

Static analysis and all that

Martin Steffen

Autumn 2024



Chapter 1

Introduction

Course “Static analysis and all that”

Martin Steffen

IN5440 / autumn 2024



Chapter 1

Learning Targets of Chapter “Introduction”.

Apart from a motivational introduction, the chapter gives a high-level overview over larger topics covered in the lecture. They are treated here just as a teaser and in less depth compared to later but there is already technical content.



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms



Chapter 1

Outline of Chapter "Introduction".

Motivation

- General remarks

Data-flow analysis

- A simplistic while-language
- Equational approach
- Constraint-based approach

Control-flow analysis (constraint-based)

- Control-flow analysis

Type and effect systems

- Introduction
- Annotated type systems
- Annotated type constructors
- Effect systems

Algorithms



Section

Motivation

General remarks

Chapter 1 “Introduction”
Course “Static analysis and all that”
Martin Steffen
IN5440 / autumn 2024

Static analysis: why and what?

- what
  - *static*: at "compile time"
  - *analysis*: deduction of program properties
    - automatic/decidable
    - formally, based on semantics
- why
  - error catching
- catching common "stupid" errors without bothering the user much
- spotting errors early
- certain similarities to model checking
- examples: type checking, uninitialized variables, potential nil-pointer deref's, unused code
- optimization: based on analysis, transform the "code"[1], such the the result is "better"
  - examples: precalculation of results, optimized register allocation ...

[1] source code, intermediate code at various levels



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

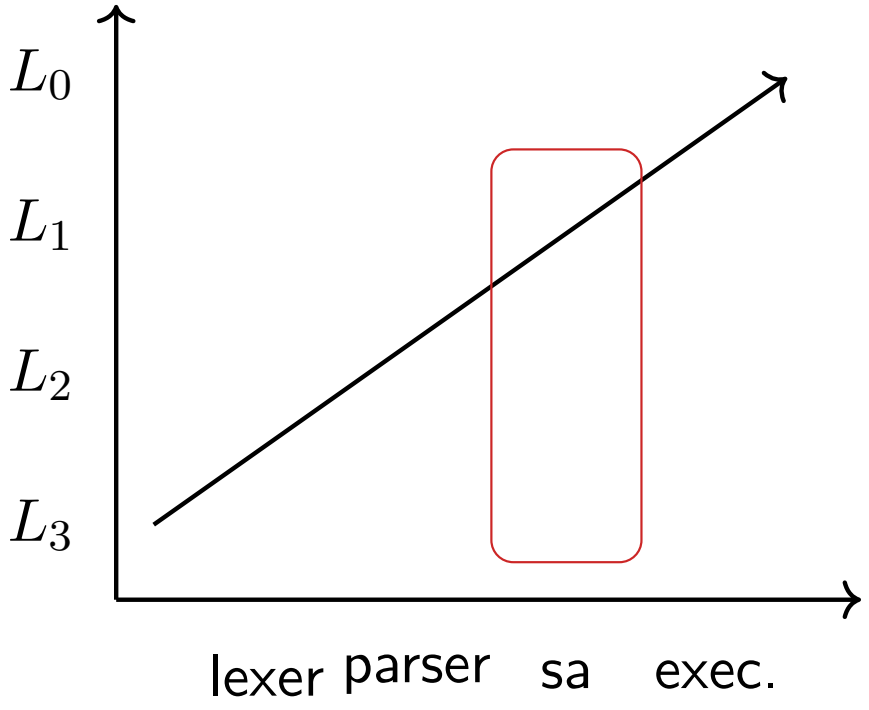
Effect systems

Algorithms

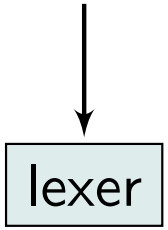
The nature of static analysis

- compiler with differerent *phases*
- corresponding to *Chomsky's hierarchy*
- static = in principle: before run-time
- since: run-time most often: undecidable

$\Rightarrow$ static analysis as approximation



Phases





Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

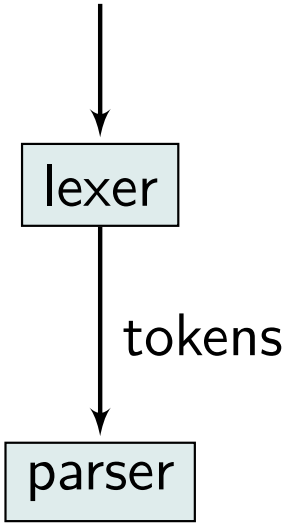
Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Phases





Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

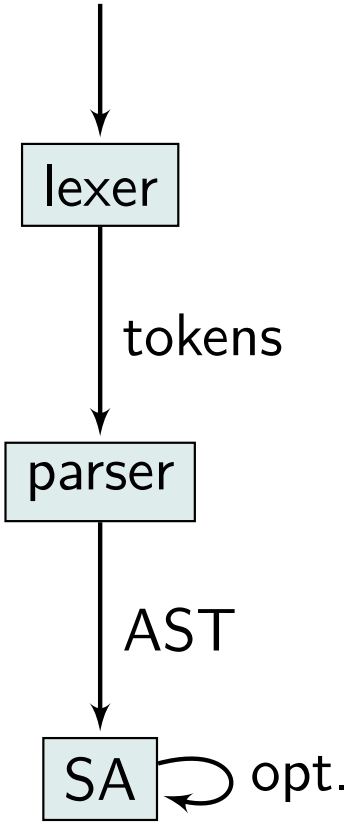
Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Phases





Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

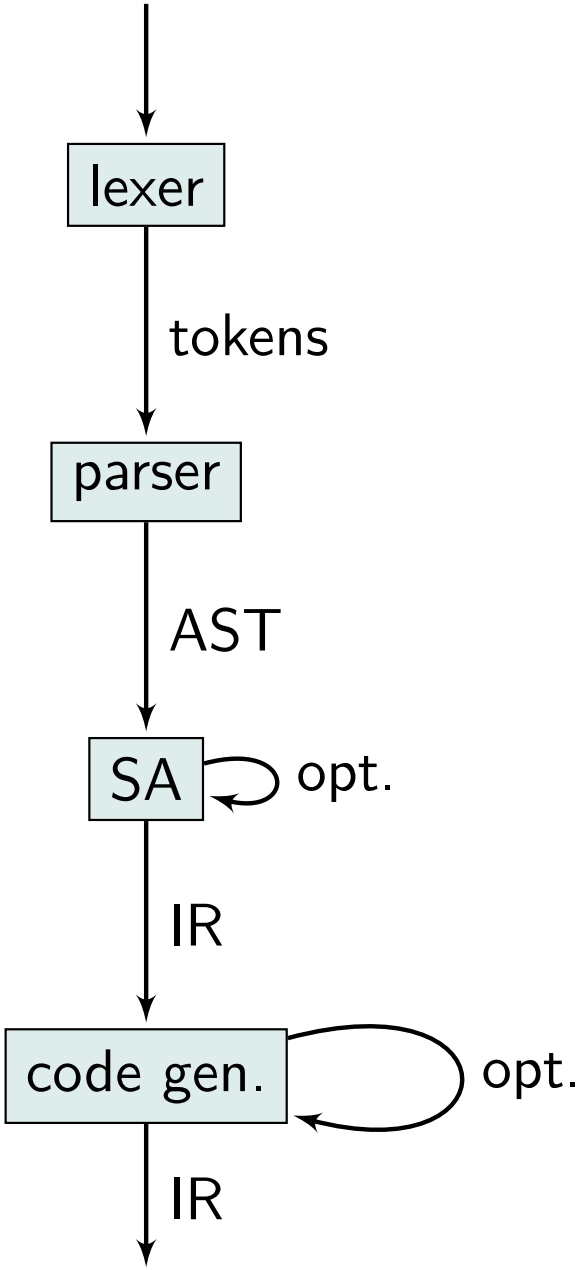
Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Phases





Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

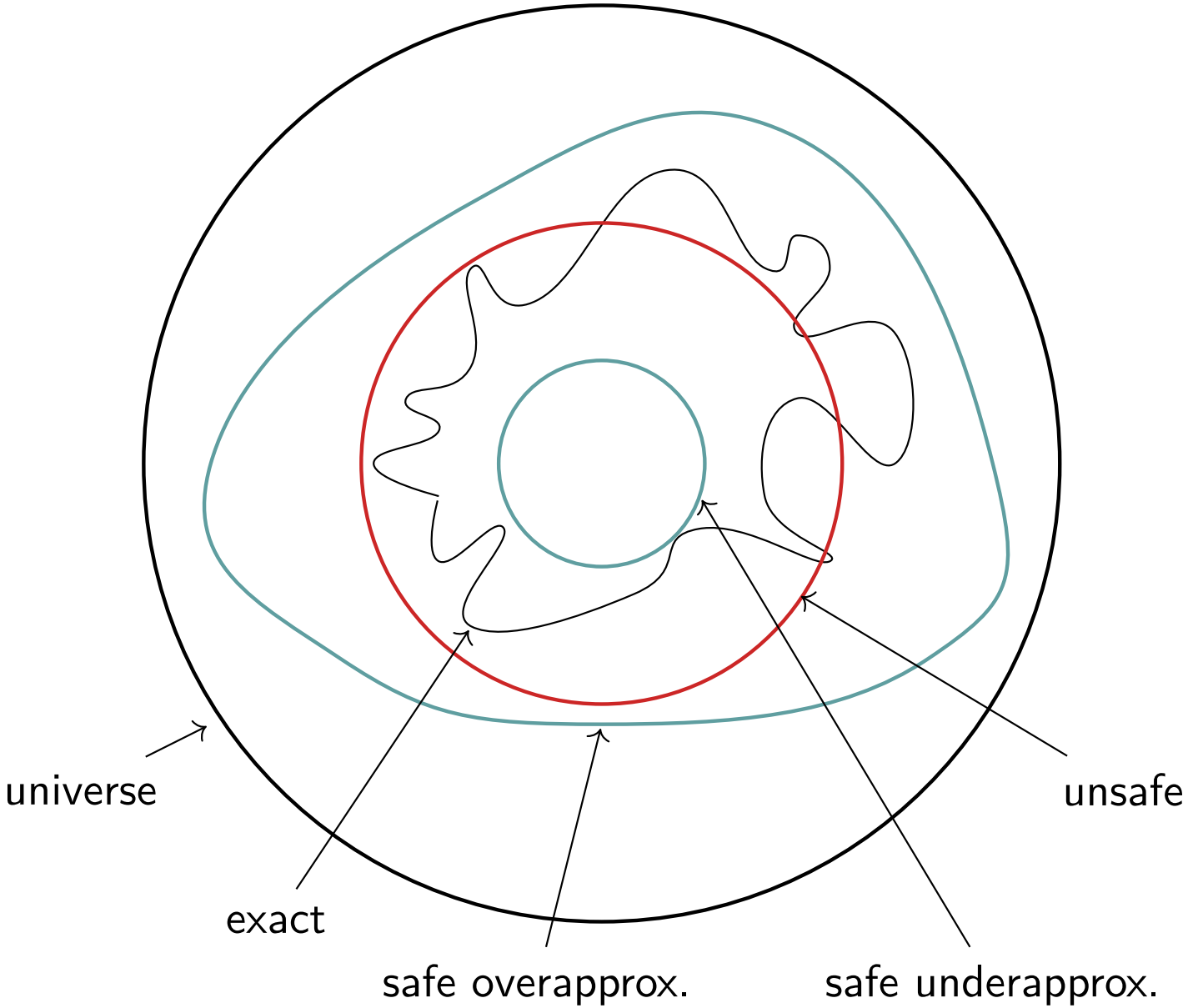
Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Static analysis as approximation





Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Optimal compiler?

Full employment theorem for compiler writers

It's a (mathematically proven!) fact that for any compiler, there exists another one which beats it.

- slightly more than the *non-existence* of optimal compiler or *undecidability* of such a compiler
- theorem
  - it just *guarantees* that there is room for improvement
  - does not say *how!* Actually finding a better one: *undecidable*



Targets & Outline
Motivation
General remarks
Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach
Control-flow analysis (constraint-based)
Control-flow analysis
Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems
Algorithms



Section

Data-flow analysis

A simplistic while-language
Equational approach
Constraint-based approach

Chapter 1 “Introduction”
Course “Static analysis and all that”
Martin Steffen
IN5440 / autumn 2024

While-language

- simple, prototypical imperative language
  - "untyped"
  - simple control structure: while, conditional, sequencing
  - simple data (numerals, booleans)
- abstract syntax $\neq$ concrete syntax
- disambiguation when needed: $(\ldots)$, or $\{\ldots\}$ or begin ... end



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Labelling

- associate *flow* information

$\Rightarrow$ labels

- *elementary block* = labelled item
- identify basic building blocks
- consistent/unique labelling



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Abstract syntax

$$
\begin{array}{lcll}
a & ::= & x \mid n \mid a\,\mathsf{op}_a\,a & \text{arithm. expressions} \\
b & ::= & \mathsf{true} \mid \mathsf{false} \mid \mathsf{not}\ b \mid b\,\mathsf{op}_b\,b \mid a\,\mathsf{op}_r\,a & \text{boolean expr.} \\
S & ::= & x := a \mid \mathsf{skip} \mid S; S & \text{statements} \\
 & \mid & \texttt{if}\ b\ \texttt{then}\ S\ \texttt{else}\ S \mid \texttt{while}\ b\ \texttt{do}\ S &
\end{array}
$$

Abstract syntax

$$
\begin{array}{rcll}
a & ::= & x \mid n \mid a\,\mathtt{op}_a\,a & \text{arithm. expressions} \\
b & ::= & \mathsf{true} \mid \mathsf{false} \mid \mathsf{not}\ b \mid b\,\mathtt{op}_b\,b \mid a\,\mathtt{op}_r\,a & \text{boolean expr.} \\
S & ::= & [x := a]^l \mid [\mathsf{skip}]^l \mid S; S & \text{statements} \\
 & \mid & \mathtt{if}\,[b]^l\,\mathtt{then}\,S\,\mathtt{else}\,S \mid \mathtt{while}\,[b]^l\,\mathtt{do}\,S &
\end{array}
$$

Example: factorial

$$y := x; z := 1; \texttt{while } y > 1 \texttt{ do}(z := z * y; y := y - 1); y := 0$$

