset$
E	$\{init(S_*)\}$	$\{init(S_*)\}$	$final(S_*)$	$final(S_*)$
F	$flow(S_*)$	$flow(S_*)$	$flow^R(S_*)$	$flow^R(S_*)$
$\mathcal{F}$	$\{f : L \to L \mid \exists l_k, l_g.\ f(l) = (l \setminus l_k) \cup l_g\}$			
f_l	$f_l(l) = (l \setminus kill([B]^l) \cup gen([B]^l))$ where $[B]^l \in blocks(S_*)$			



Section

Equation solving

Chapter 2 “Data flow analysis”
Course “Static analysis and all that”
Martin Steffen
IN5440 / autumn 2022

Solving the analyses

- given: set of equations (or constraints) over finite sets of variables
- domain of variables: complete lattices + ascending chain condition
- *2 solutions* for the monotone frameworks
  - MFP: "maximal fix point"
  - MOP: "meet over all paths"



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

MFP

- terminology: historically "MFP" stands for *maximal* fix point (not minimal)
- iterative worklist algorithm:
  - central data structure: *worklist*
  - list (or container/set) of pairs
- related to *chaotic iteration*



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Chaotic iteration

Input: equations for reaching defs for the given program

Output: least solution: $\vec{\mathsf{RD}} = (\mathsf{RD}_1, \ldots, \mathsf{RD}_{12})$

Initialization:

$$\mathsf{RD}_1 := \emptyset; \ldots; \mathsf{RD}_{12} := \emptyset$$

Iteration:

while $\mathsf{RD}_j \neq F_j(\mathsf{RD}_1, \ldots, \mathsf{RD}_{12})$ **for some** j **do**

$$\mathsf{RD}_j := F_j(\mathsf{RD}_1, \ldots, \mathsf{RD}_{12})$$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Worklist algorithms

- *fixpoint* iteration algorithm
- general kind of algorithms, for DFA, CFA, ...
- same for *equational and /constraint* systems
- "specialization" i.e., *determinization* of chaotic iteration

$\Rightarrow$ worklist: central data structure, "container" containing "the work still to be done"

- for more details (different traversal strategies): see Chap. 6 from [1]



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

WL-algo for DFA

- WL-algo for *monotone frameworks*

$\Rightarrow$ input: instance of monotone framework

- two central data structures
  - worklist: *flow-edges* yet to be (re-)considered:
    1. *removed* when *effect* of transfer function has been taken care of
    2. *(re-)added*, when point 1 *endangers* satisfaction of (in-)equations
  - array to store the "current state" of $Analysis_\circ$
- one central *control structure* (after *initialization*): loop until worklist empty



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

```
Input:  (L, ℱ, F, E, ι, f)
Output: MFP∘, MFP•
Method: step 1: initialization
                W := nil;
                for all (l, l') ∈ F do W := (l, l') :: W;
                for all l ∈ F or ∈ E do
                   if l ∈ E then Analysis[l] := ι
                            else Analysis[l] := ⊥L;
        step 2: iteration
                while W ≠ nil do
                  (l, l') := ( fst(head(W)), snd(head(W)));
                  W := tail W;
                  if fl(Analysis[l]) ⋢ Analysis[l']
                  then  Analysis[l'] := Analysis[l'] ⊔ fl(Analysis[l]);
                        for all l'' with (l', l'') ∈ F do
                              W := (l', l'') :: W;
         step 3: presenting the result:
                  for all l ∈ F or ∈ E do
                    MFP∘(l) := Analysis[l];
                    MFP•(l) := fl(Analysis[l])
```

ML Code

```
  let rec solve (wl1 : edge list) : unit =
    match wl1 with
    |    [] -> ()                                          (* wl done *)
    |    (l,l')::wl' ->
         let ana_pre   : var list = lookx (ana,l) (* extract ``states *)
         and  ana_post : var list = lookx (ana,l')
         in let ana_exitpre : var list = f_trans(ana_pre,l)
         in
         if not (subset (ana_exitpre,ana_post))
         then
           (enter (ana,l',union(ana_post,ana_exitpre));
            let (new_edges : edge list) =
              (let (preds : node list) = Flow.Graph.pred (l')
              in List.map (fun n -> (l',n)) preds)
            in solve (new_edges @ wl')
           )
         else                                    (* Nothing to do here. *)
           (solve (wl'))
  in
  solve wl_init;
  fun (x: node) -> lookx (ana, x)
;;
```



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

MFP: properties

Lemma

The algo

- *terminates and*
- *calculates the least solution*

Proof.

- termination: ascending chain condition & loop is enlarging
- least FP:
  - invariant: array always below $Analysis_\circ$
  - at loop exit: array "solves" (in-)equations

□



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Time complexity

- estimation of *upper bound* of number basic steps
  - at most b different labels in E
  - at most $e \geq b$ pairs in the flow F
  - height of the lattice: at most h
  - non-loop steps: $O(b + e)$
  - *loop*: at most h times addition to the WL

$\Rightarrow$

$$O(e \cdot h) \tag{5}$$

or $\leq O(b^2 h)$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs



Section

Interprocedural analysis

- Introduction
- Extending the semantics and the CFGs
- Naive analysis (non-context-sensitive)
- Taking paths into account
- Context-sensitive analysis

Chapter 2 “Data flow analysis”
Course “Static analysis and all that”
Martin Steffen
IN5440 / autumn 2022

Adding procedures

- so far: *very simplified* language:
  - minimalistic imperative language
  - reading and writing to variables plus
  - simple control flow, given as flow graph
- now: *procedures*: interprocedural analysis
- complications:
  - calls/return (control flow)
  - parameter passing
  - scopes
  - higher-order functions/procedures
- here: top-level procedures, mutual recursion, *call-by-value*
  - *call-by-result*



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Extending the syntax

- $\texttt{begin}\, D_*\ S_*\, \texttt{end}$

$$D ::= \texttt{proc}\, p(\texttt{val}\, x, \texttt{res}\, y)\texttt{is}^{l_n}\, S\, \texttt{end}^{l_x} \mid D\ D$$

- procedure names p
- statements

$$S ::= \ldots [\texttt{call}\, p(a, z)]^{l_c}_{l_r}$$

- note: call statement with *2 labels*
- *statically scoped* language, CBV parameter passing (1st parameter), and CBN for second
- mutual recursion possible
- assumption: unique labelling, only declared procedures are called, all procedures have different names.



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Example: Fibonacci

$$
\begin{array}{ll}
\texttt{begin} & \texttt{proc}\, \mathit{fib}(\texttt{val}\, z, u, \texttt{res}\, v)\texttt{is}^{1} \\
& \quad \texttt{if} \quad [z < 3]^{2} \\
& \quad \texttt{then} \quad [v := u + 1]^{3} \\
& \quad \texttt{else} \quad [\texttt{call}\, \mathit{fib}(z - 1, u, v)]^{4}_{5}; \\
& \qquad\qquad\quad [\texttt{call}\, \mathit{fib}(z - 2, v, v)]^{6}_{7} \\
& \texttt{end}^{8}; \\
& [\texttt{call}\, \mathit{fib}(x, 0, y)]^{9}_{10} \\
\texttt{end} &
\end{array}
$$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Block, labels, etc.

$$
\begin{aligned}
init([\texttt{call}\, p(a,z)]^{l_c}_{l_r}) &= l_c \\
final([\texttt{call}\, p(a,z)]^{l_c}_{l_r}) &= \{l_r\} \\
blocks([\texttt{call}\, p(a,z)]^{l_c}_{l_r}) &= \{[\texttt{call}\, p(a,z)]^{l_c}_{l_r}\} \\
labels([\texttt{call}\, p(a,z)]^{l_c}_{l_r}) &= \{l_c, l_r\} \\
flow([\texttt{call}\, p(a,z)]^{l_c}_{l_r}) &=
\end{aligned}
$$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Block, labels, etc.

$$
\begin{aligned}
init([\texttt{call}\, p(a,z)]^{l_c}_{l_r}) &= l_c \\
final([\texttt{call}\, p(a,z)]^{l_c}_{l_r}) &= \{l_r\} \\
blocks([\texttt{call}\, p(a,z)]^{l_c}_{l_r}) &= \{[\texttt{call}\, p(a,z)]^{l_c}_{l_r}\} \\
labels([\texttt{call}\, p(a,z)]^{l_c}_{l_r}) &= \{l_c, l_r\} \\
flow([\texttt{call}\, p(a,z)]^{l_c}_{l_r}) &= \{(\mathbf{l_c}; \mathbf{l_n}), (\mathbf{l_x}; \mathbf{l_r})\}
\end{aligned}
$$

where $\texttt{proc}\, p(\texttt{val}\, x, \texttt{res}\, y)\texttt{is}^{l_n} S \texttt{end}^{l_x}$ is in D_*.

