Principles of Program Analysis:

A Sampler of Approaches

Transparencies based on Chapter 1 of the book: Flemming Nielson, Hanne Riis Nielson and Chris Hankin: Principles of Program Analysis. Springer Verlag 2005.

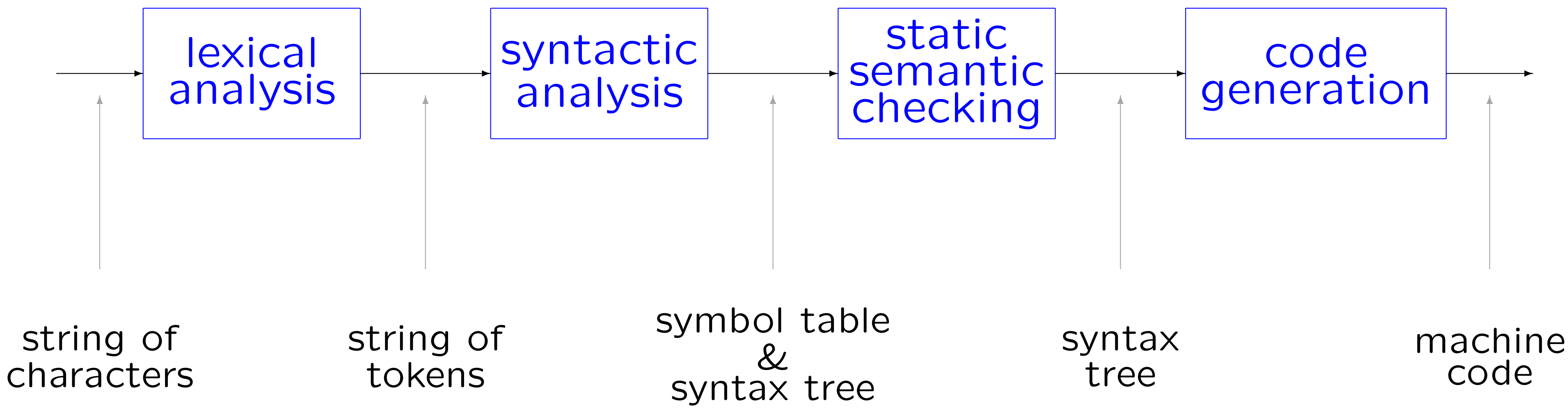
Compiler Optimisation

The classical use of program analysis is to facilitate the construction of compilers generating "optimal" code.

We begin by outlining the structure of optimising compilers.

We then prepare the setting for a worked example where we "optimise" a naive implementation of Algol-like arrays in a C-like language by performing a series of analyses and transformations.

The structure of a simple compiler

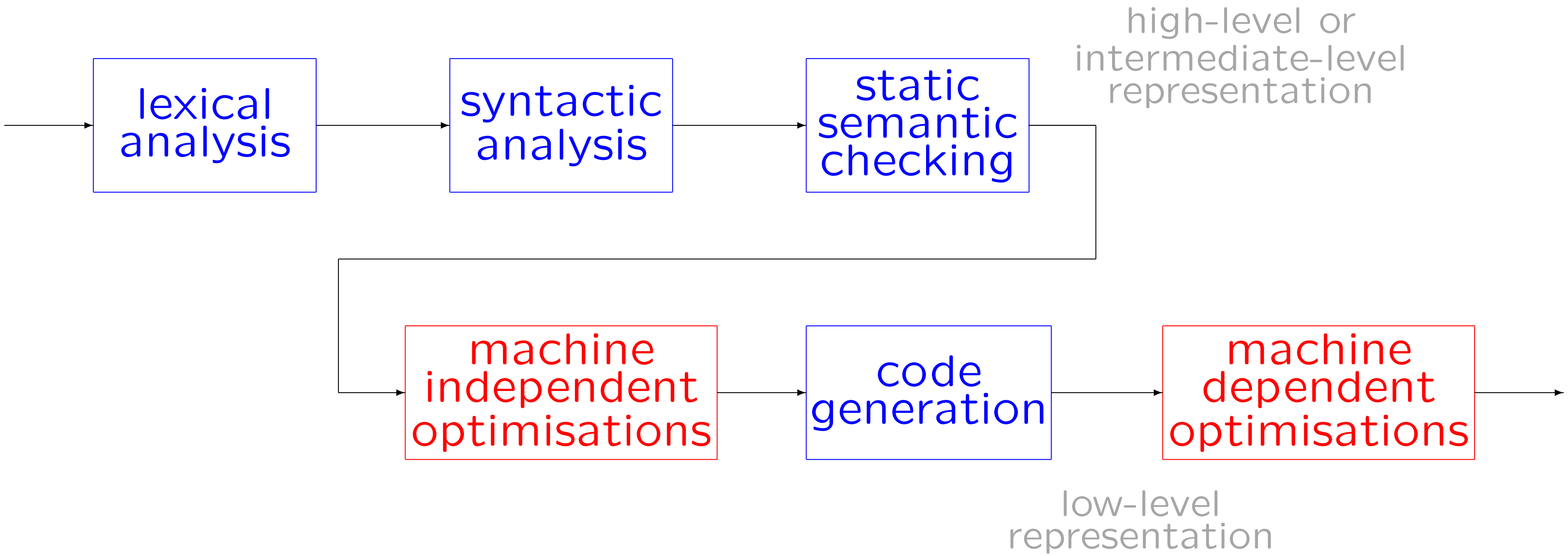


Characteristics of a simple compiler:

- many phases – one or more passes

- the compiler is fast – but the code is not very efficient

The structure of an optimising compiler

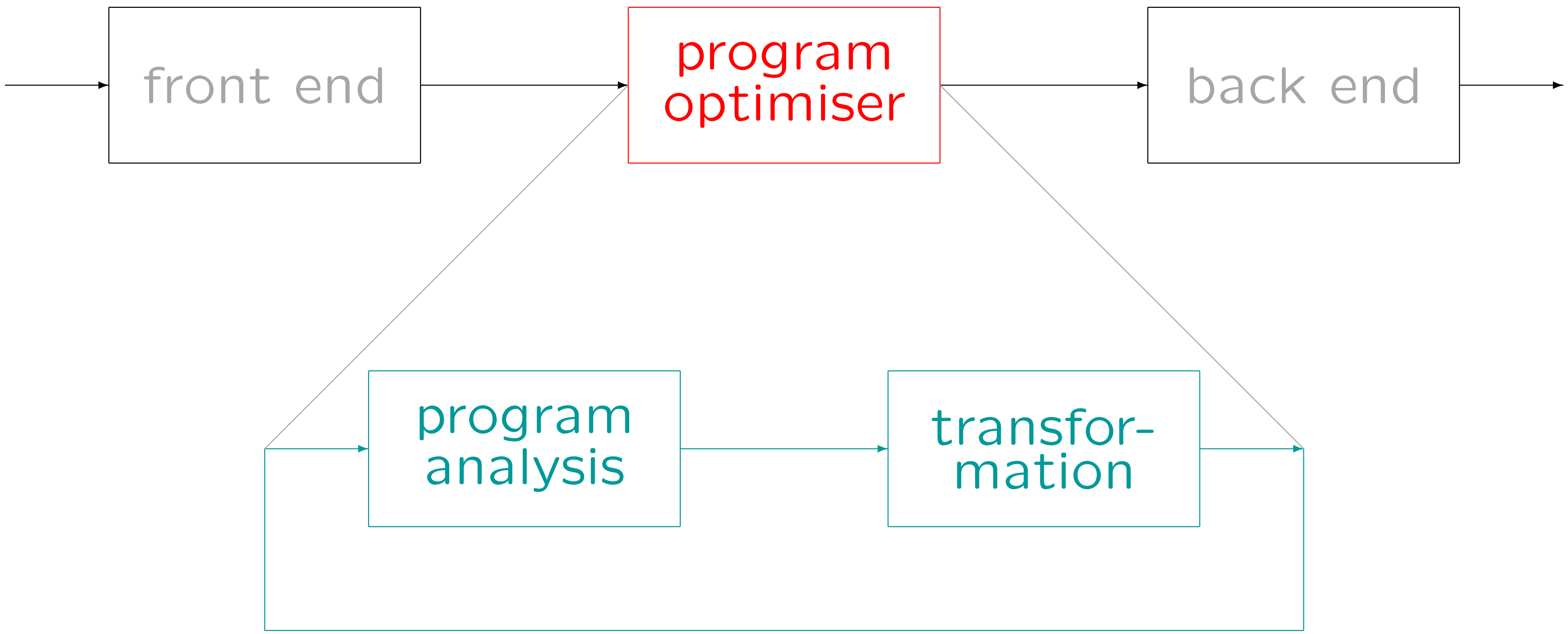


Characteristics of the optimising compiler:

- high-level optimisations: easy to adapt to new architectures

- low-level optimisations: less likely to port to new architectures

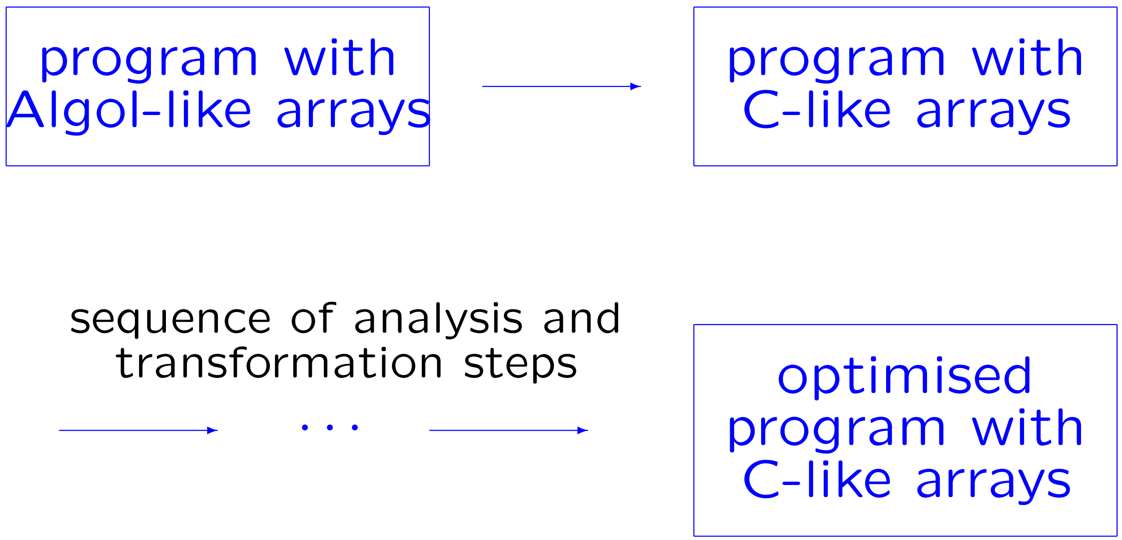
The structure of the optimisation phase



Avoid redundant computations: reuse available results, move loop invariant computations out of loops, ...

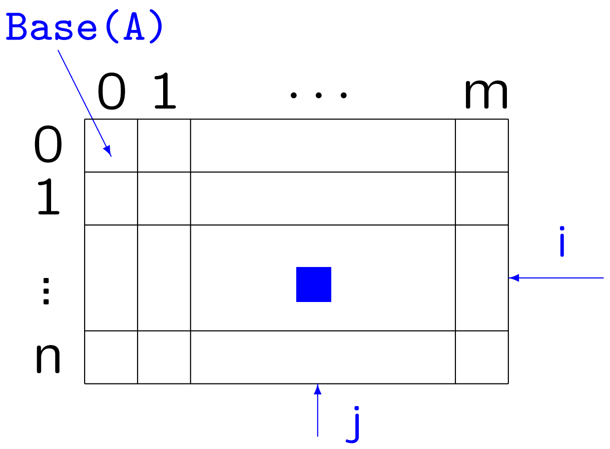
Avoid superfluous computations: results known not to be needed, results known already at compile time, ...

Example: Array Optimisation



Array representation: Algol vs. C

`A: array [0:n, 0:m] of integer`



Accessing the (i,j)'th element of `A`:

in Algol:

`A[i,j]`

in C:

`Cont(Base(A) + i * (m+1) + j)`

An example program and its naive realisation

Algol-like arrays:

```
i := 0;
while i <= n do
   j := 0;
   while j <= m do
      A[i,j] := B[i,j] + C[i,j];
      j := j+1
   od;
   i := i+1
od
```

C-like arrays:

```
i := 0;
while i <= n do
   j := 0;
   while j <= m do
      temp := Base(A) + i * (m+1) + j;
      Cont(temp) := Cont(Base(B) + i * (m+1) + j)
                    + Cont(Base(C) + i * (m+1) + j);
      j := j+1
   od;
   i := i+1
od
```

Available Expressions analysis and Common Subexpression Elimination

```
i := 0;
while i <= n do
   j := 0;
   while j <= m do
      temp := Base(A) + i*(m+1) + j;
      Cont(temp) := Cont(Base(B) + i*(m+1) + j)
                  + Cont(Base(C) + i*(m+1) + j);
      j := j+1
   od;
   i := i+1
od
```

first computation

re-computations

```
t1 := i * (m+1) + j;
temp := Base(A) + t1;
Cont(temp) := Cont(Base(B)+t1)
            + Cont(Base(C)+t1);
```

Detection of Loop Invariants and Invariant Code Motion

loop invariant

```
i := 0;
while i <= n do
   j := 0;
   while j <= m do
      t1 := i * (m+1) + j;
      temp := Base(A) + t1;
      Cont(temp) := Cont(Base(B) + t1)
                    + Cont(Base(C) + t1);
      j := j+1
   od;
   i := i+1
od
```

```
t2 := i * (m+1);
while j <= m do
   t1 := t2 + j;
   temp := ...
   Cont(temp) := ...
   j := ...
od
```

Detection of Induction Variables and Reduction of Strength

```
i := 0;
while i <= n do
   j := 0;
   t2 := i * (m+1);
   while j <= m do
      t1 := t2 + j;
      temp := Base(A) + t1;
      Cont(temp) := Cont(Base(B) + t1)
                    + Cont(Base(C) + t1);
      j := j+1
   od;
   i := i+1
od
```

induction variable

```
i := 0;
t3 := 0;
while i <= n do
   j := 0;
   t2 := t3;
   while j <= m do ...  od
   i := i + 1;
   t3 := t3 + (m+1)
od
```