- input variable: x
- output variable: z

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



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

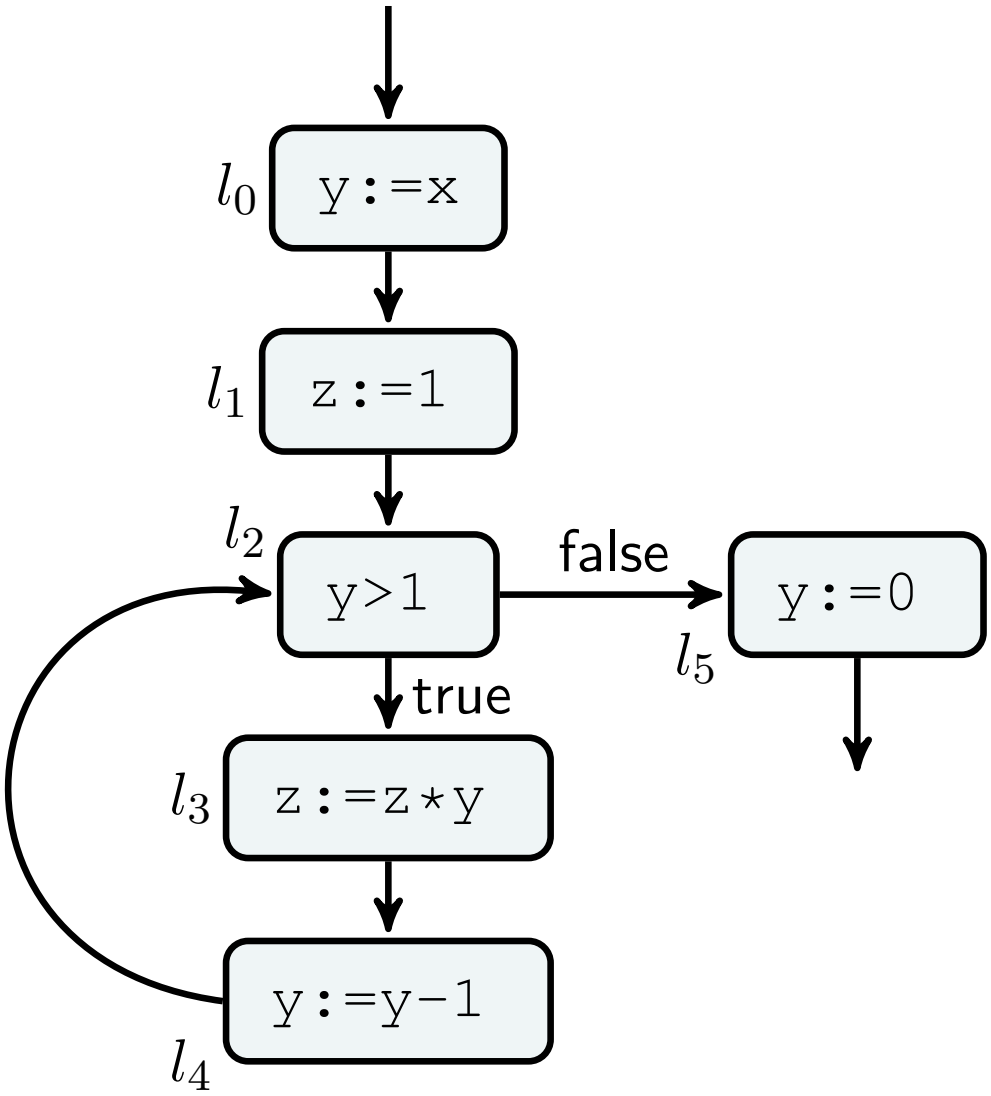
Annotated type systems

Annotated type constructors

Effect systems

Algorithms

CFG factorial





CFG in “real life” (LLVM)

```
int faculty (int x) {
  int y,z;
  y = x;
  z = 1;
  while(y > 1) {
    z = z*y;
    y = y-1;
  };
  y=0;
  return z;
};
```

Listing: Faculty, as C program



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

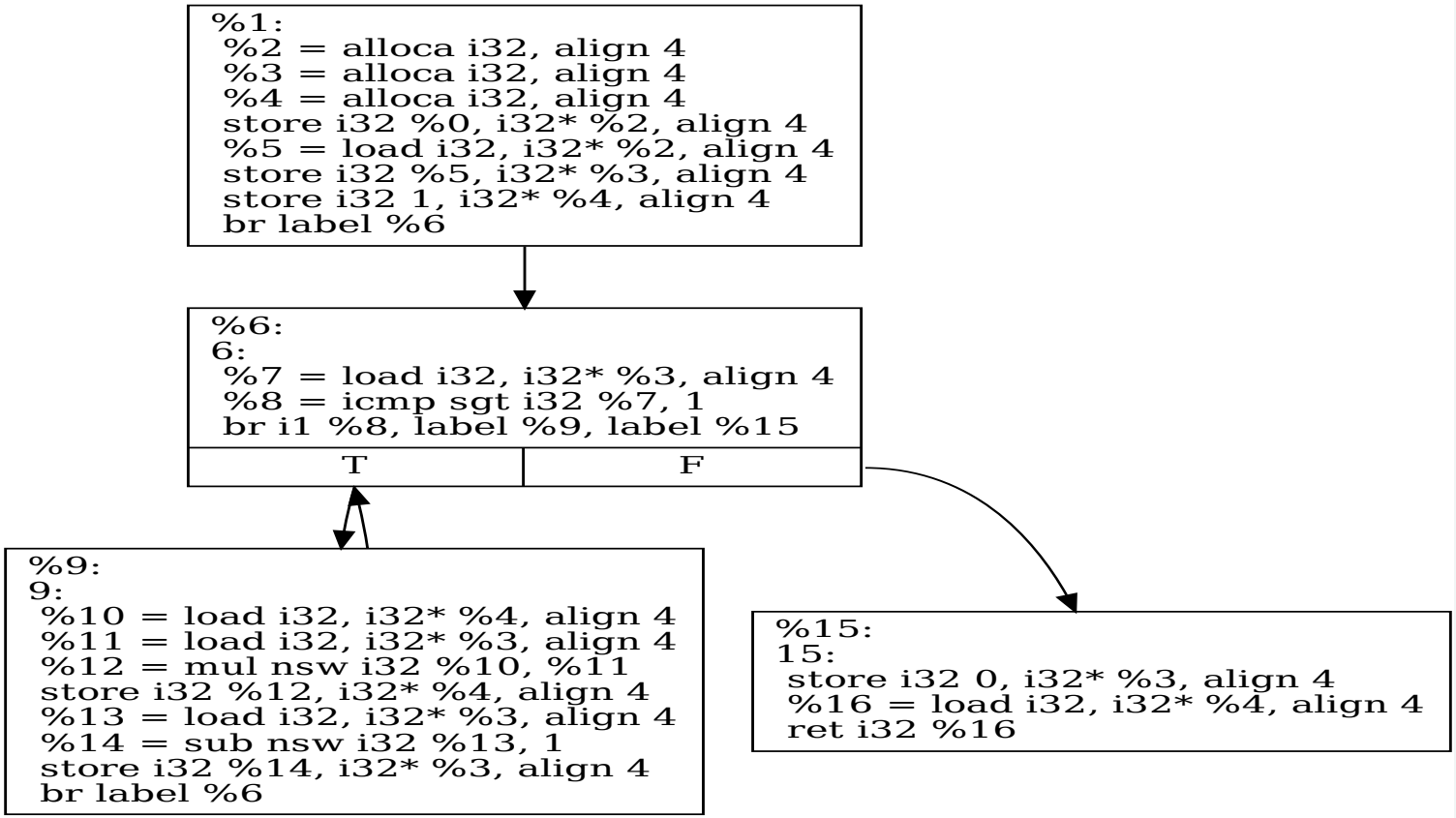
Annotated type systems

Annotated type constructors

Effect systems

Algorithms

CFG in “real life” (LLVM)



CFG for 'faculty' function



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Factorial: reaching definitions analysis

- "definition" of x: assignment to x: $x := a$
- better name: reaching assignment analysis
- first, simple example of data flow analysis

Reaching def's

An *assignment* (= "definition") $[x := a]^l$ *may* reach a program point, if there *exists* an execution where x was *last assigned to* at l, when the mentioned program point is reached.