- two *new* kinds of flows (written slightly different(!)): *calling* and *returning*
- *static* dispatch only



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

For procedure declarations

$$
\begin{aligned}
init(p) &= \\
final(p) &= \\
blocks(p) &= \cup blocks(S) \\
labels(p) &= \\
flow(p) &=
\end{aligned}
$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

For procedure declarations

$$
\begin{aligned}
init(p) &= l_n \\
final(p) &= \{l_x\} \\
blocks(p) &= \{\mathtt{is}^{l_n}, \mathtt{end}^{l_x}\} \cup blocks(S) \\
labels(p) &= \{l_n, l_x\} \cup labels(S) \\
flow(p) &= \{(l_n, init(S))\} \cup flow(S) \cup \{(l, l_x) \mid l \in final(S)\}
\end{aligned}
$$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

"Standard" flow of complete program

not yet interprocedural flow (IF)

$$
\begin{array}{rcl}
init_* & = & init(S_*) \\
final_* & = & final(S_*) \\
blocks_* & = & \bigcup\{blocks(p) \mid \texttt{proc}\, p(\texttt{val}\, x, \texttt{res}\, y)\texttt{is}^{l_n}\, S\, \texttt{end}^{l_x} \in D_*\} \\
 & & \cup blocks(S_*) \\
labels_* & = & \bigcup\{labels(p) \mid \texttt{proc}\, p(\texttt{val}\, x, \texttt{res}\, y)\texttt{is}^{l_n}\, S\, \texttt{end}^{l_x} \in D_*\} \\
 & & \cup labels(S_*) \\
flow_* & = & \bigcup\{flow(p) \mid \texttt{proc}\, p(\texttt{val}\, x, \texttt{res}\, y)\texttt{is}^{l_n}\, S\, \texttt{end}^{l_x} \in D_*\} \\
 & & \cup flow(S_*)
\end{array}
$$

As before: S_*: notation for complete program "of interest"

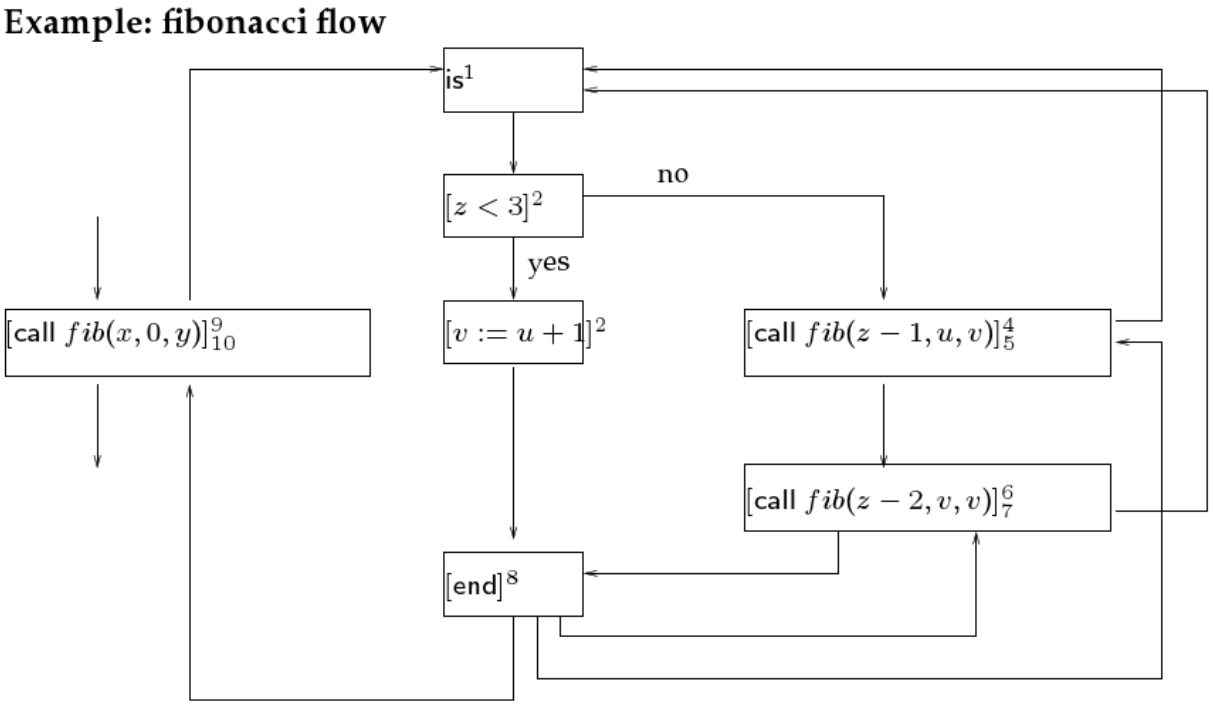
New kind of edges: Interprocedural flow (IF)

- inter-procedural: from call-site to procedure, and back: $(l_c; l_n)$ and $(l_x; l_r)$.
- more *precise* (= better) capture of flow
- abbreviation: IF for $inter\text{-}flow_*$ or $inter\text{-}flow_*^R$

IF

$$inter\text{-}flow_* = \{(l_c, l_n, l_x, l_r) \mid P_* \text{ contains } \begin{array}{l}[\texttt{call}\, p(a, z)]^{l_c}_{l_r} \text{ and} \\ \texttt{proc}(\texttt{val}\, x, \texttt{res}\, y)\texttt{is}^{l_n}\, S\, \texttt{end}^{l_x}\end{array} \}$$

Example: Fibonacci flow



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

- Determining the control flow graph
- Available expressions
- Reaching definitions
- Very busy expressions
- Live variable analysis

Theoretical properties and semantics

- Semantics
- Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

- Introduction
- Extending the semantics and the CFGs

Semantics: stores, locations,...

- not only new *syntax*
- new semantical concept: local data!
  - different "incarnations" of a variable $\Rightarrow$ *locations*
  - remember: $\sigma \in \mathbf{State} = \mathbf{Var}_* \to \mathbb{Z}$

Representation of "memory"

$$
\begin{array}{lcll}
\xi & \in & \mathbf{Loc} & \text{locations} \\
\rho & \in & \mathbf{Env} = \mathbf{Var}_* \to \mathbf{Loc} & \text{environment} \\
\varsigma & \in & \mathbf{Store} = \mathbf{Loc} \to_{fin} \mathbb{Z} & \text{store}
\end{array}
$$

- $\sigma = \varsigma \circ \rho$: total $\Rightarrow$ $ran(\rho) \subseteq dom(\varsigma)$
- top-level environment: ρ_*: all var's are mapped to unique locations (no aliasing !!!!)



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

SOS steps

- steps *relative* to *environment* ρ

$$\rho \vdash \langle S, \varsigma \rangle \rightarrow \langle \acute{S}, \acute{\varsigma} \rangle \tag{6}$$

or

$$\rho \vdash \langle S, \varsigma \rangle \rightarrow \acute{\varsigma} \tag{7}$$

- old rules needs to be adapted
- "global" environment ρ_* (for global vars)



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Call-rule

$$\frac{\begin{array}{c}\texttt{proc}\, p(\texttt{val}\, x, \texttt{res}\, y)\texttt{is}^{l_n}\ S\ \texttt{end}^{l_x} \in D_* \\ \xi_1, \xi_2 \notin dom(\varsigma) \\ \acute{\varsigma} = \end{array}}{\rho \vdash \langle [\texttt{call}\, p(a, z)]^{l_c}_{l_r}, \varsigma\rangle \rightarrow \langle \mathsf{bind}\ \rho_*[x \mapsto \xi_1][y \mapsto \xi_2]\ \texttt{in}\, S\ \texttt{then}\, z := y, \acute{\varsigma}\rangle}\ \textsc{Call}$$

Call-rule

$$\frac{\begin{array}{c}\texttt{proc}\, p(\texttt{val}\, x, \texttt{res}\, y)\texttt{is}^{l_n}\ S\ \texttt{end}^{l_x} \in D_* \\ \xi_1, \xi_2 \notin dom(\varsigma) \qquad v \in \mathbb{Z} \\ \acute{\varsigma} = \varsigma[\xi_1 \mapsto [\![a]\!]^{\mathcal{A}}_{\varsigma \circ \rho}][\xi_2 \mapsto v]\end{array}}{\rho \vdash \langle [\texttt{call}\, p(a, z)]^{l_c}_{l_r}, \varsigma\rangle \rightarrow \langle \mathsf{bind}\ \rho_*[x \mapsto \xi_1][y \mapsto \xi_2]\ \texttt{in}\, S\, \texttt{then}\, z := y, \acute{\varsigma}\rangle}\ \textsc{Call}$$

Bind-construct

$$\frac{\acute{\rho} \vdash \langle S, \varsigma \rangle \to \langle \acute{S}, \acute{\varsigma} \rangle}{\rho \vdash \langle \text{bind } \acute{\rho} \texttt{ in } S \texttt{ then } z := y, \varsigma \rangle \to} \textsc{Bind}_1$$

$$\frac{\acute{\rho} \vdash \langle S, \varsigma \rangle \to \acute{\varsigma}}{\rho \vdash \langle \text{bind } \acute{\rho} \texttt{ in } S \texttt{ then } z := y, \varsigma \rangle \to} \textsc{Bind}_2$$

- bind-syntax: "runtime syntax"

$\Rightarrow$ formulation of correctness must be adapted, too (Chap. 3)[1]

[1] Not covered in the lecture.