Equivalent Expressions analysis and Copy Propagation

```
i := 0;
t3 := 0;
while i <= n do
   j := 0;
   t2 := t3;
   while j <= m do
      t1 := t2 + j;
      temp := Base(A) + t1;
      Cont(temp) := Cont(Base(B) + t1)
                    + Cont(Base(C) + t1);
      j := j+1
   od;
   i := i+1;
   t3 := t3 + (m+1)
od
```

t2 = t3

```
while j <= m do
   t1 := t3 + j;
   temp := ...;
   Cont(temp) := ...;
   j := ...
od
```

Live Variables analysis and Dead Code Elimination

```
i := 0;
t3 := 0;
while i <= n do
   j := 0;
   t2 := t3;
   while j <= m do
      t1 := t3 + j;
      temp := Base(A) + t1;
      Cont(temp) := Cont(Base(B) + t1)
                    + Cont(Base(C) + t1);
      j := j+1
   od;
   i := i+1;
   t3 := t3 + (m+1)
od
```

dead variable

```
i := 0;
t3 := 0;
while i <= n do
   j := 0;
   while j <= m do
      t1 := t3 + j;
      temp := Base(A) + t1;
      Cont(temp) := Cont(Base(B) + t1)
                    + Cont(Base(C) + t1);
      j := j+1
   od;
   i := i+1;
   t3 := t3 + (m+1)
od
```

Summary of analyses and transformations

Analysis	Transformation
Available expressions analysis	Common subexpression elimination
Detection of loop invariants	Invariant code motion
Detection of induction variables	Strength reduction
Equivalent expression analysis	Copy propagation
Live variables analysis	Dead code elimination

The Essence of Program Analysis

Program analysis offers techniques for predicting
statically at compile-time
safe & efficient approximations
to the set of configurations or behaviours arising
dynamically at run-time

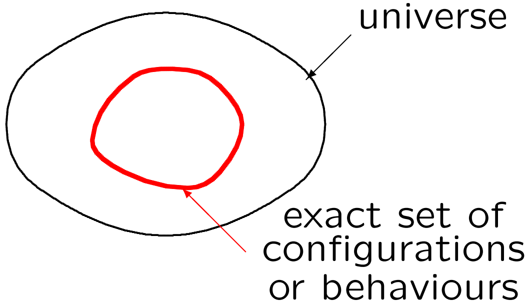
we cannot expect exact answers!

Safe: faithful to the semantics

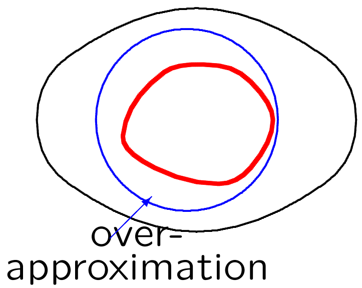
Efficient: implementation with
- good time performance and
- low space consumption

The Nature of Approximation

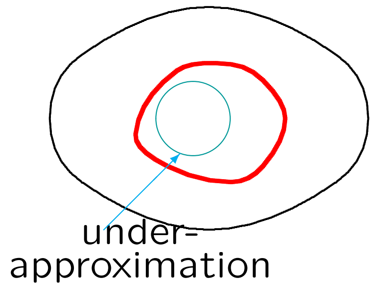
The exact world



Over-approximation



Under-approximation



Slogans: Err on the safe side!
Trade precision for efficiency!

Approaches to Program Analysis

A family of techniques . . .

- data flow analysis
- constraint based analysis
- abstract interpretation
- type and effect systems
- . . .
- flow logic:
  a unifying framework

. . . that differ in their focus:

- algorithmic methods
- semantic foundations
- language paradigms
  — imperative/procedural
  — object oriented
  — logical
  — functional
  — concurrent/distributive
  — mobile
  — . . .

Data Flow Analysis

- **Technique:** Data Flow Analysis
- **Example:** Reaching Definitions analysis
  - idea
  - constructing an equation system
  - solving the equations
  - theoretical underpinnings

Example program

Program with labels for elementary blocks:

$[y := x]^1$;

$[z := 1]^2$;

while $[y > 0]^3$ **do**

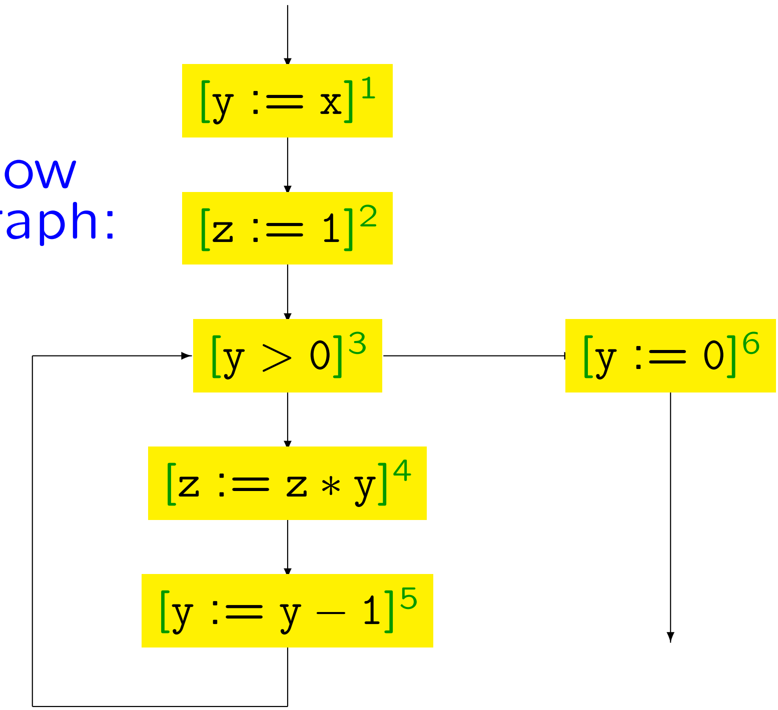
$[z := z * y]^4$;

$[y := y - 1]^5$

od;

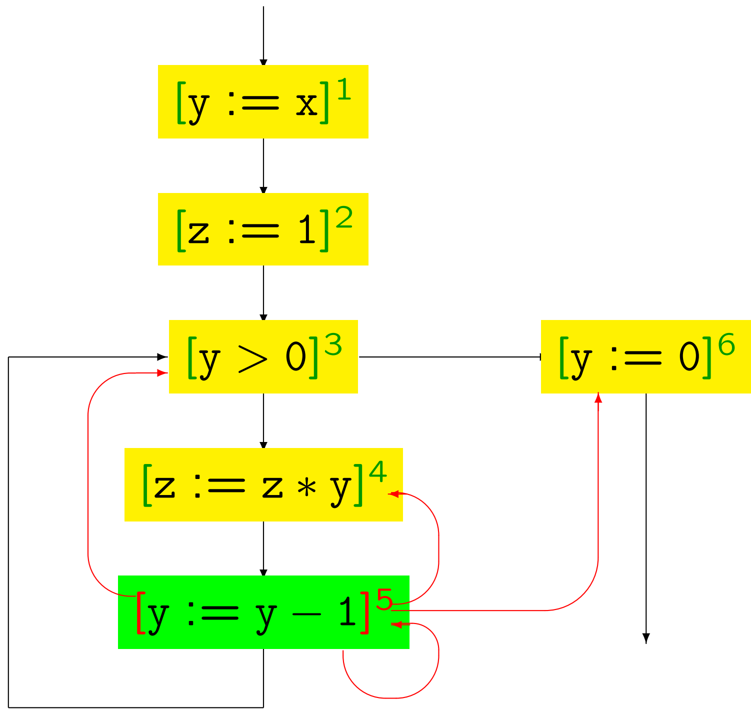
$[y := 0]^6$

Flow graph:



Example: Reaching Definitions

The assignment $[x := a]^\ell$ reaches ℓ' if there is an execution where x was last assigned at ℓ



Reaching Definitions analysis (1)

Program		Reaching definitions
	←	$\{(x,?),(y,?),(z,?)\}$
$[y := x]^1$;		
	←	$\{(x,?),(y,1),(z,?)\}$
$[z := 1]^2$;		
	←	$\{(x,?),(y,1),(z,2)\}$
while $[y > 0]^3$ do		
	←	$\{(x,?),(y,1),(z,2)\}$
$[z := z * y]^4$;		
	←	
$[y := y - 1]^5$		
	←	
od;		
	←	$\{(x,?),(y,1),(z,2)\}$
$[y := 0]^6$		
	←	

Reaching Definitions analysis (2)

	$\leftarrow$	$\{(x,?),(y,?),(z,?)\}$
$[y := x]^1$;		
	$\leftarrow$	$\{(x,?),(y,1),(z,?)\}$
$[z := 1]^2$;		
	$\leftarrow$	$\{(x,?),(y,1),(z,2)\} \cup \{(y,5),(z,4)\}$
while $[y > 0]^3$ **do**		
	$\leftarrow$	$\{(x,?),(y,1),(z,2)\}$
$[z := z * y]^4$;		
	$\leftarrow$	$\{(x,?),(y,1),(z,4)\}$
$[y := y - 1]^5$		
	$\leftarrow$	$\{(x,?),(y,5),(z,4)\}$
od;		
	$\leftarrow$	$\{(x,?),(y,1),(z,2)\}$
$[y := 0]^6$		
	$\leftarrow$	

Reaching Definitions analysis (3)

$\longleftarrow \{(x,?),(y,?),(z,?)\}$

$[y := x]^1;$

$\longleftarrow \{(x,?),(y,1),(z,?)\}$

$[z := 1]^2;$

$\longleftarrow \{(x,?),(y,1),(y,5),(z,2),(z,4)\} \cup \{(y,5),(z,4)\}$

$\texttt{while}\ [y > 0]^3\ \texttt{do}$

$\longleftarrow \{(x,?),(y,1),(y,5),(z,2),(z,4)\}$

$\quad [z := z * y]^4;$

$\longleftarrow \{(x,?),(y,1),(y,5),(z,4)\}$

$\quad [y := y - 1]^5$

$\longleftarrow \{(x,?),(y,5),(z,4)\}$

$\texttt{od};$

$\longleftarrow \{(x,?),(y,1),(y,5),(z,2),(z,4)\}$

$[y := 0]^6$

$\longleftarrow$

The best solution

$$
\begin{array}{ll}
 & \longleftarrow \{(x,?),(y,?),(z,?)\} \\
[y := x]^1; & \\
 & \longleftarrow \{(x,?),(y,1),(z,?)\} \\
[z := 1]^2; & \\
 & \longleftarrow \{(x,?),(y,1),(y,5),(z,2),(z,4)\} \\
\texttt{while}\ [y > 0]^3\ \texttt{do} & \\
 & \longleftarrow \{(x,?),(y,1),(y,5),(z,2),(z,4)\} \\
\quad [z := z * y]^4; & \\
 & \longleftarrow \{(x,?),(y,1),(y,5),(z,4)\} \\
\quad [y := y - 1]^5 & \\
 & \longleftarrow \{(x,?),(y,5),(z,4)\} \\
\texttt{od}; & \\
 & \longleftarrow \{(x,?),(y,1),(y,5),(z,2),(z,4)\} \\
[y := 0]^6 & \\
 & \longleftarrow \{(x,?),(y,6),(z,2),(z,4)\}
\end{array}
$$

A safe solution — but not the best

$\{(x,?),(y,?),(z,?)\}$

$[y := x]^1$;

$\{(x,?),(y,1),(z,?)\}$

$[z := 1]^2$;

$\{(x,?),(y,1),(y,5),(z,2),(z,4)\}$

while $[y > 0]^3$ **do**

$\{(x,?),(y,1),(y,5),(z,2),(z,4)\}$

  $[z := z * y]^4$;

$\{(x,?),(y,1),(y,5),\boxed{(z,2)},(z,4)\}$

  $[y := y - 1]^5$

$\{(x,?),\boxed{(y,1)},(y,5),\boxed{(z,2)},(z,4)\}$

od;

$\{(x,?),(y,1),(y,5),(z,2),(z,4)\}$

$[y := 0]^6$

$\{(x,?),(y,6),(z,2),(z,4)\}$

An unsafe solution

$$\{(x,?),(y,?),(z,?)\}$$

$[y := x]^1;$

$$\{(x,?),(y,1),(z,?)\}$$

$[z := 1]^2;$

$$\{(x,?),(y,1),(z,2),(y,5),(z,4)\}$$

while $[y > 0]^3$ **do**

$$\{(x,?),\boxed{(y,1)},(y,5),\boxed{(z,2)},(z,4)\}$$

$\quad [z := z * y]^4;$

$$\{(x,?),\boxed{(y,1)},(y,5),(z,4)\}$$

$\quad [y := y - 1]^5$

$$\{(x,?),(y,5),(z,4)\}$$

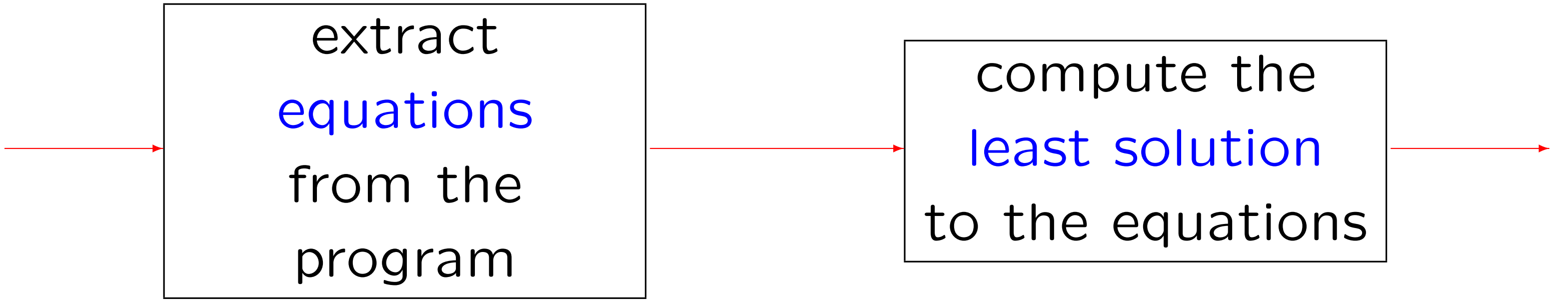
od;

$$\{(x,?),\boxed{(y,1)},(y,5),\boxed{(z,2)},(z,4)\}$$

$[y := 0]^6$

$$\{(x,?),(y,6),(z,2),(z,4)\}$$

How to automate the analysis

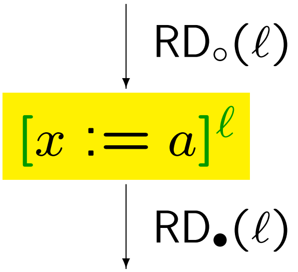


Analysis information:

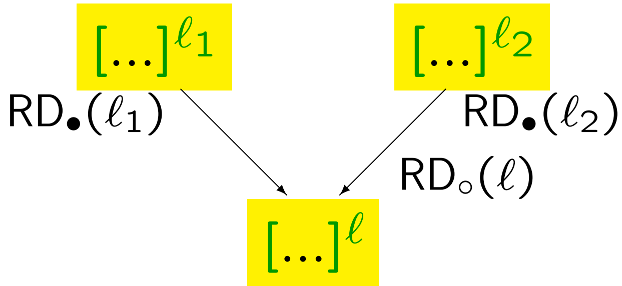
- $\mathsf{RD}_\circ(\ell)$: information available at the entry of block ℓ

- $\mathsf{RD}_\bullet(\ell)$: information available at the exit of block ℓ

Two kinds of equations



$$RD_\circ(\ell) \setminus \{(x,\ell') \mid \ell' \in \mathbf{Lab}\} \cup \{(x,\ell)\} = RD_\bullet(\ell)$$



$$RD_\bullet(\ell_1) \cup RD_\bullet(\ell_2) = RD_\circ(\ell)$$

Flow through assignments and tests

$[y := x]^1$;	$RD_\bullet(1) = RD_\circ(1) \setminus \{(y,\ell) \mid \ell \in \mathbf{Lab}\} \cup \{(y,1)\}$
$[z := 1]^2$;	$RD_\bullet(2) = RD_\circ(2) \setminus \{(z,\ell) \mid \ell \in \mathbf{Lab}\} \cup \{(z,2)\}$
`while` $[y > 0]^3$ `do`	$RD_\bullet(3) = RD_\circ(3)$
$[z := z * y]^4$;	$RD_\bullet(4) = RD_\circ(4) \setminus \{(z,\ell) \mid \ell \in \mathbf{Lab}\} \cup \{(z,4)\}$
$[y := y - 1]^5$	$RD_\bullet(5) = RD_\circ(5) \setminus \{(y,\ell) \mid \ell \in \mathbf{Lab}\} \cup \{(y,5)\}$
`od`;	
$[y := 0]^6$	$RD_\bullet(6) = RD_\circ(6) \setminus \{(y,\ell) \mid \ell \in \mathbf{Lab}\} \cup \{(y,6)\}$

Lab $= \{1,2,3,4,5,6\}$

6 equations in $RD_\circ(1), \cdots, RD_\bullet(6)$

Flow along the control

$[y := x]^1;$

$[z := 1]^2;$

$\texttt{while}\ [y > 0]^3\ \texttt{do}$

$\quad [z := z * y]^4;$

$\quad [y := y - 1]^5$

$\texttt{od};$

$[y := 0]^6$

$\mathsf{RD}_\circ(1) = \{(x, ?), (y, ?), (z, ?)\}$

$\mathsf{RD}_\circ(2) = \mathsf{RD}_\bullet(1)$

$\mathsf{RD}_\circ(3) = \mathsf{RD}_\bullet(2) \cup \mathsf{RD}_\bullet(5)$

$\mathsf{RD}_\circ(4) = \mathsf{RD}_\bullet(3)$

$\mathsf{RD}_\circ(5) = \mathsf{RD}_\bullet(4)$

$\mathsf{RD}_\circ(6) = \mathsf{RD}_\bullet(3)$

$\mathbf{Lab} = \{1,2,3,4,5,6\}$

6 equations in $\mathsf{RD}_\circ(1), \cdots, \mathsf{RD}_\bullet(6)$

Summary of equation system

$$\mathsf{RD}_\bullet(1) = \mathsf{RD}_\circ(1) \setminus \{(y,\ell) \mid \ell \in \mathbf{Lab}\} \cup \{(y,1)\}$$
$$\mathsf{RD}_\bullet(2) = \mathsf{RD}_\circ(2) \setminus \{(z,\ell) \mid \ell \in \mathbf{Lab}\} \cup \{(z,2)\}$$
$$\mathsf{RD}_\bullet(3) = \mathsf{RD}_\circ(3)$$
$$\mathsf{RD}_\bullet(4) = \mathsf{RD}_\circ(4) \setminus \{(z,\ell) \mid \ell \in \mathbf{Lab}\} \cup \{(z,4)\}$$
$$\mathsf{RD}_\bullet(5) = \mathsf{RD}_\circ(5) \setminus \{(y,\ell) \mid \ell \in \mathbf{Lab}\} \cup \{(y,5)\}$$
$$\mathsf{RD}_\bullet(6) = \mathsf{RD}_\circ(6) \setminus \{(y,\ell) \mid \ell \in \mathbf{Lab}\} \cup \{(y,6)\}$$

$$\mathsf{RD}_\circ(1) = \{(x,?),(y,?),(z,?)\}$$
$$\mathsf{RD}_\circ(2) = \mathsf{RD}_\bullet(1)$$
$$\mathsf{RD}_\circ(3) = \mathsf{RD}_\bullet(2) \cup \mathsf{RD}_\bullet(5)$$
$$\mathsf{RD}_\circ(4) = \mathsf{RD}_\bullet(3)$$
$$\mathsf{RD}_\circ(5) = \mathsf{RD}_\bullet(4)$$
$$\mathsf{RD}_\circ(6) = \mathsf{RD}_\bullet(3)$$

- 12 sets: $\mathsf{RD}_\circ(1), \cdots, \mathsf{RD}_\bullet(6)$

  all being subsets of $\mathbf{Var} \times \mathbf{Lab}$
- 12 equations:

  $\mathsf{RD}_j = \mathsf{F}_j(\mathsf{RD}_\circ(1), \cdots, \mathsf{RD}_\bullet(6))$
- one function:

  $\mathsf{F} : \mathcal{P}(\mathbf{Var} \times \mathbf{Lab})^{12} \to \mathcal{P}(\mathbf{Var} \times \mathbf{Lab})^{12}$
- we want the least fixed point of F — this is the best solution to the equation system

How to solve the equations

A simple iterative algorithm

- Initialisation
  $\mathsf{RD}_1 := \emptyset; \cdots ; \mathsf{RD}_{12} := \emptyset;$

- Iteration
  while $\mathsf{RD}_j \neq \mathsf{F}_j(\mathsf{RD}_1, \cdots, \mathsf{RD}_{12})$ for some j
  do
  $\quad \mathsf{RD}_j := \mathsf{F}_j(\mathsf{RD}_1, \cdots, \mathsf{RD}_{12})$

The algorithm terminates and computes the least fixed point of F.