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

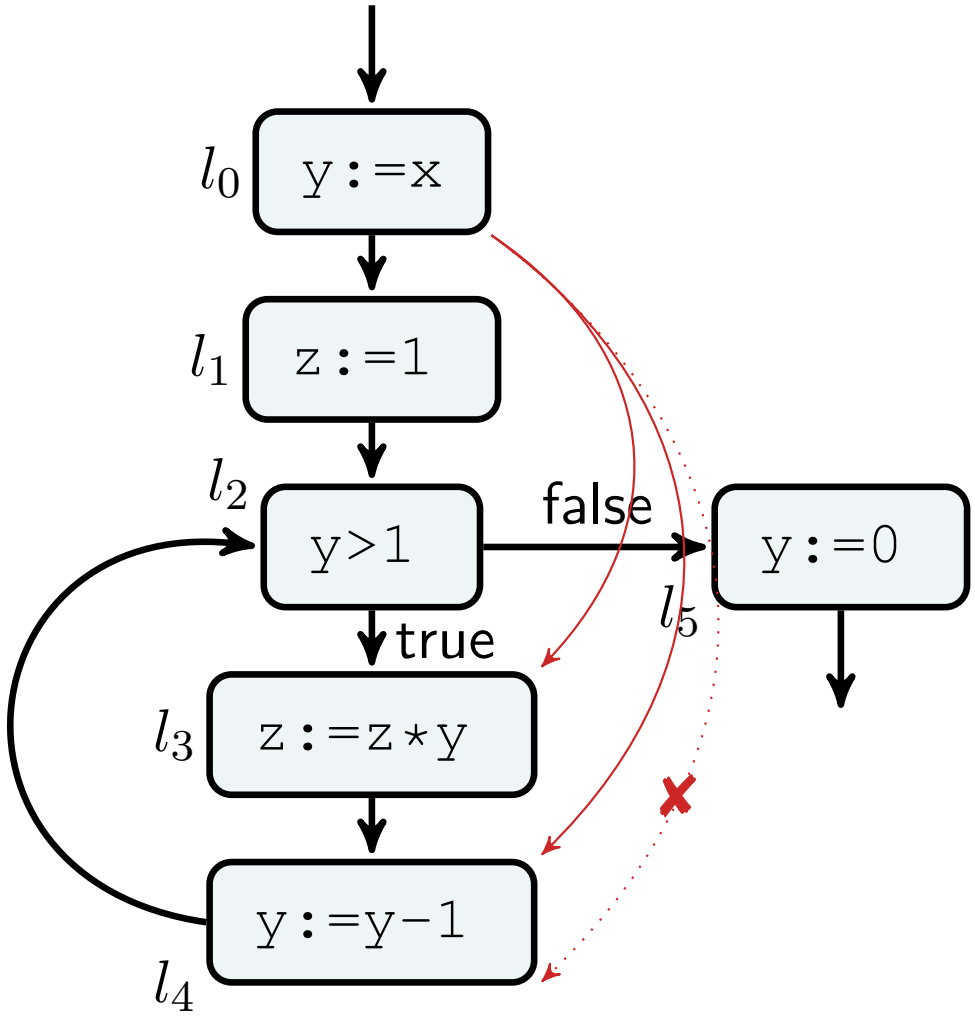
Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Factorial: reaching definitions



- data of interest: tuples of variable × label (or node)
- note: *distinguish* between *entry* and *exit* of a node.

Factorial: reaching assignments

- " points " in the program: *entry* and *exit* to elementary blocks/labels
- $?$: special label (not occurring otherwise), representing *entry* to the program, i.e., $(x, ?)$ represents initial (uninitialized) value of x
- full information: pair of "functions"

$$\mathsf{RD} = (\mathsf{RD}_{entry}, \mathsf{RD}_{exit}) \tag{2}$$

- tabular form (array): see next slide



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

Factorial: reaching assignments table

l	RD_{entry}	RD_{exit}
0	$(x,?),(y,?),(z,?)$	$(x,?),(y,0),(z,?)$
1	$(x,?),(y,0),(z,?)$	$(x,?),(y,0),(z,1)$
2	$(x,?),(y,0),(y,4),(z,1),(z,3)$	$(x,?),(y,0),(y,4),(z,1),(z,3)$
3	$(x,?),(y,0),(y,4),(z,1),(z,3)$	$(x,?),(y,0),(y,4),(z,3)$
4	$(x,?),(y,0),(y,4),(z,3)$	$(x,?),(y,4),(z,3)$
5	$(x,?),(y,0),(y,4),(z,1),(z,3)$	$(x,?),(y,5),(z,1),(z,3)$

Reaching assignments: remarks

- *elementary* blocks of the form
  - $[b]^l$: entry/exit information coincides
  - $[x := a]^l$: entry/exit information (in general) different
- at program exit: $(x, ?)$, x is input variable
- table: *"best"* information = *smallest* sets:
  - additional pairs in the table: still *safe*
  - removing labels: *unsafe*
- note: still an approximation
  - no *real* (= run time) data, no real execution, only *data flow*
  - *approximate* since
    - in *concrete* runs: at each point *in that run*, there is exactly *one* last assignment, not a *set*
    - *label* represents (potentially infinitely many) runs
  - e.g.: at program exit in concrete run: *either* $(z, 1)$ or else $(z, 3)$



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

Data flow analysis

- standard: representation of program as control flow graph (aka flow graph)
  - nodes: elementary blocks with labels (or basic block)
  - edges: flow of control
- two approaches, both (especially here) quite similar
  - *equational* approach
  - *constraint-based* approach



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

From flow graphs to equations

- associate an equation system with the flow graph:
  - describing the "flow of information"
  - here:
    - the information related to reaching assignments
    - information imagined to flow forwards
- solutions of the equations
  - describe *safe* approximations
  - not unique, interest in the *least* (or *largest*) solution
  - here: give back RD of equation (2) on slide 26



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Equations for RD and factorial: intra-block



first type: *local*, intra-block":

- flow through each individual block
- relating for each elementary block its exit with its entry

elementary block: $[y := x]^0$

$$\mathsf{RD}_{exit}(0) \quad = \quad \mathsf{RD}_{entry}(0) \setminus \{(y, l) \mid l \in \mathbf{Lab}\} \cup \{(y, 0)\} \tag{3}$$

Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Equations for RD and factorial: intra-block



first type: *local*, intra-block":

- flow through each individual block
- relating for each elementary block its exit with its entry

elementary block: $[y > 1]^2$

$$\begin{aligned} \mathsf{RD}_{exit}(0) &= \mathsf{RD}_{entry}(0) \setminus \{(y, l) \mid l \in \mathbf{Lab}\} \cup \{(y, 0)\} \\ \mathsf{RD}_{exit}(2) &= \mathsf{RD}_{entry}(2) \end{aligned} \tag{3}$$

Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Equations for RD and factorial: intra-block



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

first type: *local*, intra-block":

- flow through each individual block
- relating for each elementary block its exit with its entry

all equations with RD_{exit} as "left-hand side"

$$
\begin{array}{lcl}
\mathsf{RD}_{exit}(0) & = & \mathsf{RD}_{entry}(0) \setminus \{(y,l) \mid l \in \mathbf{Lab}\} \cup \{(y,0)\} \\
\mathsf{RD}_{exit}(1) & = & \mathsf{RD}_{entry}(1) \setminus \{(z,l) \mid l \in \mathbf{Lab}\} \cup \{(z,1)\} \\
\mathsf{RD}_{exit}(2) & = & \mathsf{RD}_{entry}(2) \\
\mathsf{RD}_{exit}(3) & = & \mathsf{RD}_{entry}(3) \setminus \{(z,l) \mid l \in \mathbf{Lab}\} \cup \{(z,3)\} \\
\mathsf{RD}_{exit}(4) & = & \mathsf{RD}_{entry}(4) \setminus \{(y,l) \mid l \in \mathbf{Lab}\} \cup \{(y,4)\} \\
\mathsf{RD}_{exit}(5) & = & \mathsf{RD}_{entry}(5) \setminus \{(y,l) \mid l \in \mathbf{Lab}\} \cup \{(y,5)\}
\end{array}
\tag{3}
$$

Inter-block flow

second type: *global*, inter-block

- flow *between* the elementary blocks, following the control-flow edges
- relating the *entry* of each block with the *exits* of *other* blocks, that are connected via an *edge* (exception: the initial block has no incoming edge)
- *initial* block: mark variables as *uninitialized*

$$\mathsf{RD}_{entry}(1) = \mathsf{RD}_{exit}(0) \tag{4}$$

$$\begin{aligned} \mathsf{RD}_{entry}(3) &= \mathsf{RD}_{exit}(2) \\ \mathsf{RD}_{entry}(4) &= \mathsf{RD}_{exit}(3) \\ \mathsf{RD}_{entry}(5) &= \mathsf{RD}_{exit}(2) \end{aligned}$$