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Bind-construct

$$\frac{\acute{\rho} \vdash \langle S, \varsigma \rangle \to \langle \acute{S}, \acute{\varsigma} \rangle}{\rho \vdash \langle \text{bind } \acute{\rho} \texttt{ in } S \texttt{ then } z := y, \varsigma \rangle \to \langle \text{bind } \acute{\rho} \texttt{ in } \acute{S} \texttt{ then } z := y, \acute{\varsigma} \rangle} \text{BIND}_1$$

$$\frac{\acute{\rho} \vdash \langle S, \varsigma \rangle \to \acute{\varsigma}}{\rho \vdash \langle \text{bind } \acute{\rho} \texttt{ in } S \texttt{ then } z := y, \varsigma \rangle \to \acute{\varsigma}[\rho(z) \mapsto \acute{\varsigma}(\acute{\rho}(y))]} \text{BIND}_2$$

- bind-syntax: "runtime syntax"

$\Rightarrow$ formulation of correctness must be adapted, too (Chap. 3)[1]

[1] Not covered in the lecture.



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Transfer function: Naive formulation

- first attempt
- assumptions:
  - for each *proc. call:* 2 transfer functions: f_{l_c} (call) and f_{l_r} (return)
  - for each *proc. definition:* 2 transfer functions: f_{l_n} (enter) and f_{l_x} (exit)
- given: *mon. framework* $(L, \mathcal{F}, F, E, \iota, f)$

Naive

- treat IF edges $(l_c; l_n)$ and $(l_x; l_r)$ as ordinary flow edges (l_1, l_2)
- *ignore* parameter passing: *transfer* functions for proc. calls and proc definitions are *identity*



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Equation system ("naive" version)

$$
\begin{aligned}
A_\bullet(l) &= f_l(A_\circ(l)) \\
A_\circ(l) &= \bigsqcup\{A_\bullet(l') \mid (l', l) \in F \text{ or } (l'; l) \in F\} \sqcup \iota^l_E
\end{aligned}
$$

with

$$
\iota^l_E = \begin{cases} \iota & \text{if } l \in E \\ \bot & \text{if } l \notin E \end{cases}
$$

- analysis: *safe*
- unnecessarily imprecise, too abstract



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Paths

- remember: "MFP"
- historically: MOP stands for meet over all paths
- here: dually mostly *joins*
- 2 "versions" of a path:
  - path to entry of a block: blocks traversed from the "extremal block" of the program, but not including it
  - path to exit of a block

Paths

$$
\begin{aligned}
path_\circ(l) &= \{[l_1, \ldots \mathbf{l_{n-1}}] \mid l_i \rightarrow_{flow} l_{i+1} \wedge l_n = l \wedge l_1 \in E\} \\
path_\bullet(l) &= \{[l_1, \ldots \mathbf{l_n}] \mid l_i \rightarrow_{flow} l_{i+1} \wedge l_n = l \wedge l_1 \in E\}
\end{aligned}
$$

- transfer function for paths $\vec{l}$

$$f_{\vec{l}} = f_{l_n} \circ \ldots f_{l_1} \circ id$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Meet over all paths

- paths:
  - forward: paths from init block to entry of a block
  - backwards: paths from exits of a block to a final block
- two versions for the MOP solution (for given l):
  - up-to but not including l
  - up-to including l

MOP

$$MOP_\circ(l) = \bigsqcup\{f_{\vec{l}}(\iota) \mid \vec{l} \in path_\circ(l)\}$$
$$MOP_\bullet(l) = \bigsqcup\{f_{\vec{l}}(\iota) \mid \vec{l} \in path_\bullet(l)\}$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

MOP vs. MFP

- MOP: can be undecidable
  - MFP *approximates* MOP ("$MFP \sqsupseteq MOP$")

Lemma

$$MFP_\circ \sqsupseteq MOP_\circ \text{ and } MFP_\bullet \sqsupseteq MOP_\bullet \tag{8}$$

In case of a distributive framework

$$MFP_\circ = MOP_\circ \text{ and } MFP_\bullet = MOP_\bullet \tag{9}$$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

MVP

- take calls and returns (IF) serious
- restrict attention to valid ("possible") paths

$\Rightarrow$ capture the nesting structure

- from MOP to MVP: "meet over all valid paths"
- *complete* path:
  - appropriate call-nesting
  - all calls are answered



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Complete paths

- given $P_* = \texttt{begin}\, D_*\ S_*\, \texttt{end}$
- CP_{l_1,l_2}: complete paths from l_1 to l_2
- generated by the following *productions* (l's are the terminals) (we assume forward analysis here)
- basically a context-free grammar

$$\frac{}{CP_{l,l} \longrightarrow l}$$

$$\frac{(l_1, l_2) \in F}{CP_{l_1,l_3} \longrightarrow l_1, CP_{l_2,l_3}}$$

$$\frac{(l_c, l_n, l_x, l_r) \in IF}{CP_{l_c,l} \longrightarrow l_c, CP_{l_n,l_x}, CP_{l_r,l}}$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Example: Fibonacci

- concrete grammar for fibonacci program:

$$
\begin{aligned}
CP_{9,10} &\longrightarrow 9, CP_{1,8}, CP_{10,10} \\
CP_{10,10} &\longrightarrow 10 \\
CP_{1,8} &\longrightarrow 1, CP_{2,8} \\
CP_{2,8} &\longrightarrow 2, CP_{3,8} \\
CP_{2,8} &\longrightarrow 2, CP_{4,8} \\
CP_{3,8} &\longrightarrow 3, CP_{8,8} \\
CP_{8,8} &\longrightarrow 8 \\
CP_{4,8} &\longrightarrow 4, CP_{1,8}, CP_{5,8} \\
CP_{5,8} &\longrightarrow 5, CP_{6,8} \\
CP_{6,8} &\longrightarrow 6, CP_{1,8}, CP_{7,8} \\
CP_{7,8} &\longrightarrow 7, CP_{8,8}
\end{aligned}
$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Valid paths (context-free grammar)

Valid path (generated from non-terminal VP_*):

- start at extremal node (E),
- all proc *exits* have matching *entries*

$$\frac{l_1 \in E \qquad l_2 \in \mathbf{Lab}_*}{VP_* \longrightarrow VP_{l_1,l_2}} \qquad \frac{}{VP_{l,l} \longrightarrow l}$$

$$\frac{(l_1, l_2) \in F}{VP_{l_1,l_3} \longrightarrow l_1, VP_{l_2,l_3}}$$

$$\frac{(l_c, l_n, l_x, l_r) \in IF}{VP_{l_c,l} \longrightarrow l_c, CP_{l_n,l_x}, VP_{l_r,l}} \qquad \frac{(l_c, l_n, l_x, l_r) \in IF}{VP_{l_c,l} \longrightarrow l_c, VP_{l_n,l}}$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

MVP

- adapt the definition of paths

$$
\begin{aligned}
vpath_\circ(l) &= \{[l_1, \ldots \mathbf{l_{n-1}}] \mid l_n = l \wedge [l_1, \ldots, l_n] \text{ valid}\} \\
vpath_\bullet(l) &= \{[l_1, \ldots \mathbf{l_n}] \mid l_n = l \wedge [l_1, \ldots, l_n] \text{ valid}\}
\end{aligned}
$$

- MVP solution:

$$
\begin{aligned}
MVP_\circ(l) &= \bigsqcup\{f_{\vec{l}}(\iota) \mid \vec{l} \in vpath_\circ(l)\} \\
MVP_\bullet(l) &= \bigsqcup\{f_{\vec{l}}(\iota) \mid \vec{l} \in vpath_\bullet(l)\}
\end{aligned}
$$

- but still: "meets over paths" is *impractical*

Fixpoint calculations

next: how to reconcile the path approach with MFP



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Contexts

- MVP/MOP *undecidable* (but more precise than basic MFP)

$\Rightarrow$ instead of MVP: "embellish" MFP

$$\delta \in \Delta \tag{10}$$

- δ: context information
- for instance: representing/recording of the *path* taken

$\Rightarrow$ "embellishment": adding contexts

Embellished monotone framework

$$(\hat{L}, \hat{\mathcal{F}}, F, E, \hat{\iota}, \hat{f})$$

- intra-procedural (no change of embellishment Δ)
- inter-procedural



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Intra-procedural: basically unchanged

- this part: "independent" of Δ
  - property *lattice* $\hat{L} = \Delta \to L$
  - monotone functions $\hat{\mathcal{F}}$
  - transfer functions: pointwise

$$\hat{f}_l(\hat{d})(\delta) = f_l(\hat{d}(\delta)) \tag{11}$$

- flow equations: "unchanged" for intra-proc. part

$$\begin{aligned} A_\bullet(l) &= \hat{f}_l(A_\circ(l)) \\ A_s \circ (l) &= \bigsqcup\{A_\bullet(l') \mid (l', l) \in F \text{ or } (l'; l) \in F)\} \sqcup \iota^{\hat{l}}_E \end{aligned} \tag{12}$$

- in equation for $A_\bullet$: except for labels l for proc. calls (i.e., not l_c and l_r)



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Sign analysis (unembellished)

- $\mathbf{Sign} = \{-, 0, +\}$, $L_{sign} = 2^{\mathbf{Var}_* \to \mathbf{Sign}}$
- abstract states $\sigma^{sign} \in L_{sign}$
- for *expressions:*
  $[\![_]\!]^{\mathcal{A}_{sign}}_{_} : \mathbf{AExp} \to (\mathbf{Var}_* \to \mathbf{Sign}) \to 2^{\mathbf{Sign}}$