The example equations

$RD_\circ$	1	2	3	4	5	6
0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
1	$x_?, y_?, z_?$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
2	$x_?, y_?, z_?$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
3	$x_?, y_?, z_?$	$x_?, y_1, z_?$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
4	$x_?, y_?, z_?$	$x_?, y_1, z_?$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
5	$x_?, y_?, z_?$	$x_?, y_1, z_?$	$x_?, y_1, z_2$	$\emptyset$	$\emptyset$	$\emptyset$
6	$x_?, y_?, z_?$	$x_?, y_1, z_?$	$x_?, y_1, z_2$	$\emptyset$	$\emptyset$	$\emptyset$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

$RD_\bullet$	1	2	3	4	5	6
0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
2	$x_?, y_1, z_?$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
3	$x_?, y_1, z_?$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
4	$x_?, y_1, z_?$	$x_?, y_1, z_2$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
5	$x_?, y_1, z_?$	$x_?, y_1, z_2$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
6	$x_?, y_1, z_?$	$x_?, y_1, z_2$	$x_?, y_1, z_2$	$\emptyset$	$\emptyset$	$\emptyset$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

The equations:

$$\begin{aligned}
RD_\bullet(1) &= RD_\circ(1) \setminus \{(y,\ell) \mid \cdots\} \cup \{(y,1)\}\\
RD_\bullet(2) &= RD_\circ(2) \setminus \{(z,\ell) \mid \cdots\} \cup \{(z,2)\}\\
RD_\bullet(3) &= RD_\circ(3)\\
RD_\bullet(4) &= RD_\circ(4) \setminus \{(z,\ell) \mid \cdots\} \cup \{(z,4)\}\\
RD_\bullet(5) &= RD_\circ(5) \setminus \{(y,\ell) \mid \cdots\} \cup \{(y,5)\}\\
RD_\bullet(6) &= RD_\circ(6) \setminus \{(y,\ell) \mid \cdots\} \cup \{(y,6)\}
\end{aligned}$$

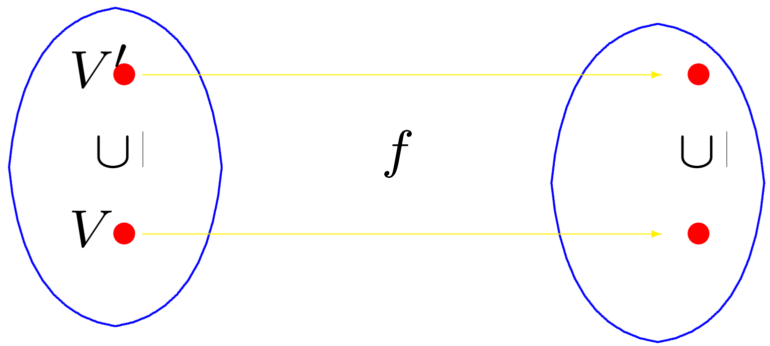
$$\begin{aligned}
RD_\circ(1) &= \{(x,?),(y,?),(z,?)\}\\
RD_\circ(2) &= RD_\bullet(1)\\
RD_\circ(3) &= RD_\bullet(2) \cup RD_\bullet(5)\\
RD_\circ(4) &= RD_\bullet(3)\\
RD_\circ(5) &= RD_\bullet(4)\\
RD_\circ(6) &= RD_\bullet(3)
\end{aligned}$$

Why does it work? (1)

A function $f : \mathcal{P}(S) \to \mathcal{P}(S)$ is a monotone function if

$$V \subseteq V' \quad \Rightarrow \quad f(V) \subseteq f(V')$$

(the larger the argument – the larger the result)



Why does it work? (2)

A set L equipped with an ordering $\subseteq$ satisfies the Ascending Chain Condition if all chains

$$V_0 \subseteq V_1 \subseteq V_2 \subseteq V_3 \subseteq \cdots$$

stabilise, that is, if there exists some n such that $V_n = V_{n+1} = V_{n+2} = \cdots$

If S is a finite set then $\mathcal{P}(S)$ equipped with the subset ordering $\subseteq$ satisfies the Ascending Chain Condition — the chains cannot grow forever since each element is a subset of a finite set.

Fact

For a given program $\mathbf{Var} \times \mathbf{Lab}$ will be a finite set so $\mathcal{P}(\mathbf{Var} \times \mathbf{Lab})$ with the subset ordering satisfies the Ascending Chain Condition.

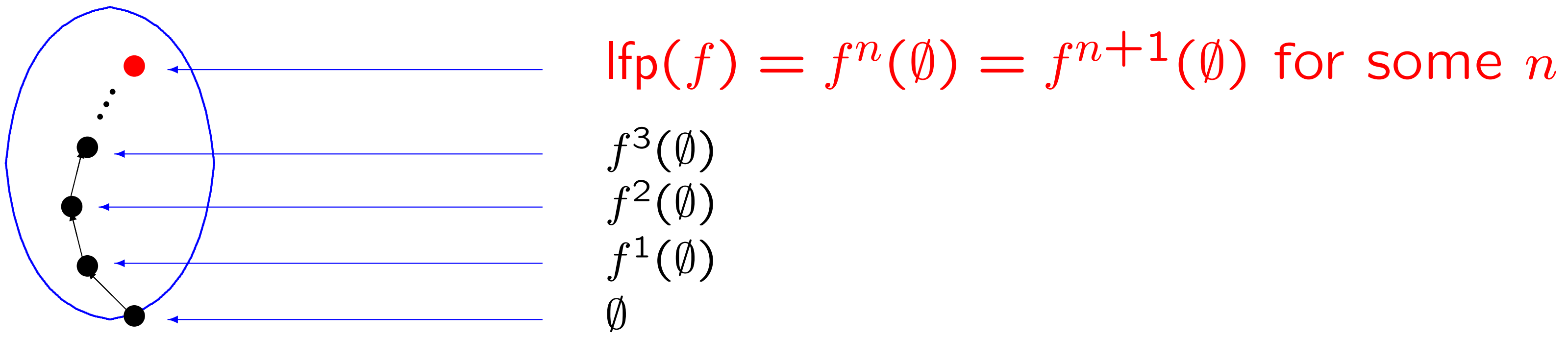
Why does it work? (3)

Let $f : \mathcal{P}(S) \rightarrow \mathcal{P}(S)$ be a monotone function. Then

$$\emptyset \subseteq f(\emptyset) \subseteq f^2(\emptyset) \subseteq f^3(\emptyset) \subseteq \cdots$$

Assume that S is a finite set; then the Ascending Chain Condition is satisfied. This means that the chain cannot be growing infinitely so there exists n such that $f^n(\emptyset) = f^{n+1}(\emptyset) = \cdots$

$f^n(\emptyset)$ is the least fixed point of f



Correctness of the algorithm

- Initialisation

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

  Invariant: $\vec{\mathsf{RD}} \sqsubseteq \mathsf{F}^n(\vec{\emptyset})$ since $\vec{\mathsf{RD}} = \vec{\emptyset}$ is the least element

- Iteration

  while $\mathsf{RD}_j \neq \mathsf{F}_j(\mathsf{RD}_1, \cdots, \mathsf{RD}_{12})$ for some j

  do assume $\vec{\mathsf{RD}}$ is $\vec{\mathsf{RD}}'$ and $\vec{\mathsf{RD}}' \sqsubseteq \mathsf{F}^n(\vec{\emptyset})$

  $\mathsf{RD}_j := \mathsf{F}_j(\mathsf{RD}_1, \cdots, \mathsf{RD}_{12})$

  then $\vec{\mathsf{RD}} \sqsubseteq \mathsf{F}(\vec{\mathsf{RD}}') \sqsubseteq \mathsf{F}^{n+1}(\vec{\emptyset}) = \mathsf{F}^n(\vec{\emptyset})$ when $\mathsf{lfp}(\mathsf{F}) = \mathsf{F}^n(\vec{\emptyset})$

If the algorithm terminates then it computes the least fixed point of F.