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Inter-block flow

second type: *global*, inter-block

- flow *between* the elementary blocks, following the control-flow edges
- relating the *entry* of each block with the *exits* of *other* blocks, that are connected via an *edge* (exception: the initial block has no incoming edge)
- *initial* block: mark variables as *uninitialized*

$$
\begin{array}{lcll}
\mathsf{RD}_{entry}(1) & = & \mathsf{RD}_{exit}(0) & (4)\\
\mathsf{RD}_{entry}(2) & = & \mathsf{RD}_{exit}(1) \cup \mathsf{RD}_{exit}(4) & \\
\mathsf{RD}_{entry}(3) & = & \mathsf{RD}_{exit}(2) & \\
\mathsf{RD}_{entry}(4) & = & \mathsf{RD}_{exit}(3) & \\
\mathsf{RD}_{entry}(5) & = & \mathsf{RD}_{exit}(2) &
\end{array}
$$



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

Inter-block flow

second type: *global*, inter-block

- flow *between* the elementary blocks, following the control-flow edges
- relating the *entry* of each block with the *exits* of *other* blocks, that are connected via an *edge* (exception: the initial block has no incoming edge)
- *initial* block: mark variables as *uninitialized*

$$
\begin{array}{lcll}
\mathsf{RD}_{entry}(1) & = & \mathsf{RD}_{exit}(0) & (4)\\
\mathsf{RD}_{entry}(2) & = & \mathsf{RD}_{exit}(1) \cup \mathsf{RD}_{exit}(4) & \\
\mathsf{RD}_{entry}(3) & = & \mathsf{RD}_{exit}(2) & \\
\mathsf{RD}_{entry}(4) & = & \mathsf{RD}_{exit}(3) & \\
\mathsf{RD}_{entry}(5) & = & \mathsf{RD}_{exit}(2) &
\end{array}
$$

$$
\mathsf{RD}_{entry}(0) = \{(x,?),(y,?),(z,?)\}
$$



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

General scheme (for RD)

Intra

- for assignments $[x := a]^l$

$$\mathsf{RD}_{exit}(l) = \mathsf{RD}_{entry}(l) \setminus \{(x, l') \mid l' \in \mathbf{Lab}\} \cup \{(x, l)\} \tag{5}$$

- for other blocks ($[b]^l$ or $[\mathsf{skip}]^l$), side-effect free:

$$\mathsf{RD}_{exit}(l) = \mathsf{RD}_{entry}(l) \tag{6}$$

Inter

$$\mathsf{RD}_{entry}(l) = \bigcup_{l' \to l} \mathsf{RD}_{exit}(l') \tag{7}$$

Initial l: label of the initial block (isolated entry)

$$\mathsf{RD}_{entry}(l) = \{(x, ?) \mid x \text{ is a program variable}\} \tag{8}$$

The equation system as fix point

- RD example: solution to the equation system $=$ *12 sets*

$$\mathrm{RD}_{entry}(0), \ldots, \mathrm{RD}_{exit}(5)$$

- i.e., the $\mathrm{RD}_{entry}(l), \mathrm{RD}_{exit}(l)$ are the *variables* of the equation system, of *type*: sets of pairs of the form (x, l)
- $\vec{\mathrm{RD}}$: the mentioned twelve-tuple of variables

$\Rightarrow$ equation system understood as function F

Equations

$$\vec{\mathrm{RD}} = F(\vec{\mathrm{RD}})$$



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

The least solution

- $\mathbf{Var}_*$ = variables "of interest" (i.e., occurring), $\mathbf{Lab}_*$: labels of interest
- here $\mathbf{Var}_* = \{x, y, z\}$, $\mathbf{Lab}_* = \{?, 1, \ldots, 6\}$

$$F : (2^{\mathbf{Var}_* \times \mathbf{Lab}_*})^{12} \to (2^{\mathbf{Var}_* \times \mathbf{Lab}_*})^{12} \tag{9}$$

- domain $(2^{\mathbf{Var}_* \times \mathbf{Lab}_*})^{12}$: *partially ordered* pointwise:

$$\vec{\mathsf{RD}} \sqsubseteq \vec{\mathsf{RD}}' \text{ iff } \forall i.\ \mathsf{RD}_i \subseteq \mathsf{RD}'_i \tag{10}$$

$\Rightarrow$ complete lattice



Constraint-based approach

- next, for DFA: a simple *variant* of the equational approach
- *rearrangement* of the entry-exit relationships
- instead of equations: *inequations* (sub-set instead of set-equality)
- in more complex settings: constraints become more complex, no split in exit- and entry-constraints



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Factorial program: intra-block constraints

elementary block: $[y := x]^0$

$$\mathsf{RD}_{exit}(0) \supseteq \mathsf{RD}_{entry}(0) \setminus \{(y, l) \mid l \in \mathbf{Lab}\}$$
$$\mathsf{RD}_{exit}(0) \supseteq \{(y, 0)\}$$



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Factorial program: intra-block constraints

elementary block: $[y > 1]^2$

$$\mathsf{RD}_{exit}(2) \supseteq \mathsf{RD}_{entry}(2)$$



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Factorial program: intra-block constraints



all equations with RD_{exit} as left-hand side

$$
\begin{aligned}
\mathsf{RD}_{exit}(0) &\supseteq \mathsf{RD}_{entry}(0) \setminus \{(y,l) \mid l \in \mathbf{Lab}\} \\
\mathsf{RD}_{exit}(0) &\supseteq \{(y,0)\} \\
\mathsf{RD}_{exit}(1) &\supseteq \mathsf{RD}_{entry}(1) \setminus \{(z,l) \mid l \in \mathbf{Lab}\} \\
\mathsf{RD}_{exit}(1) &\supseteq \{(z,1)\} \\
\mathsf{RD}_{exit}(2) &\supseteq \mathsf{RD}_{entry}(2) \\
\mathsf{RD}_{exit}(3) &\supseteq \mathsf{RD}_{entry}(3) \setminus \{(z,l) \mid l \in \mathbf{Lab}\} \\
\mathsf{RD}_{exit}(3) &\supseteq \{(z,3)\} \\
\mathsf{RD}_{exit}(4) &\supseteq \mathsf{RD}_{entry}(4) \setminus \{(y,l) \mid l \in \mathbf{Lab}\} \\
\mathsf{RD}_{exit}(4) &\supseteq \{(y,4)\} \\
\mathsf{RD}_{exit}(5) &\supseteq \mathsf{RD}_{entry}(5) \setminus \{(y,l) \mid l \in \mathbf{Lab}\} \\
\mathsf{RD}_{exit}(5) &\supseteq \{(y,5)\}
\end{aligned}
$$

Targets & Outline
Motivation
General remarks
Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach
Control-flow analysis (constraint-based)
Control-flow analysis
Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems
Algorithms

Factorial program: inter-block constraints

$$
\begin{array}{lcl}
\mathsf{RD}_{entry}(1) & = & \mathsf{RD}_{exit}(0) \\
\mathsf{RD}_{entry}(2) & = & \mathsf{RD}_{exit}(1) \cup \mathsf{RD}_{exit}(4) \\
\mathsf{RD}_{entry}(3) & = & \mathsf{RD}_{exit}(2) \\
\mathsf{RD}_{entry}(4) & = & \mathsf{RD}_{exit}(3) \\
\mathsf{RD}_{entry}(5) & = & \mathsf{RD}_{exit}(2)
\end{array}
$$

$$
\mathsf{RD}_{entry}(0) = \{(x,?),(y,?),(z,?)\}
$$



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

Factorial program: inter-block constraints

splitting of composed right-hand sides + using $\supseteq$ instead of $=$:

$$
\begin{array}{lcl}
\mathsf{RD}_{entry}(1) & \supseteq & \mathsf{RD}_{exit}(0) \\
\mathsf{RD}_{entry}(2) & \supseteq & \mathsf{RD}_{exit}(1) \\
\mathsf{RD}_{entry}(2) & \supseteq & \mathsf{RD}_{exit}(4) \\
\mathsf{RD}_{entry}(3) & \supseteq & \mathsf{RD}_{exit}(2) \\
\mathsf{RD}_{entry}(4) & \supseteq & \mathsf{RD}_{exit}(3) \\
\mathsf{RD}_{entry}(5) & \supseteq & \mathsf{RD}_{exit}(2)
\end{array}
$$

$$
\mathsf{RD}_{entry}(0) \supseteq \{(x,?),(y,?),(z,?)\}
$$



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

Least solution revisited

instead of $F(\vec{\mathsf{RD}}) = \vec{\mathsf{RD}}$

- clear: solution to the equation system $\Rightarrow$ solution to the constraint system
- important: least solutions *coincides*!