Transfer function for $[x := a]^l$

$$f_l^{sign}(Y) = \bigcup \{\phi_l^{sign}(\sigma^{sign}) \mid \sigma^{sign} \in Y\} \tag{13}$$

where $Y \subseteq \mathbf{Var}_* \to \mathbf{Sign}$ and

$$\phi_l^{sign}(\sigma^{sign}) = \{\sigma^{sign}[x \mapsto s] \mid s \in [\![a]\!]^{\mathcal{A}_{sign}}_{\sigma^{sign}}\} \tag{14}$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Sign analysis: embellished

$$\begin{aligned} \hat{L}_{sign} &= \Delta \to L_{sign} \\ &= \Delta \to 2^{\mathbf{Var}_* \to \mathbf{Sign}} \simeq 2^{\Delta \times (\mathbf{Var}_* \to \mathbf{Sign})} \end{aligned} \tag{15}$$

Transfer function for $[x := a]^l$

$$\hat{f}_l^{sign}(Z) = \bigcup \{ \{\delta\} \times \phi_l^{sign}(\sigma^{sign}) \mid (\delta, \sigma^{sign}) \in Z \} \tag{16}$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Inter-procedural

- procedure *definition* $\texttt{proc}(\texttt{val}\, x, \texttt{res}\, y)\ \texttt{is}^{l_n}\ S\ \texttt{end}^{l_x}$:

$$\hat{f}_{l_n}, \hat{f}_{l_x} : (\Delta \to L) \to (\Delta \to L) = id$$

- procedure call: $(l_c, l_n, l_x, l_r) \in IF$
- here: forward analysis
- call: 2 transfer functions/2 sets of equations, i.e., for all $(l_c, l_n, l_x, l_r) \in IF$

2 transfer functions

1. for calls: $\hat{f^1}_{l_c} : (\Delta \to L) \to (\Delta \to L)$

$$A_\bullet(l_c) = \hat{f^1}_{l_c}(A_\circ(l_c)) \tag{17}$$

1. for returns: $\hat{f^2}_{l_c,l_r} : (\Delta \to L) \times (\Delta \to L) \to (\Delta \to L)$

$$A_\bullet(l_r) = \hat{f^2}_{l_c,l_r}(A_\circ(l_c), A_\circ(l_r))) \tag{18}$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

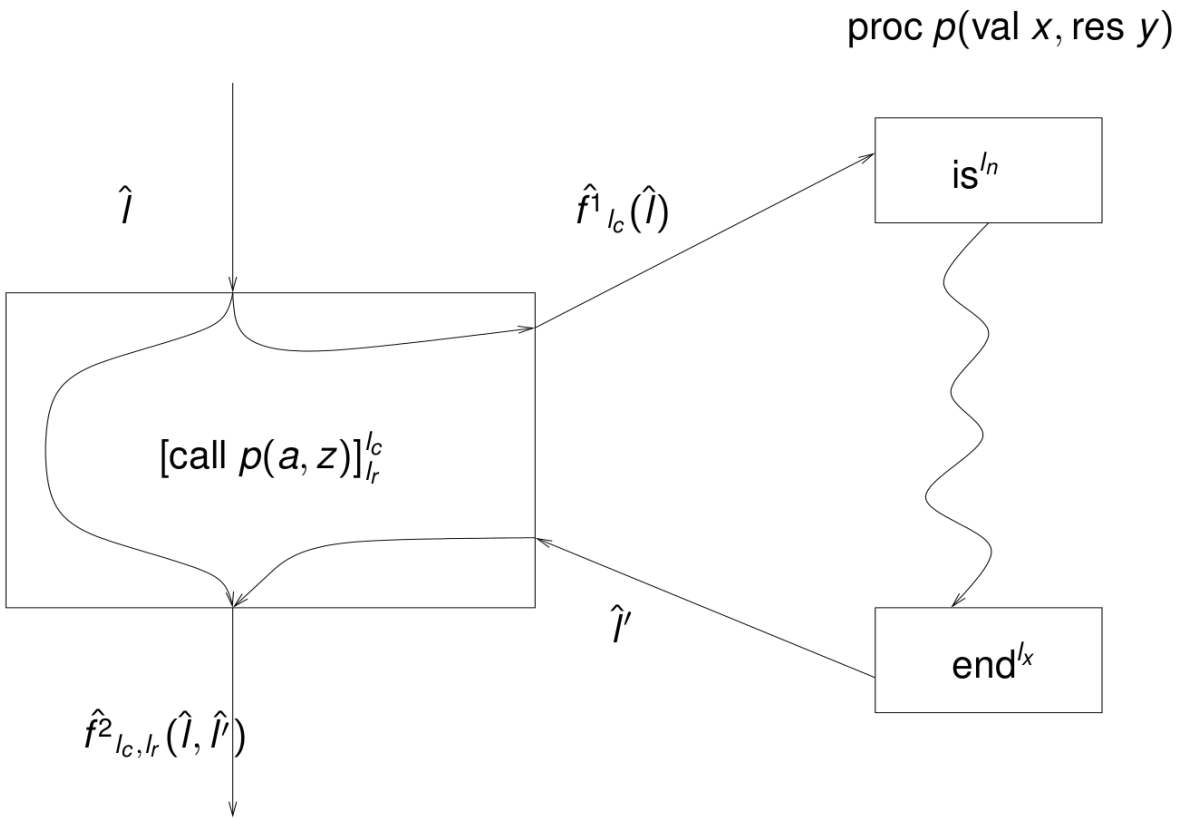
Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Procedure call



Note: *we* write $\hat{d}$ for elements of the embellished lattice (not $\hat{l}$ as in the pic).

Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

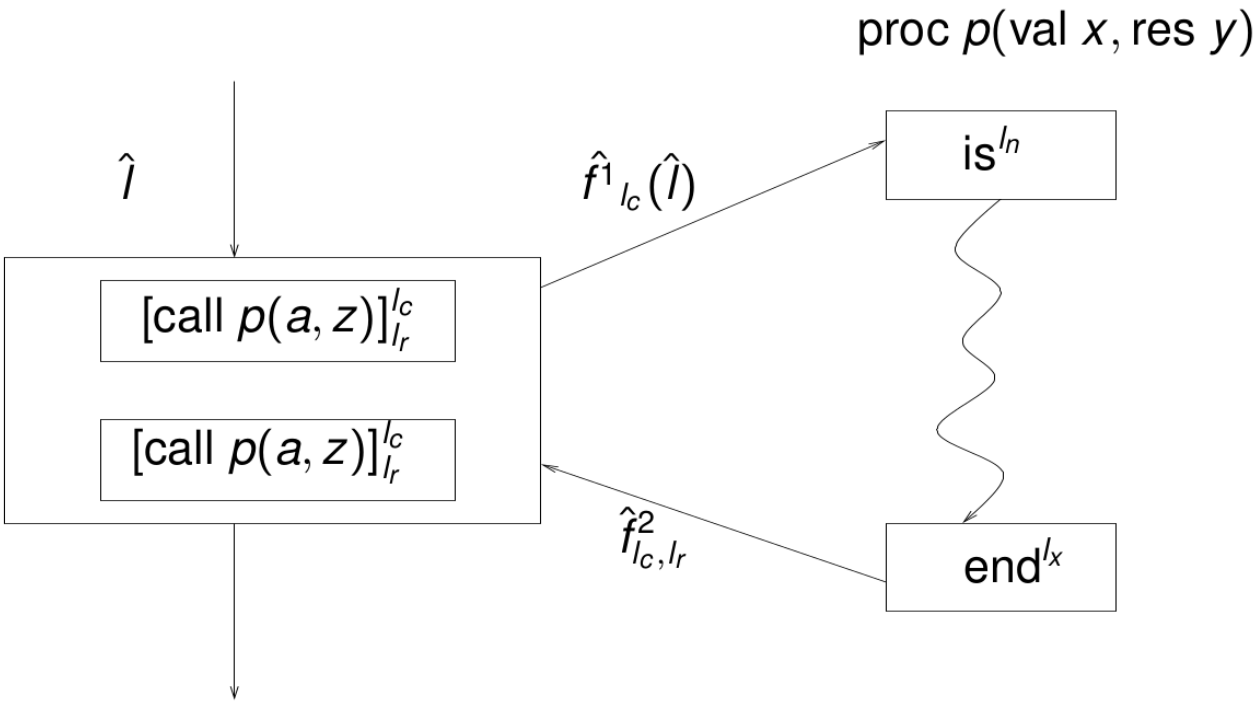
Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Ignoring the call context

$$\hat{f}^2_{l_c,l_r}(\hat{d},\hat{d}') = \hat{f}^2_{l_r}(\hat{d}')$$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

- Determining the control flow graph
- Available expressions
- Reaching definitions
- Very busy expressions
- Live variable analysis