The algorithm terminates because $\mathsf{RD}_j \subset \mathsf{F}_j(\mathsf{RD}_1, \cdots, \mathsf{RD}_{12})$ is only possible finitely many times since $\mathcal{P}(\mathbf{Var} \times \mathbf{Lab})^{12}$ satisfies the Ascending Chain Condition.

Contraint Based Analysis

- **Technique:** Constraint Based Analysis

- **Example:** Control Flow Analysis
  - idea
  - constructing a constraint system
  - solving the constraints
  - theoretical underpinnings

Example: Control Flow Analysis

```
let f = fn x => x 7
    g = fn y => y
    h = fn z => 3
in f g + f (g h)
```

$\downarrow$ `g 7`

$\downarrow$ `f h` $\downarrow$ `h 7`

Aim: For each function application, which function abstractions may be applied?

function applications	function abstractions that may be applied
`x 7`	`g`, `h`
`f g`	`f`
`g h`	`g`
`f (g h)`	`f`

Solutions

```
let f = fn x => x 7
    g = fn y => y
    h = fn z => 3
in f g + f (g h)
```

The best solution:

function applications	function abstractions that may be applied
x 7	g, h
f g	f
g h	g
f (g h)	f

A safe solution – but not the best:

function applications	function abstractions that may be applied
x 7	g, h, $\boxed{\texttt{f}}$
f g	f
g h	g
f (g h)	f

An unsafe solution:

function applications	function abstractions that may be applied
x 7	g, $\boxed{\texttt{h}}$
f g	f
g h	g
f (g h)	f

An application of control flow analysis

```
let f = fn x => x 7
    g = fn y => y
    h = fn z => 3
in f g + f (g h)
```

Aim: For each function application, which function abstractions may be applied?

Partial evaluation of function call:

```
let f = fn x => case x of
                          g:   7
                          h:   3
    g = fn y => y
    h = fn z => 3
in f g + f (g h)
```

function applications	function abstractions that may be applied
x 7	g, h
f g	f
g h	g
f (g h)	f

The underlying analysis problem

```
let f = fn x => x 7
    g = fn y => y
    h = fn z => 3
in f g + f (g h)
```

Aim: for each function application, which function abstractions may be applied?

The analysis will compute:

- for each subexpression, which function abstractions may it denote?
  — e.g. `(g h)` may evaluate to `h`
  introduce abstract cache C
- for each variable, which function abstractions may it denote?
  — e.g. `x` may be `g` or `h`
  introduce abstract environment R

The best solution to the analysis problem

Add labels to subexpressions:

$\texttt{let f = fn x => (x}^{1}\ \texttt{7}^{2}\texttt{)}^{3}$

$\quad\texttt{g = fn y => y}^{4}$

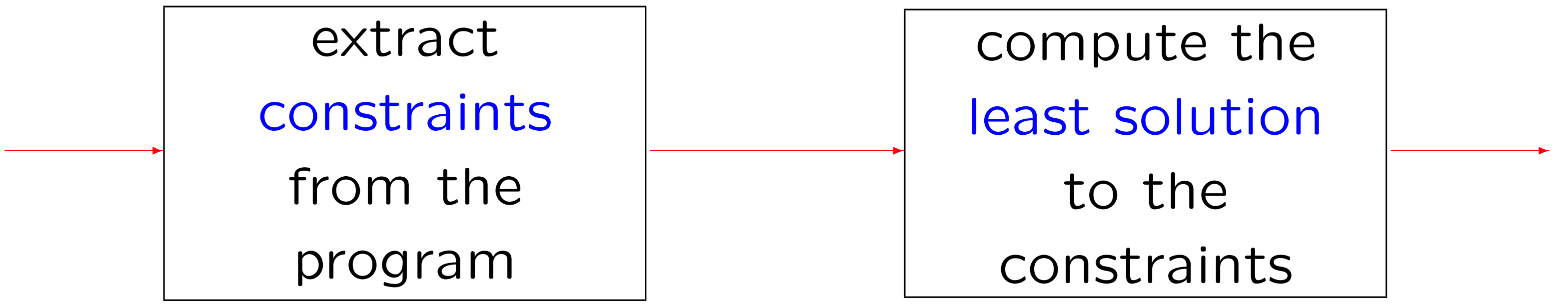
$\quad\texttt{h = fn z => 3}^{5}$

$\texttt{in (f}^{6}\ \texttt{g}^{7}\texttt{)}^{8}\ \texttt{+ (f}^{9}\ \texttt{(g}^{10}\ \texttt{h}^{11}\texttt{)}^{12}\texttt{)}^{13}$

R	variable may be bound to
x	$\{\texttt{fn y => y}^{4}, \texttt{fn z => 3}^{5}\}$
y	$\{\texttt{fn z => 3}^{5}\}$
z	$\emptyset$
f	$\{\texttt{fn x => (x}^{1}\ \texttt{7}^{2}\texttt{)}^{3}\}$
g	$\{\texttt{fn y => y}^{4}\}$
h	$\{\texttt{fn z => 3}^{5}\}$

C	subexpression may evaluate to
1	$\{\texttt{fn y => y}^{4}, \texttt{fn z => 3}^{5}\}$
2	$\emptyset$
3	$\emptyset$
4	$\{\texttt{fn z => 3}^{5}\}$
5	$\emptyset$
6	$\{\texttt{fn x => (x}^{1}\ \texttt{7}^{2}\texttt{)}^{3}\}$
7	$\{\texttt{fn y => y}^{4}\}$
8	$\emptyset$
9	$\{\texttt{fn x => (x}^{1}\ \texttt{7}^{2}\texttt{)}^{3}\}$
10	$\{\texttt{fn y => y}^{4}\}$
11	$\{\texttt{fn z => 3}^{5}\}$
12	$\{\texttt{fn z => 3}^{5}\}$
13	$\emptyset$

How to automate the analysis



Analysis information:

- R(x): information available for the variable x
- C(ℓ): information available at the subexpression labelled ℓ

Three kinds of constraints

- `let`-bound variables evaluate to their abstraction

- variables evaluate to their (abstract) values

- if a function abstraction is applied to an argument then
  - the argument is a possible value of the formal parameter
  - the value of the body of the abstraction is a possible value of the application

`let`-bound variables

`let`-bound variables evaluate to their abstractions

$$\texttt{let } f \texttt{ = fn } x \texttt{ => } e \quad \text{gives rise to the constraint } \{\texttt{fn } x \texttt{ => } e\} \subseteq \mathsf{R}(f)$$

$\texttt{let f = fn x => (x}^1\ \texttt{7}^2\texttt{)}^3 \longleftarrow \{\texttt{fn x => (x}^1\ \texttt{7}^2\texttt{)}^3\} \subseteq \mathsf{R}(\texttt{f})$

$\texttt{g = fn y => y}^4 \longleftarrow \{\texttt{fn y => y}^4\} \subseteq \mathsf{R}(\texttt{g})$

$\texttt{in (f}^5\ \texttt{g}^6\texttt{)}^7$

Variables

Variables evaluate to their abstract values

x^ℓ gives rise to the constraint $\mathsf{R}(x) \subseteq \mathsf{C}(\ell)$

```
let f = fn x => (x^1 7^2)^3
    g = fn y => y^4
in (f^5 g^6)^7
```