Pre-fixpoint

$$F(\vec{\mathsf{RD}}) \sqsubseteq \vec{\mathsf{RD}} \tag{11}$$



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms



Section

Control-flow analysis (constraint-based)

Control-flow analysis

Chapter 1 “Introduction”
Course “Static analysis and all that”
Martin Steffen
IN5440 / autumn 2024

Control-flow analysis

Goal CFA

which elem. blocks lead to which other elem. blocks

- for while-language: immediate (labelled elem. blocks, resp., graph)
- complex for: more *advanced* features, *higher-order* languages, oo languages ...
- here: prototypical higher-order functional language λ-calculus
- formulated as constraint-based analysis



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

Simple example

```
let  f  = fn x => x 1;
     g  = fn y => y + 2;
     h  = fn z => z + 3;
in (f g) + (f h)
```

- *higher-order* function f
- for simplicity: untyped
- local definitions via let-in
- interesting *above:* $x\ 1$

Goal (more specifically)

For each function application, which function may be applied.



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

Labelling

- more complex language $\Rightarrow$ more *complex* labelling
- "elem. blocks" can be *nested*
- *all* syntactic constructs (expressions) are labelled
- consider:

Unlabelled abstract syntax

$$(\mathtt{fn}\, x \Rightarrow x)\ (\mathtt{fn}\, y \Rightarrow y)$$

- functional language: side-effect free

$\Rightarrow$ *no* need to distinguish *entry* and *exit* of labelled blocks.



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

Labelling

- more complex language ⇒ more *complex* labelling
- "elem. blocks" can be *nested*
- *all* syntactic constructs (expressions) are labelled
- consider:

Full labelling

$$[\ [\mathtt{fn}\, x \Rightarrow [x]^1]^2\ [\mathtt{fn}\, y \Rightarrow [y]^3]^4\ ]^5$$

- functional language: side-effect free

⇒ *no* need to distinguish *entry* and *exit* of labelled blocks.



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

Data of the analysis



Data of the analysis:

Pairs $(\hat{C}, \hat{\rho})$ of mappings:

abstract cache: $\hat{C}(l)$: set of values/function abstractions, the subexpression labelled l may evaluate to

abstract env.: $\hat{\rho}$: values, x may be bound to

Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

The constraint system

- ignoring "`let`" here: *three* syntactic constructs $\Rightarrow$ *three* kinds of constraints
- relating $\hat{C}$, $\hat{\rho}$, and the program in form of subset constraints (subsets, order-relation)



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

The constraint system

- ignoring "`let`" here: *three* syntactic constructs $\Rightarrow$ *three* kinds of constraints
- relating $\hat{C}$, $\hat{\rho}$, and the program in form of subset constraints (subsets, order-relation)

3 syntactic classes

- function abstraction: $[\texttt{fn}\, x \Rightarrow x]^l$
- variables: $[x]^l$
- application: $[f\ g]^l$



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

Constraint system for the small example

Labelled example

$$[\ [\mathtt{fn}\, x \Rightarrow [x]^1]^2\ [\mathtt{fn}\, y \Rightarrow [y]^3]^4\ ]^5$$



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

Constraint system for the small example

Labelled example

$$[\ [\mathtt{fn}\, x \Rightarrow [x]^1]^2\ [\mathtt{fn}\, y \Rightarrow [y]^3]^4\ ]^5$$

- function abstractions

$$\begin{array}{rcl} \{\mathtt{fn}\, x \Rightarrow [x]^1\} & \subseteq & \hat{C}(2) \\ \{\mathtt{fn}\, y \Rightarrow [y]^3\} & \subseteq & \hat{C}(4) \end{array}$$



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Constraint system for the small example

Labelled example

$$[\ [\mathtt{fn}\, x \Rightarrow [x]^1]^2\ [\mathtt{fn}\, y \Rightarrow [y]^3]^4\ ]^5$$

- variables (occurrences of use)

$$\begin{array}{lcl} \hat{\rho}(x) & \subseteq & \hat{C}(1) \\ \hat{\rho}(y) & \subseteq & \hat{C}(3) \end{array}$$



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Constraint system for the small example

Labelled example

$$[\ [\mathtt{fn}\, x \Rightarrow [x]^1]^2\ [\mathtt{fn}\, y \Rightarrow [y]^3]^4\ ]^5$$

- application: connecting function entry and (body) exit with the argument

$$\begin{array}{lcl} \hat{C}(4) & \subseteq & \hat{\rho}(x) \\ \hat{C}(1) & \subseteq & \hat{C}(5) \end{array}$$



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

Constraint system for the small example

Labelled example

$$[\ [\mathtt{fn}\, x \Rightarrow [x]^1]^2\ [\mathtt{fn}\, y \Rightarrow [y]^3]^4\ ]^5$$

- application: connecting function entry and (body) exit with the argument but:
- also $[\mathtt{fn}\, y \Rightarrow [y]^3]^4$ is a candidate at 2! (according to $\hat{C}(2)$)

$$\begin{array}{lcl} \hat{C}(4) & \subseteq & \hat{\rho}(x) \\ \hat{C}(1) & \subseteq & \hat{C}(5) \\ \hat{\mathbf{C}}(\mathbf{4}) & \subseteq & \hat{\rho}(y) \\ \hat{C}(3) & \subseteq & \hat{\mathbf{C}}(\mathbf{5}) \end{array}$$



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

Constraint system for the small example

Labelled example

$$[\ [\mathtt{fn}\, x \Rightarrow [x]^1]^2\ [\mathtt{fn}\, y \Rightarrow [y]^3]^4\ ]^5$$

$$\begin{array}{lcll}
\{\mathtt{fn}\, x \Rightarrow [x]^1\} \subseteq \hat{C}(2) & \Rightarrow & \hat{C}(4) & \subseteq \hat{\rho}(x) \\
\{\mathtt{fn}\, x \Rightarrow [x]^1\} \subseteq \hat{C}(2) & \Rightarrow & \hat{C}(1) & \subseteq \hat{C}(5) \\
\{\mathtt{fn}\, y \Rightarrow [y]^3\} \subseteq \hat{C}(2) & \Rightarrow & \hat{\mathbf{C}}(\mathbf{4}) & \subseteq \hat{\rho}(y) \\
\{\mathtt{fn}\, y \Rightarrow [y]^3\} \subseteq \hat{C}(2) & \Rightarrow & \hat{C}(3) & \subseteq \hat{\mathbf{C}}(\mathbf{5})
\end{array}$$



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

The least (= best) solution

$$
\begin{array}{rcl}
\hat{C}(1) & = & \{\mathtt{fn}\, y \Rightarrow [y]^3\} \\
\hat{C}(2) & = & \{\mathtt{fn}\, x \Rightarrow [x]^1\} \\
\hat{C}(3) & = & \emptyset \\
\hat{C}(4) & = & \{\mathtt{fn}\, y \Rightarrow [y]^3\} \\
\hat{C}(5) & = & \{\mathtt{fn}\, y \Rightarrow [y]^3\} \\
\hline
\hat{\rho}(x) & = & \{\mathtt{fn}\, y \Rightarrow [y]^3\} \\
\hat{\rho}(y) & = & \emptyset
\end{array}
$$

One interesting bit here in the solution is: $\hat{\rho}(y) = \emptyset$: that means, the variable y never evaluated, i.e., the function is not applied at all.



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms



Section

Type and effect systems

Introduction
Annotated type systems
Annotated type constructors
Effect systems

Chapter 1 “Introduction”
Course “Static analysis and all that”
Martin Steffen
IN5440 / autumn 2024

Effects: Intro

- type system: "classical" static analysis:

$$t : \tau$$

- *judgment*: "term or program phrase has type τ"
- in general: *context-sensitive* judgments (remember Chomsky . . .)