Theoretical properties and semantics

- Semantics
- Intermezzo: Lattices

Monotone frameworks

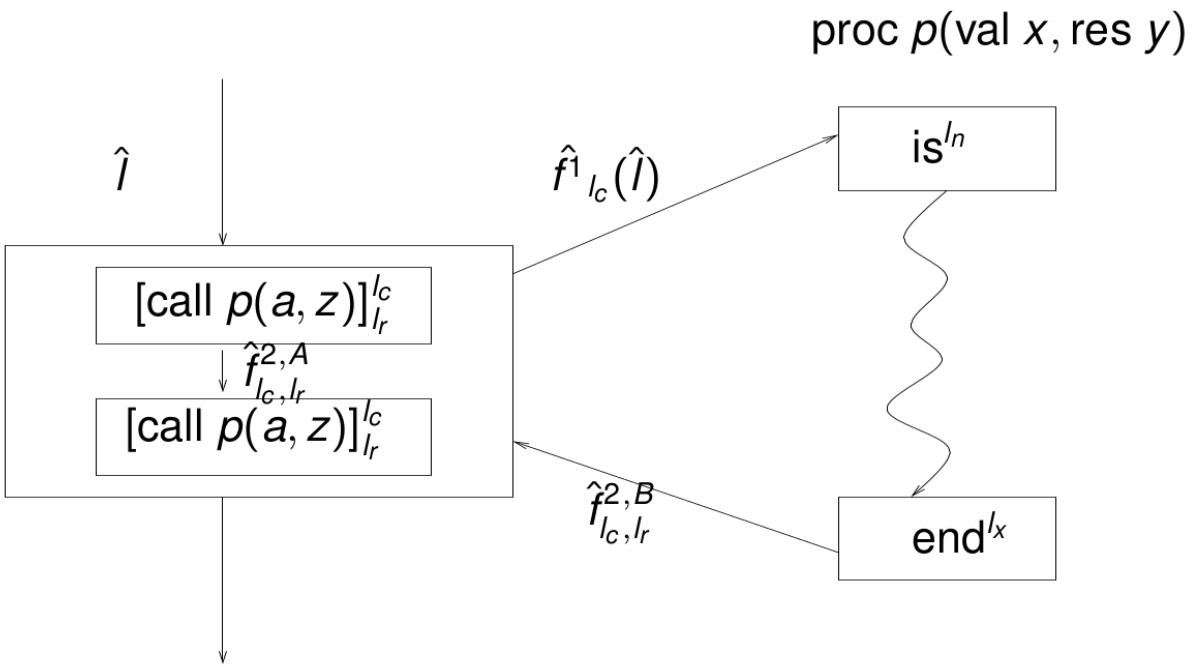
Equation solving

Interprocedural analysis

- Introduction
- Extending the semantics and the CFGs

Merging call contexts

$$\hat{f}^2_{l_c,l_r}(\hat{d},\hat{d}') = \hat{f}^{2A}_{l_c,l_r}(\hat{d}) \sqcup \hat{f}^{2B}_{l_c,l_r}(\hat{d}')$$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

- Determining the control flow graph
- Available expressions
- Reaching definitions
- Very busy expressions
- Live variable analysis

Theoretical properties and semantics

- Semantics
- Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

- Introduction
- Extending the semantics and the CFGs

Context sensitivity

- IF-edges: allow to relate returns to matching calls
- context insensitive: proc-body analysed *combining* flow information from all call-sites.
- *contexts*: used to distinguish different call-sites *and* match calls and returns

$\Rightarrow$ context *sensitive* analysis $\Rightarrow$ more precision + more effort

In the following:



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

2 *specializations*:

1. control (“call strings”)
2. data

(combinations are, of course, possible)



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Call strings

- context = (some form of) *path*
- call-string = sequence of currently "active" calls
- concentrating on calls: flow-edges (l_c, l_n), where just l_c is recorded

$$\Delta = \mathbf{Lab}^* \qquad \text{call strings}$$

- *extremal* value (from $\hat{L} = \Delta \to L$)

$$\hat{\iota}(\delta) =$$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Call strings

- context = (some form of) *path*
- call-string = sequence of currently "active" calls
- concentrating on calls: flow-edges (l_c, l_n), where just l_c is recorded

$$\Delta = \mathbf{Lab}^* \qquad \text{call strings}$$

- *extremal* value (from $\hat{L} = \Delta \to L$)

$$\hat{\iota}(\delta) = \begin{cases} \iota & \text{if } \delta = \epsilon \\ \bot & \text{otherwise} \end{cases}$$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

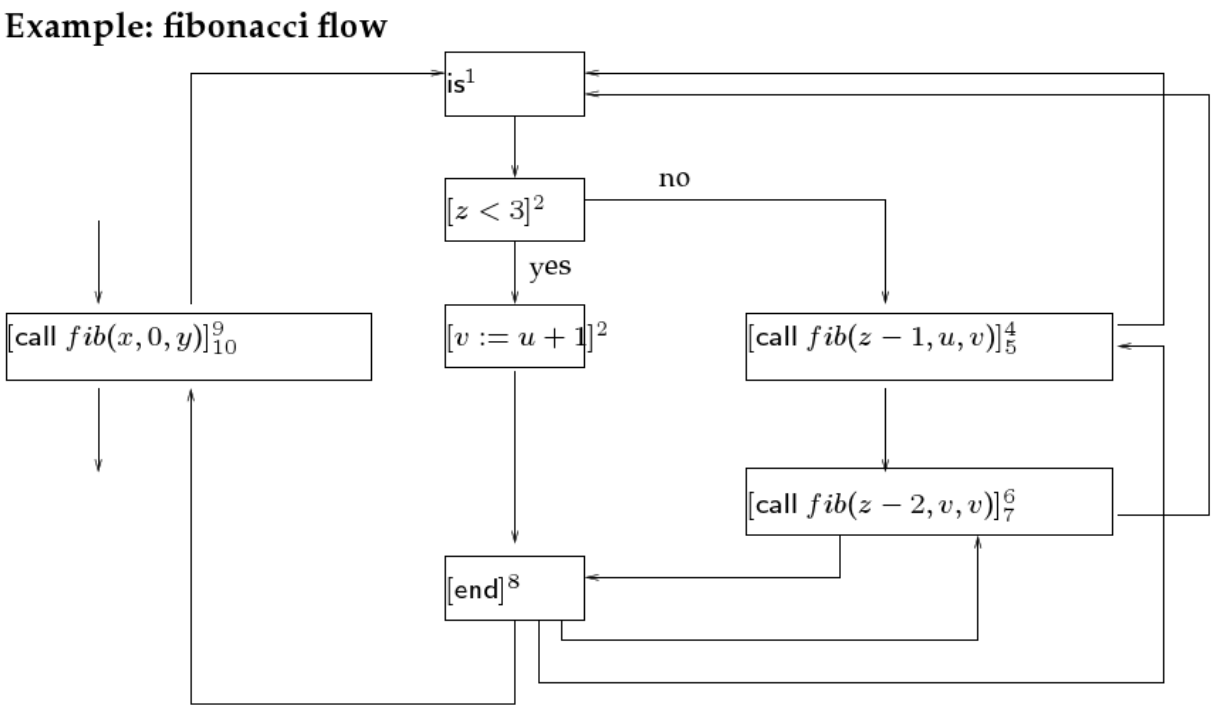
Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Fibonacci flow



Targets & Outline

Intraprocedural analysis

- Determining the control flow graph
- Available expressions
- Reaching definitions
- Very busy expressions
- Live variable analysis

Theoretical properties and semantics

- Semantics
- Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

- Introduction
- Extending the semantics and the CFGs

Fibonacci call strings

some call strings:

$$\epsilon, [9], [9,4], [9,6], [9,4,4], [9,4,6], [9,6,4], [9,6,6], \ldots$$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Transfer functions for call strings

- here: forward analysis
- 2 cases: define $\hat{f}^1_{l_c}$ and $\hat{f}^2_{l_c,l_r}$

Transfer functions

- calls (basically: check that the path ends with l_c):

$$\begin{array}{rcl} (\hat{f}^1_{l_c}\ \hat{d})\ [\delta; l_c] & = & f^1_{l_c}(\hat{d}(\delta)) \\ (\hat{f}^1_{l_c}\ \hat{d})\ _ & = & \bot \end{array} \tag{19}$$

- returns (basically: match return with (a same-level) call)

$$\begin{array}{rcl} (\hat{f}^2_{l_c,l_r}\ (\hat{d}, \hat{d}'))\ \delta & = & f^2_{l_c,l_r}(\hat{d}(\delta), \hat{d}'([\delta; l_c])) \\ (\hat{f}^2_{l_c,l_r}\ (\hat{d}, \hat{d}'))\ _ & = & \bot \end{array} \tag{20}$$

- given: underlying $f^1_{l_c}$ and $f^2_{l_c,l_r}$.
- rather "higher-order" way of connecting the flows, using the call-strings as contexts
- *connection* between the arguments (via δ) of f_{l_c,l_r}



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Sign analysis (continued)

- so far: "unconcrete", i.e.,
- given some underlying analysis: how to make it context-sensitive
- call-strings as context
- now: apply to some simple case: signs
- remember: $\hat{L} \simeq 2^{\Delta \times (\mathbf{Var}_* \to \mathbf{Sign})}$ (see Eq. (15))
- before: standard embellished $\hat{f}_l^{\mathbf{Sign}}$ (with the help of $\phi_l^{\mathbf{Sign}}$)
- now: *inter-procedural*