Judgement :

$$\Gamma \vdash t : \tau$$

- Γ: *assumption* or *context*
- here: *"non-standard"* type systems: effects and annotations
- natural setting: typed languages, here: *trivial!* setting (while-language)



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

"Trival" type system

- setting: while-language
- each statement maps: state to states
- Σ: type of *states*

judgement

$$\vdash S : \Sigma \to \Sigma \tag{12}$$

- specified as a *derivation* system
- note: *partial* correctness assertion



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

"Trival" type system: rules

$$\vdash [x := a]^l : \Sigma \to \Sigma \qquad \textsc{Ass}$$

$$[\mathsf{skip}]^l : \Sigma \to \Sigma \qquad \textsc{Skip}$$

$$\frac{\vdash S_1 : \Sigma \to \Sigma \qquad S_2 : \Sigma \to \Sigma}{\vdash S_1; S_2 : \Sigma \to \Sigma}\ \textsc{Seq}$$

$$\frac{\vdash S : \Sigma \to \Sigma}{\vdash \mathtt{while}[b]^l\ \mathtt{do}\ S : \Sigma \to \Sigma}\ \textsc{While}$$

$$\frac{\vdash S_1 : \Sigma \to \Sigma \qquad \vdash S_2 : \Sigma \to \Sigma}{\vdash \mathtt{if}[b]^l\ \mathtt{then}\ S_1\ \mathtt{else}\ S_2 : \Sigma \to \Sigma}\ \textsc{Cond}$$



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Types, effects, and annotations

annot. type system

$$\vdash S : \Sigma_1 \to \Sigma_2 \quad (13)$$

effect system

$$\vdash S : \Sigma \xrightarrow{\varphi} \Sigma \quad (14)$$

type and effect system (TES)

- *effect* system + *annotated* type system
- borderline fuzzy
- annotated type system
  - Σ_i: property of state ("$\Sigma_i \subseteq \Sigma$")
  - "abstract" properties: invariants, a variable is positive, etc.
- effect system
  - "statement S maps state to state, with (potential ...) effect φ"
  - *effect* φ: e.g.: errors, exceptions, file/resource access, ...



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Annotated type systems

- example again: *reaching definitions* for while-language
- 2 flavors
  1. annotated base types: $S : \mathsf{RD}_1 \to \mathsf{RD}_2$
  2. annotated type constructors: $S : \Sigma \xrightarrow[\mathsf{RD}]{X} \Sigma$



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

RD with annotated base types

judgement

$$\vdash S : \mathsf{RD}_1 \to \mathsf{RD}_2 \tag{15}$$

- $\mathsf{RD} \subseteq 2^{\mathbf{Var} \times \mathbf{Lab}}$
- auxiliary functions
  - note: every S has one "initial" elementary block, potentially more than one "at the end"
  - $init(S)$: the (unique) label at the entry of S
  - $final(S)$: the set of labels at the exits of S

"meaning" of judgment $\vdash S : \mathsf{RD}_1 \to \mathsf{RD}_2$

"RD_1 is the set of var/label reaching the entry of S and RD_2 the corresponding set at the exit(s) of S":

$$\begin{aligned} \mathsf{RD}_1 &= \mathsf{RD}_{entry}(init(S)) \\ \mathsf{RD}_2 &= \bigcup\{\mathsf{RD}_{exit}(l) \mid l \in final(S)\} \end{aligned}$$



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

$$\vdash [x := a]^{l'} : \mathsf{RD} \to \mathsf{RD} \setminus \{(x, l) \mid l \in \mathbf{Lab}\} \cup \{(x, l')\} \qquad \textsc{ass}$$

$$\vdash [\mathsf{skip}]^{l} : \mathsf{RD} \to \mathsf{RD} \qquad \textsc{skip}$$

$$\frac{\vdash S_1 : \mathsf{RD}_1 \to \mathsf{RD}_2 \qquad \vdash S_2 : \mathsf{RD}_2 \to \mathsf{RD}_3}{\vdash S_1; S_2 : \mathsf{RD}_1 \to \mathsf{RD}_3} \textsc{Seq}$$

$$\frac{\vdash S_1 : \mathsf{RD}_1 \to \mathsf{RD}_2 \qquad \vdash S_2 : \mathsf{RD}_1 \to \mathsf{RD}_2}{\vdash \mathtt{if}[b]^{l} \,\mathtt{then}\, S_1 \,\mathtt{else}\, S_2 : \mathsf{RD}_1 \to \mathsf{RD}_2} \textsc{If}$$

$$\frac{\vdash S : \mathsf{RD} \to \mathsf{RD}}{\vdash \mathtt{while}[b]^{l} \,\mathtt{do}\, S : \mathsf{RD} \to \mathsf{RD}} \textsc{While}$$

$$\frac{\vdash S : \mathsf{RD}'_1 \to \mathsf{RD}'_2 \qquad \mathsf{RD}_1 \subseteq \mathsf{RD}'_1 \qquad \mathsf{RD}'_2 \subseteq \mathsf{RD}_2}{\vdash S : \mathsf{RD}_1 \to \mathsf{RD}_2} \textsc{Sub}$$



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

Meaning of annotated judgments

"Meaning" of judgment $S : \mathsf{RD}_1 \to \mathsf{RD}_2$:

"RD_1 is *the* set of var/label reaching the entry of S and RD_2 the corresponding set at the exit(s) of S":

$$\begin{array}{lcl} \mathsf{RD}_1 & = & \mathsf{RD}_{entry}(init(S)) \\ \mathsf{RD}_2 & = & \bigcup\{\mathsf{RD}_{exit} l \mid l \in final(S)\} \end{array}$$

- Be careful:

$$\texttt{if}\,[b]^l\,\texttt{then}\,S_1\,\texttt{else}\,S_2$$

- more concretely

$$\texttt{if}\,[b]^l\,\texttt{then}\,[x := y]^{l_1}\,\texttt{else}\,[y := x]^{l_2}$$



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Meaning of annotated judgments

Once again: "Meaning" of judgment $S : \mathsf{RD}_1 \to \mathsf{RD}_2$:

"if RD_1 is a set of var/label reaching the entry of S, then RD_2 is a corresponding set at the exit(s) of S":

$$\begin{array}{rrcl} & \text{if} \quad \mathsf{RD}_1 & \subseteq & \mathsf{RD}_{entry}(init(S)) \\ \text{then} \quad \forall l \in final(S).\ \mathsf{RD}_{exit}(l) & & \subseteq & \mathsf{RD}_2 \end{array}$$



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

Derivation

$$\dfrac{[y := x]^0 : \mathsf{RD}_0 \to \{?_x, 0, ?_z\} \qquad \dfrac{[z := 1]^1 : \{?_x, 0, ?_z\} \to \{?_x, 0, 1\} \quad f_3 : \{?_x, 0, 1\} \to \mathsf{RD}_{final}}{f_2 : \{?_x, 0, ?_z\} \to \mathsf{RD}_{final}}}{f : \mathsf{RD}_0 \to \mathsf{RD}_{final}}$$

$$\mathsf{RD}_0 = \{?_x, ?_y, ?_z\} \quad \mathsf{RD}_{final} = \{?_x, 5, 1, 3\}$$

type sub-derivation for the rest $f_3 = \texttt{while} \ldots ; [y := 0]^5$
loop invariant

$$\mathsf{RD}_{body} = \{?_x, 0, 4, 1, 3\}$$

Derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{[z := _]^3 : \mathsf{RD}_{body} \to \{?_x, 0, 4, 3, 1\} \quad [y := _]^4 : \{?_x, 0, 4, 3\} \to \{?_x, 4, 3\}}{f_{body} : \mathsf{RD}_{body} \to \{?_x, 4, 3\}}}{f_{body} : \mathsf{RD}_{body} \to \mathsf{RD}_{body}}\ \textsc{Sub}}{f_{while} : \mathsf{RD}_{body} \to \mathsf{RD}_{body}}}{f_{while} : \{?_x, 0, 1\} \to \mathsf{RD}_{body}}\ \textsc{Sub} \qquad [y := 0]^5 : \mathsf{RD}_{body} \to \mathsf{RD}_{fi}$$

$$f_3 : \{?_x, 0, 1\} \to \mathsf{RD}_{final}$$

Annotated type constructors

- alternative approach of annotated type systems
- arrow constructor itself *annotated*
- annotion of $\rightarrow$: flavor of effect system
- judgment

$$S : \Sigma \xrightarrow[\text{RD}]{} \Sigma$$

- annotation with RD (corresponding to the post-condition from above) alone is *not enough*