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Sign analysis: aux. functions ϕ

still unembellished

calls: abstract parameter-passing

$$\phi_{l_c}^{sign1}(\sigma^{sign}) \quad = \quad \{\sigma^{sign}[\mapsto][\mapsto] \mid s \in [\![a]\!]_{\sigma^{sign}}^{\mathcal{A}_{sign}}, \ \}$$

returns (analogously)

$$\phi_{l_c,l_r}^{sign2}(\sigma_1^{sign}, \sigma_2^{sign}) \quad = \quad \{\sigma_2^{sign}[\mapsto]\}$$

(formal params: x, y, where y is the *result parameter*, actual parameter z)

- non-det "assignment" to y
- remember: operational semantics,

Sign analysis: aux. functions ϕ

still unembellished

calls: abstract parameter-passing

$$\phi^{sign1}_{l_c}(\sigma^{sign}) \quad = \quad \{\sigma^{sign}[x \mapsto s][y \mapsto s'] \mid s \in [\![a]\!]^{\mathcal{A}_{sign}}_{\sigma^{sign}},\ s' \in \{-, 0, +\}\}$$

returns (analogously)

$$\phi^{sign2}_{l_c,l_r}(\sigma^{sign}_1, \sigma^{sign}_2) \quad = \quad \{\sigma^{sign}_2[x, y, z \mapsto \sigma^{sign}_1(x), \sigma^{sign}_1(y), \sigma^{sign}_2(y)]\}$$

(formal params: x, y, where y is the *result parameter*, actual parameter z)

- non-det "assignment" to y
- remember: operational semantics,

Sign analysis: embellished f's for call and returns

calls: abstract parameter-passing + glueing calls-returns

$$\hat{f}^{sign1}_{l_c}(Z) \quad = \quad \bigcup\{\{\delta'\} \times \phi^{sign1}_{l_c}(\sigma^{sign}) \mid (\delta, \sigma^{sign}) \in Z, \delta' =)\} \tag{21}$$

Returns: analogously

$$\hat{f}^{sign2}_{l_c,l_r}(Z, Z') \quad = \quad \bigcup\{\{\delta\} \times \phi^{sign2}_{l_c,l_r}(\sigma^{sign}_1, \sigma^{sign}_2) \mid \quad (\delta, \sigma^{sign}_1) \in Z \ \} \tag{22}$$

(formal params: x, y, actual parameter z)

Sign analysis: embellished f's for call and returns

calls: abstract parameter-passing + glueing calls-returns

$$\hat{f}^{sign1}_{l_c}(Z) \quad = \quad \bigcup\{\{\delta'\} \times \phi^{sign1}_{l_c}(\sigma^{sign}) \mid (\delta, \sigma^{sign}) \in Z, \delta' = [\delta; l_c])\} \tag{21}$$

Returns: analogously

$$\hat{f}^{sign2}_{l_c,l_r}(Z, Z') \quad = \quad \bigcup\{\{\delta\} \times \phi^{sign2}_{l_c,l_r}(\sigma_1^{sign}, \sigma_2^{sign}) \mid \begin{array}{l} (\delta, \sigma_1^{sign}) \in Z \\ (\delta', \sigma_2^{sign}) \in Z' \\ \delta' = [\delta; l_c] \end{array} \}$$

(22)

(formal params: x, y, actual parameter z)

Call strings of bounded length

- recursion $\Rightarrow$ call-strings of unbounded length

$\Rightarrow$ restrict the length

$$\Delta = \mathbf{Lab}^{\leq k} \qquad \text{for some } k \geq 0$$

- for $k = 0$: context-insensitive ($\Delta = \{\epsilon\}$)



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Assumption sets

- alternative to call strings
- not tracking the path, but assumption about the "state"
- assume here: lattice $L = 2^D$

$\Rightarrow \hat{L} = \Delta \to L \simeq 2^{\Delta \times D}$

- dependency on "data" only $\Rightarrow$

(large) assumption set context

$$\Delta = 2^D$$

- $\hat{\iota} = \{(\{\iota\}, \iota)\}$ extremal value



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Transfer functions

- calls

$$\hat{f}^1_{l_c}(Z) = \bigcup\{\{\delta'\} \times \phi^1_{l_c}(d) \mid (\delta, d) \in Z \wedge \delta' = \} \quad (23)$$

where $\phi^1_{l_c} : D \to 2^D$

- note: new context δ' for the procedure body
- "caller-callee" connection via the context (= data) δ
- return

$$\hat{f}^2_{l_c,l_r}(Z, Z') = \bigcup\{\{\delta\} \times \phi^2_{l_c,l_r}(d, d') \mid (\delta, d) \in Z \wedge (\delta', d') \in Z' \wedge \delta' = \} \quad (24)$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Transfer functions

- calls

$$\hat{f}^1_{l_c}(Z) = \bigcup\{\{\delta'\} \times \phi^1_{l_c}(d) \mid (\delta, d) \in Z \wedge \delta' = \{d'' \mid (\delta, d'') \in Z\}\} \quad (23)$$

where $\phi^1_{l_c} : D \to 2^D$

- note: new context δ' for the procedure body
- "caller-callee" connection via the context (= data) δ
- return

$$\hat{f}^2_{l_c,l_r}(Z, Z') = \bigcup\{\{\delta\} \times \phi^2_{l_c,l_r}(d, d') \mid (\delta, d) \in Z \wedge (\delta', d') \in Z' \wedge \delta' = \{d'' \mid (\delta, d'') \in Z\}\} \quad (24)$$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Small assumption sets

- throw away even more information.

$$\Delta = D$$

- instead of $\hat{L} = 2^{(2^D \times D)}$: now only $\hat{L} = D \times D$.
- transfer functions simplified

$$\hat{f}^1_{l_c}(Z) = \bigcup\{\{d\} \times \phi^1_{l_c}(d) \mid (\delta, d) \in Z\}$$

$$\hat{f}^2_{l_c,l_r}(Z, Z') = \bigcup\{\{\delta\} \times \phi^2_{l_c,l_r}(d, d') \mid (\delta, d) \in Z \wedge (d, d') \in Z'\}$$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Flow-(in-)sensitivity

- "execution order" influences result of the analysis:

$$S_1; S_2 \quad \text{vs.} \quad S_2; S_1 \tag{25}$$

- flow in-sensitivity: order is irrelevant
- (quite) less precise (but "cheaper")
- for instance: $kill$ is empty
- sometimes useful in combination with inter-proc. analysis



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Set of assigned variables

- for procedure p: determine

$$\mathsf{IAV}(p)$$

global variables that may be assigned to (also indirectly) when p is called

- two aux. definitions (straightforwardly defined, obviously flow-insensitive)
  - $\mathsf{AV}(S)$: assigned variables in S
  - $\mathsf{CP}(S)$: called procedures in S

$$\mathsf{IAV}(p) = (\mathsf{AV}(S) \setminus \{x\}) \cup \bigcup \{\mathsf{IAV}(p') \mid p' \in CP(S)\} \quad (26)$$

where $\mathtt{proc}\, p(\mathtt{val}\, x, \mathtt{res}\, y)\mathtt{is}^{l_n}\, S\, \mathtt{end}^{l_x} \in D_*$

- CP $\Rightarrow$ procedure call graph (which procedure calls which one; see example)



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Example

$$
\begin{array}{ll}
\texttt{begin} & \texttt{proc}\, \mathit{fib}(\texttt{val}\, z)\texttt{is} \\
& \quad \texttt{if} \quad [z < 3] \\
& \quad \texttt{then} \quad [\texttt{call}\, \mathit{add}(a)] \\
& \quad \texttt{else} \quad [\texttt{call}\, \mathit{fib}(z-1)]; \\
& \qquad\qquad\quad [\texttt{call}\, \mathit{fib}(z-2)] \\
& \texttt{end}; \\
& \texttt{proc}\, \mathit{add}(\texttt{val}\, u)\texttt{is}(y := y+1; u := 0) \\
& \texttt{end} \\
& y := 0; [\texttt{call}\, \mathit{fib}(x)] \\
\texttt{end} &
\end{array}
$$



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

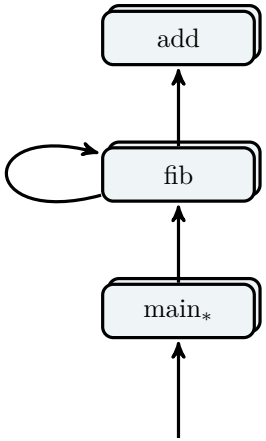
Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Example





Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

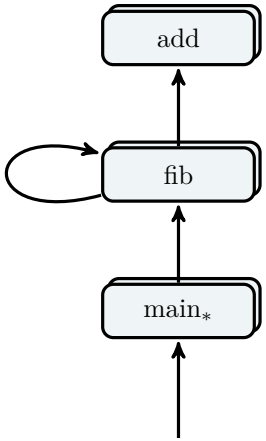
Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Example



$$\begin{aligned} \mathsf{IAV}(\mathit{fib}) &= (\emptyset \setminus \{z\}) \cup \mathsf{IAV}(\mathit{fib}) \cup \mathsf{IAV}(\mathit{add}) \\ \mathsf{IAV}(\mathit{add}) &= \{y, u\} \setminus \{u\} \end{aligned}$$

$\Rightarrow$ smallest solution

$$\mathsf{IAV}(\mathit{fib}) = \{y\}$$

Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs



Section

Static single assignment

Chapter 2 “Data flow analysis”
Course “Static analysis and all that”
Martin Steffen
IN5440 / autumn 2022

Intro

- improvement on *def-use chains*, connecting "definitions" of variables to their uses
- *important* intermediate representation
- used in *many* compilers (gcc, go, swift, llvm, . . .)
- referential transparency

SSA

A program is in SSA form if each variable is a target of *exactly one assignment* in the program *text*.