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

Annotated type constructors

- alternative approach of annotated type systems
- arrow constructor itself *annotated*
- annotion of $\rightarrow$: flavor of effect system
- judgment

$$S : \Sigma \xrightarrow[\text{RD}]{X} \Sigma$$

- annotation with RD (corresponding to the post-condition from above) alone is *not enough*
- also needed: the *variables "being" changed*

Intended meaning

"S maps states to states, where RD is the set of reaching definitions, S may produce and X the set of var's S must (= unavoidably) assign.



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

$$[x := a]^l : \Sigma \xrightarrow[\{(x,l)\}]{\{x\}} \Sigma \qquad \textsc{Ass} \qquad\qquad [\mathsf{skip}]^l : \Sigma \xrightarrow[\emptyset]{\emptyset} \Sigma \qquad \textsc{Skip}$$

$$\frac{S_1 : \Sigma \xrightarrow[\mathsf{RD}_1]{X_1} \Sigma \qquad S_2 : \Sigma \xrightarrow[\mathsf{RD}_2]{X_2} \Sigma}{S_1; S_2 : \Sigma \xrightarrow[\mathsf{RD}_1 \setminus X_2 \cup \mathsf{RD}_2]{X_1 \cup X_2} \Sigma} \textsc{Seq}$$

$$\frac{S_1 : \Sigma \xrightarrow[\mathsf{RD}]{X} \Sigma \qquad S_2 : \Sigma \xrightarrow[\mathsf{RD}]{X} \Sigma}{\mathtt{if}[b]^l \,\mathtt{then}\, S_1 \,\mathtt{else}\, S_2 : \Sigma \xrightarrow[\mathsf{RD}]{X} \Sigma} \textsc{If}$$

$$\frac{S : \Sigma \xrightarrow[\mathsf{RD}]{X} \Sigma}{\mathtt{while}[b]^l \,\mathtt{do}\, S : \Sigma \xrightarrow[\mathsf{RD}]{\emptyset} \Sigma} \textsc{While}$$

$$\frac{S : \Sigma \xrightarrow[\mathsf{RD}']{X'} \Sigma \qquad X \subseteq X' \qquad \mathsf{RD}' \subseteq \mathsf{RD}}{S : \Sigma \xrightarrow[\mathsf{RD}]{X} \Sigma} \textsc{Sub}$$



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

Effect systems

- this time: back to the *functional* language
- starting point: simple type system
- *judgment*:

$$\Gamma \vdash e : \tau$$

- Γ: *type environment* (or context), "mapping" from variable to types
- types: bool, int, and $\tau \to \tau$



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}\ \textsc{Var}$$

$$\frac{\Gamma, x{:}\tau_1 \vdash e : \tau_2}{\Gamma \vdash \ \mathtt{fn}_{\,\pi} x \Rightarrow e : \tau_1 \to \tau_2}\ \textsc{Abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \to \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1\ e_2 : \tau_2}\ \textsc{App}$$



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

Effects: Call tracking analysis

Call tracking analysis:

Determine: for each subexpression: which function abstractions may be applied, i.e., called, during the subexpression's evaluation.

$\Rightarrow$ set of function names

annotate: function type with latent effect

$\Rightarrow$ *annotated* types: $\hat{\tau}$: base types as before, arrow types:

$$\hat{\tau}_1 \xrightarrow{\varphi} \hat{\tau}_2 \tag{16}$$

- functions from τ_1 to τ_2, where in the execution, functions from set φ are called.

Judgment

$$\hat{\Gamma} \vdash e : \hat{\tau} :: \varphi \tag{17}$$



Targets & Outline

Motivation

General remarks

Data-flow analysis

A simplistic while-language

Equational approach

Constraint-based approach

Control-flow analysis (constraint-based)

Control-flow analysis

Type and effect systems

Introduction

Annotated type systems

Annotated type constructors

Effect systems

Algorithms

$$\frac{\hat{\Gamma}(x) = \hat{\tau}}{\hat{\Gamma} \vdash x : \hat{\tau} :: \emptyset}\ \textsc{Var}$$

$$\frac{\Gamma, x{:}\hat{\tau}_1 \vdash e : \hat{\tau}_2 :: \varphi}{\Gamma \vdash \mathtt{fn}_\pi x \Rightarrow e : \hat{\tau}_1 \overset{\varphi \cup \{\pi\}}{\rightarrow} \hat{\tau}_2 :: \emptyset}\ \textsc{Abs}$$

$$\frac{\hat{\Gamma} \vdash e_1 : \hat{\tau}_1 \overset{\varphi}{\rightarrow} \hat{\tau}_2 :: \varphi_1 \qquad \hat{\Gamma} \vdash e_2 : \hat{\tau}_1 :: \varphi_2}{\hat{\Gamma} \vdash e_1\ e_2 : \hat{\tau}_2 :: \varphi \cup \varphi_1 \cup \varphi_2}\ \textsc{App}$$



Call tracking: example

$$\dfrac{\dfrac{x{:}\mathsf{int} \overset{\{Y\}}{\to} \mathsf{int} \vdash x{:}\mathsf{int} \overset{\{Y\}}{\to} \mathsf{int} :: \emptyset}{\vdash (\mathtt{fn}_X\, x \Rightarrow x) : (\mathsf{int} \overset{\{Y\}}{\to} \mathsf{int}) \overset{\{X\}}{\to} (\mathsf{int} \overset{\{Y\}}{\to} \mathsf{int}) :: \emptyset} \qquad \vdash (\mathtt{fn}_Y\, y \Rightarrow y) : \mathsf{int} \overset{\{Y\}}{\to} \mathsf{int} :: \emptyset}{\vdash (\mathtt{fn}_X\, x \Rightarrow x)\ (\mathtt{fn}_Y\, y \Rightarrow y) : \mathsf{int} \overset{\{Y\}}{\to} \mathsf{int} :: \{X\}}$$



Section

Algorithms

Chapter 1 “Introduction”
Course “Static analysis and all that”
Martin Steffen
IN5440 / autumn 2024

Chaotic iteration

- back to data flow/reaching def's
- goal: solve

$$\vec{RD} = F(\vec{\mathsf{RD}}) \quad \text{or} \quad \vec{RD} \sqsubseteq F(\vec{\mathsf{RD}})$$

- F: monotone, finite domain

straightforward approach

init $\vec{\mathsf{RD}}_0 = F^0(\emptyset)$

iterate $\vec{\mathsf{RD}}_{n+1} = F(\vec{\mathsf{RD}}_n) = F^{n+1}(\emptyset)$ until stabilization

- approach to implement that: chaotic iteration
- non-deterministic stategy
- abbreviate:

$$\vec{\mathsf{RD}} \quad = \quad (\mathsf{RD}_1, \ldots, \mathsf{RD}_{12})$$



Targets & Outline

Motivation
General remarks

Data-flow analysis
A simplistic while-language
Equational approach
Constraint-based approach

Control-flow analysis (constraint-based)
Control-flow analysis

Type and effect systems
Introduction
Annotated type systems
Annotated type constructors
Effect systems

Algorithms

Chaotic iteration (for RD)

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**

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

References I

Bibliography

[1] Flemming Nielson, Hanne-Riis Nielson, and Chris L. Hankin. *Principles of Program Analysis*. Springer Verlag, 1999.