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Example in SLC

```
a := x + y
b := a - 1
a := y + b
b := x * 4
a := a + b
```

- x and y: *input* variables, "read only"
- assigned to via initialization, *"before"* the program

```
a1 := x + y
b1 := a1 - 1
a2 := y + b2
b2 := x * 4
a3 := a2 + b2
```



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Basis idea (for SLC)

- later more complex
- for straight line code: simple "rename" the variables, like: use different versions x_1, x_2, x_3 for x on the left-hand side.
- some easy *data flow analysis* needed to get a fitting "versioning" for left-hand sides



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

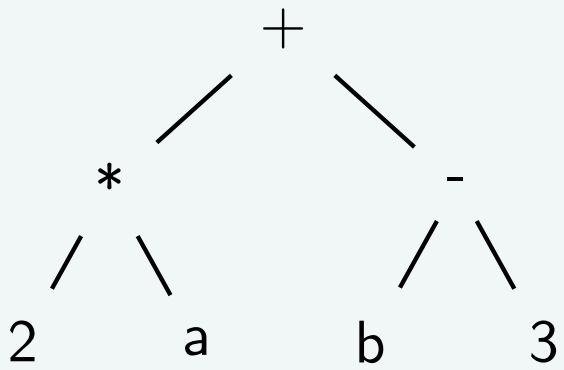
Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Compare: 3AC (here for expressions)

2*a+(b-3)



Three-address code

```
t1 = 2 * a
t2 = b - 3
t3 = t1 + t2
```


Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

- Determining the control flow graph
- Available expressions
- Reaching definitions
- Very busy expressions
- Live variable analysis

Theoretical properties and semantics

- Semantics
- Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

- Introduction
- Extending the semantics and the CFGs

Compare to 3AC and temporaries

- code generation of 3AC for *straight-line code*
- three-address code: linear IR, close to machine code
- *restricted* right-hand sides
- *temporaries*: to store intermediate results
- often
  - temporaries = abstract form of register
  - unboundedly many asssumed

  $\Rightarrow$ each one assigned to only once



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

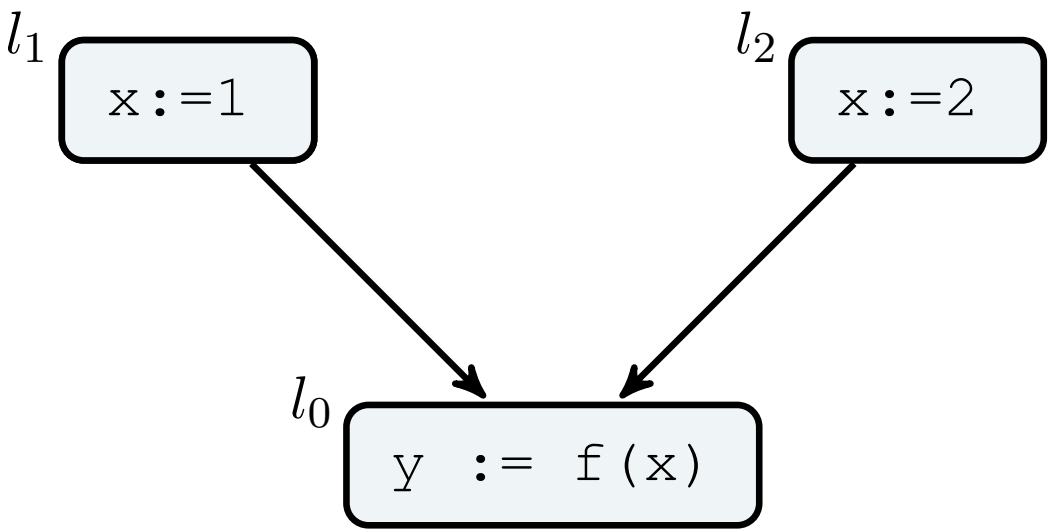
Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Join points and phony functions

- simple illustration: two "definitions" of x





Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

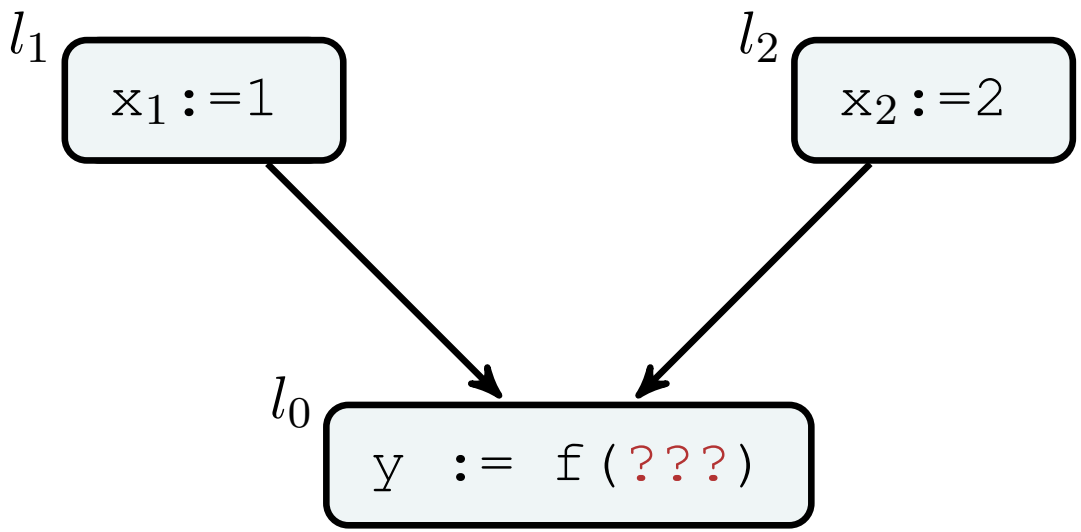
Interprocedural analysis

Introduction

Extending the semantics and the CFGs

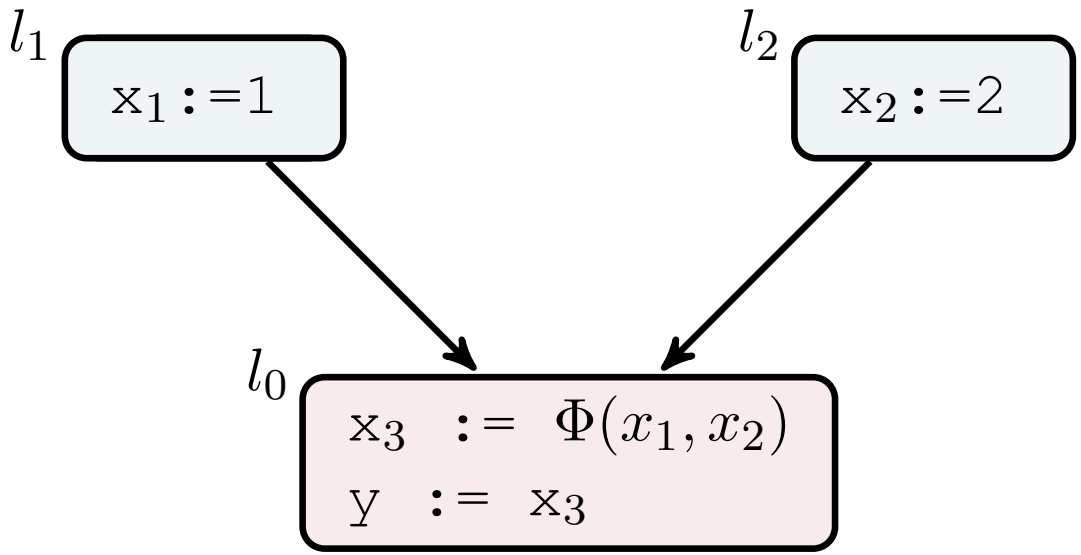
Join points and phony functions

- simple illustration: two "definitions" of x



Join points and phony functions

- simple illustration: two "definitions" of x



Phony functions Φ

Assignments using functions like $\Phi(x_1, x_2)$ placed judiciously at (join) nodes to assure SSA format.

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

SSA in a nutshell

Transformation to SSA

- SSA $= \Phi +$ variable (re)naming scheme



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Phony functions

- "non-standard" function
- encodes "control flow": value depends on if program "came from the left or from the right" in the last step
- Φ
  - "virtual", for purpose of analysis only, or
  - ultimately "real", i.e., code for Φ's will be generated

2 phase algorithm(s), in this order

1. strategical placement of Φ-functions
2. renaming of variables

- *main challenge*: placement of Φ



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Brainless SSA form

- place Φ "everywhere"

Maximal SSA recipe

Placement: For all *variables*, at the beginning of each *join block* add

$$x \leftarrow \Phi(x, \ldots, x) \ ,$$

where number of x's is the number of predecessors of the node ≥ 2

Renaming: rename variables consistently (making use of *reaching definition* analysis)

- note: over-generous placement
  - guarantees *single-assignment* format
  - is sound



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Room for improvements

- phony functions everywhere: sound but *wasteful* and generally undesirable
  - costly extra computations
  - subsequent analyses may suffer loss of precision

phony function unnessessary

- result not *live*
- no real choice: $x_i \leftarrow \Phi(x_j, x_j)$

in the following: efficent general algo for Φ-placement based on

dominance



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

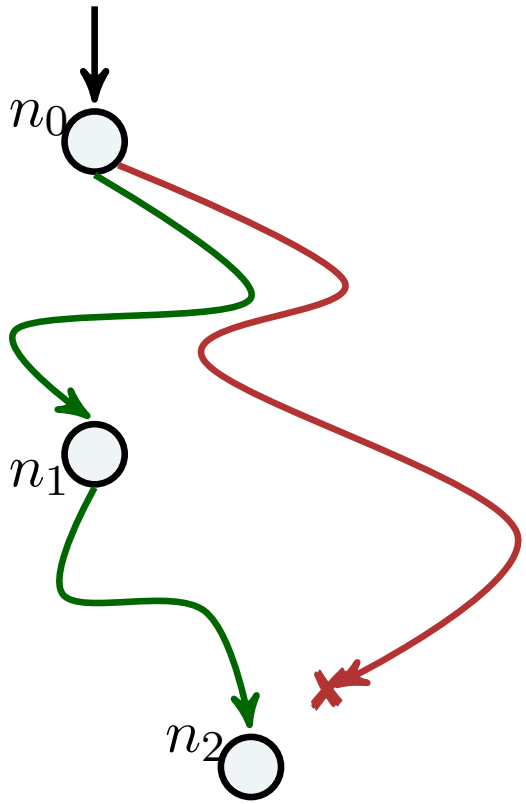
Introduction

Extending the semantics and the CFGs

Improvement

Core idea

Assume assignment $x := e$ in n_1. If all paths from n_0 to n_2 must go through n_1, then n_1's assignment to x does not need to be covered by a phony function for x at n_2.



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Domination

- CFG: directed graph with 1 entry (and 1 exit)
- "content" of the nodes / basic blocks irrelevant right now

Domination

Node n_1 dominates n_2, if all paths from entry n_0 to n_2 must pass through n_1.

- *dominators* of n
- $dom(n)$: determined by a simple data flow analysis (*must* and *forward*)
- also: *strict* dominance



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Dominance frontier

- we know, where not to put Φ for a given assignment at n = all nodes being dominated by n
- danger zone: "undominated", but where to put there?
- solution: as "early" as possible in the danger zone (thinking backwards)

Dominance frontier

The *dominance frontier* $df(n)$ of a node is the collection of m s.t.:

1. n dominates a predecessor of m ($q \rightarrowtriangle m$ and $q \in dom(n)$), and
2. n does *not* strictly dominate m.

- "strict" (non-)dominance condition & loops
- dominance frontier contains *join* nodes only



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

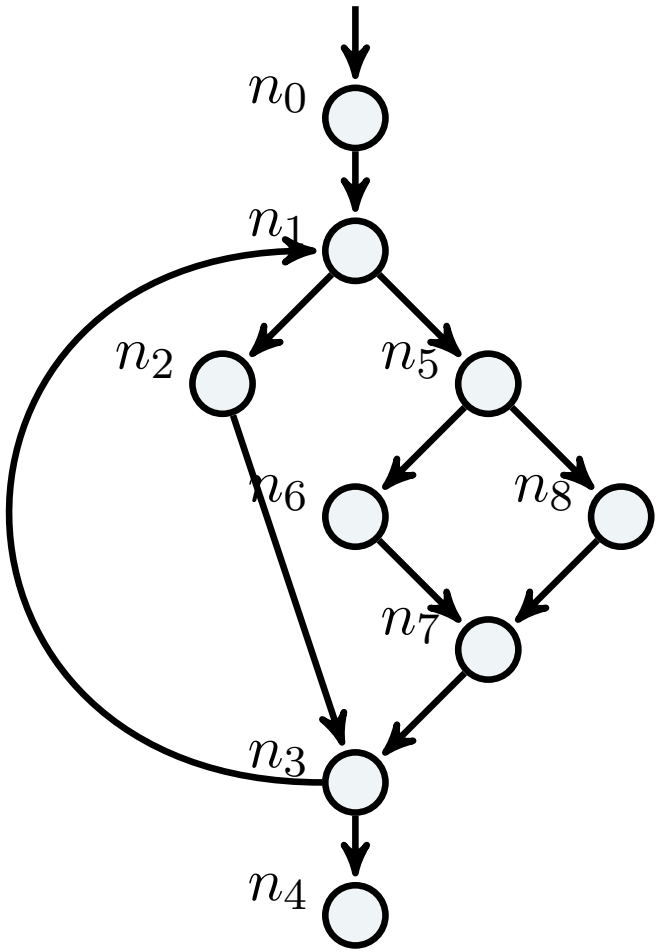
Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Dominator trees

- n's dominators: $dom(n)$
- n's immediate dominator $idom(n)$ (the dominator closest to n, if exists)



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

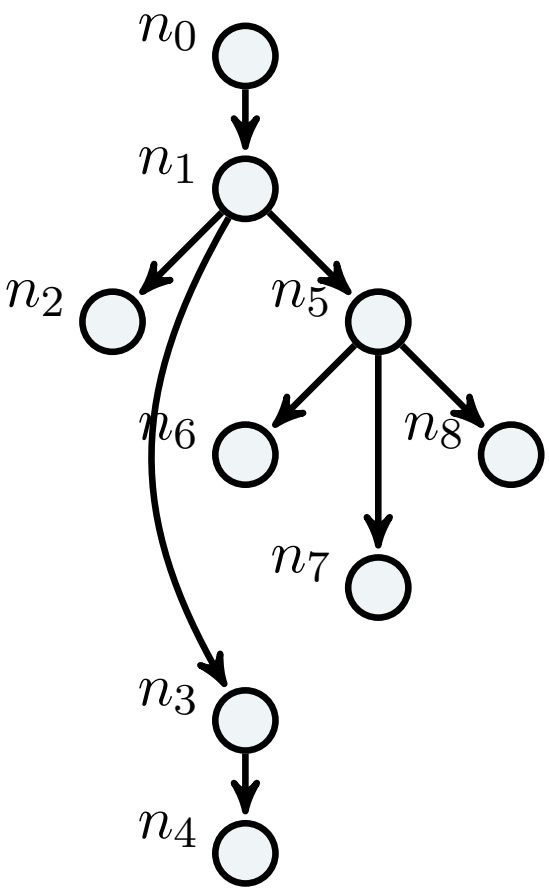
Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Dominator trees

- n's dominators: $dom(n)$
- n's immediate dominator $idom(n)$ (the dominator closest to n, if exists)



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Dominance frontier

```
for all nodes n in the CFG
    DF(n) := ∅

for all nodes n
  if n has multiple predecessors
  then for all predecessors p of n
       runner := p
       while runner ≠ idom(n)
       do DF(runner) := DF(runner) ∪ {n}
          runner     := idom(runner)
```



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Example

	n_0	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8
DF	$\emptyset$	$\{n_1\}$	$\{n_3\}$	$\{n_1\}$	$\emptyset$	$\{n_3\}$	$\{n_7\}$	$\{n_3\}$	$\{n_7\}$



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Further improvement

- left out here:
  - how to actually place the Φ
  - how to then rename the variables
- Basically: problem solved: place it at the dominance frontier



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Further improvement

- left out here:
  - how to actually place the Φ
  - how to then rename the variables
- Basically: problem solved: place it at the dominance frontier
- we forgot: adding $x_i \leftarrow \Phi(x_j, x_j)$ is another assignment

  ...
- further improvements possible (liveness: focus on "global" names)



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

Renaming

```
rename (b)
  for each Φ–function ``x ← Φ(...) in b
    rewrite x as newname(x)

  for each operation x ← y ⊕ z
    rewrite y with subscript top(stack[y])
    rewrite z with subscript top(stack[z])
    rewrite x as newname(x)
  for each successor of b in the CFG
    full in Φ–function parameters
  for each successor s of b in the dominator tree
    rename s
  for each operation ``x ← y ⊕ z'' in b
    and each Φ–function ``x ← Φ(...)''
      pop(stack[x])
```



Static analysis and all that

Martin Steffen

Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs

References I

Bibliography

[1] Nielson, F., Nielson, H.-R., and Hankin, C. L. (1999). *Principles of Program Analysis*. Springer Verlag.



Targets & Outline

Intraprocedural analysis

Determining the control flow graph

Available expressions

Reaching definitions

Very busy expressions

Live variable analysis

Theoretical properties and semantics

Semantics

Intermezzo: Lattices

Monotone frameworks

Equation solving

Interprocedural analysis

Introduction

Extending the semantics and the CFGs