$\longleftarrow \quad \mathsf{R}(\mathtt{x}) \subseteq \mathsf{C}(1)$

$\longleftarrow \quad \mathsf{R}(\mathtt{y}) \subseteq \mathsf{C}(4)$

$\longleftarrow \quad \begin{cases} \mathsf{R}(\mathtt{f}) \subseteq \mathsf{C}(5) \\ \mathsf{R}(\mathtt{g}) \subseteq \mathsf{C}(6) \end{cases}$

Function application (1)

if a function abstraction is applied to an argument then

- the argument is a possible value of the formal parameter
- the value of the body of the abstraction is a possible value of the application

---

```
let f = fn x => (x^1 7^2)^3
    g = fn y => y^4
in (f^5 g^6)^7
```

if (fn y => y^4) $\in$ C(1)
then C(2) $\subseteq$ R(y) and C(4) $\subseteq$ C(3)

if (fn x => $(x^1\ 7^2)^3$) $\in$ C(1)
then C(2) $\subseteq$ R(x) and C(3) $\subseteq$ C(3)

Conditional constraints

Function application (2)

if a function abstraction is applied to an argument then

- the argument is a possible value of the formal parameter
- the value of the body of the abstraction is a possible value of the application

---

```
let f = fn x => (x^1 7^2)^3
    g = fn y => y^4
in (f^5 g^6)^7
```

if $(\texttt{fn y => y}^4) \in \mathsf{C}(5)$
then $\mathsf{C}(6) \subseteq \mathsf{R}(\texttt{y})$ and $\mathsf{C}(4) \subseteq \mathsf{C}(7)$

if $(\texttt{fn x => (x}^1\ \texttt{7}^2\texttt{)}^3) \in \mathsf{C}(5)$
then $\mathsf{C}(6) \subseteq \mathsf{R}(\texttt{x})$ and $\mathsf{C}(3) \subseteq \mathsf{C}(7)$

Summary of constraint system

$\{\texttt{fn x => (x}^1\ \texttt{7}^2\texttt{)}^3\} \subseteq \mathsf{R}(\texttt{f})$

$\{\texttt{fn y => y}^4\} \subseteq \mathsf{R}(\texttt{g})$

$\mathsf{R}(\texttt{x}) \subseteq \mathsf{C}(1)$

$\mathsf{R}(\texttt{y}) \subseteq \mathsf{C}(4)$

$\mathsf{R}(\texttt{f}) \subseteq \mathsf{C}(5)$

$\mathsf{R}(\texttt{g}) \subseteq \mathsf{C}(6)$

$(\texttt{fn y => y}^4) \in \mathsf{C}(1) \Rightarrow \mathsf{C}(2) \subseteq \mathsf{R}(\texttt{y})$

$(\texttt{fn y => y}^4) \in \mathsf{C}(1) \Rightarrow \mathsf{C}(4) \subseteq \mathsf{C}(3)$

$(\texttt{fn x => (x}^1\ \texttt{7}^2\texttt{)}^3) \in \mathsf{C}(1) \Rightarrow \mathsf{C}(2) \subseteq \mathsf{R}(\texttt{x})$

$(\texttt{fn x => (x}^1\ \texttt{7}^2\texttt{)}^3) \in \mathsf{C}(1) \Rightarrow \mathsf{C}(3) \subseteq \mathsf{C}(3)$

$(\texttt{fn y => y}^4) \in \mathsf{C}(5) \Rightarrow \mathsf{C}(6) \subseteq \mathsf{R}(\texttt{y})$

$(\texttt{fn y => y}^4) \in \mathsf{C}(5) \Rightarrow \mathsf{C}(4) \subseteq \mathsf{C}(7)$

$(\texttt{fn x => (x}^1\ \texttt{7}^2\texttt{)}^3) \in \mathsf{C}(5) \Rightarrow \mathsf{C}(6) \subseteq \mathsf{R}(\texttt{x})$

$(\texttt{fn x => (x}^1\ \texttt{7}^2\texttt{)}^3) \in \mathsf{C}(5) \Rightarrow \mathsf{C}(3) \subseteq \mathsf{C}(7)$

- 11 sets: $\mathsf{R}(\texttt{x}), \mathsf{R}(\texttt{y}), \mathsf{R}(\texttt{f}), \mathsf{R}(\texttt{g}), \mathsf{C}(1), \cdots, \mathsf{C}(7)$; all being subsets of the set **Abstr** of function abstractions
- the constraints can be reformulated as a function:
  $\mathsf{F} : \mathcal{P}(\mathbf{Abstr})^{11} \to \mathcal{P}(\mathbf{Abstr})^{11}$
- we want the least fixed point of F — this is the best solution to the constraint system

$\mathcal{P}(S)$ is the set of all subsets of the set S; e.g. $\mathcal{P}(\{0,1\}) = \{\emptyset, \{0\}, \{1\}, \{0,1\}\}$.

The constraints define a function

$\{\texttt{fn x => (x}^1\ \texttt{7}^2\texttt{)}^3\} \subseteq \mathsf{R}(\texttt{f})$
$\{\texttt{fn y => y}^4\} \subseteq \mathsf{R}(\texttt{g})$
$\mathsf{R}(\texttt{x}) \subseteq \mathsf{C}(1)$
$\mathsf{R}(\texttt{y}) \subseteq \mathsf{C}(4)$
$\mathsf{R}(\texttt{f}) \subseteq \mathsf{C}(5)$
$\mathsf{R}(\texttt{g}) \subseteq \mathsf{C}(6)$
$(\texttt{fn y => y}^4) \in \mathsf{C}(1) \Rightarrow \mathsf{C}(2) \subseteq \mathsf{R}(\texttt{y})$
$(\texttt{fn y => y}^4) \in \mathsf{C}(1) \Rightarrow \mathsf{C}(4) \subseteq \mathsf{C}(3)$
$(\texttt{fn x => (x}^1\ \texttt{7}^2\texttt{)}^3) \in \mathsf{C}(1) \Rightarrow \mathsf{C}(2) \subseteq \mathsf{R}(\texttt{x})$
$(\texttt{fn x => (x}^1\ \texttt{7}^2\texttt{)}^3) \in \mathsf{C}(1) \Rightarrow \mathsf{C}(3) \subseteq \mathsf{C}(3)$
$(\texttt{fn y => y}^4) \in \mathsf{C}(5) \Rightarrow \mathsf{C}(6) \subseteq \mathsf{R}(\texttt{y})$
$(\texttt{fn y => y}^4) \in \mathsf{C}(5) \Rightarrow \mathsf{C}(4) \subseteq \mathsf{C}(7)$
$(\texttt{fn x => (x}^1\ \texttt{7}^2\texttt{)}^3) \in \mathsf{C}(5) \Rightarrow \mathsf{C}(6) \subseteq \mathsf{R}(\texttt{x})$
$(\texttt{fn x => (x}^1\ \texttt{7}^2\texttt{)}^3) \in \mathsf{C}(5) \Rightarrow \mathsf{C}(3) \subseteq \mathsf{C}(7)$

$\mathsf{F} : \mathcal{P}(\mathbf{Abstr})^{11} \to \mathcal{P}(\mathbf{Abstr})^{11}$
is defined by

$$
\begin{array}{l}
\mathsf{F}_{\mathsf{R}(\mathsf{f})}(\cdots, R_f, \cdots) = \\
\quad R_f \cup \{\texttt{fn x => (x}^1\ \texttt{7}^2\texttt{)}^3\} \\
\vdots \\
\mathsf{F}_{\mathsf{C}(1)}(R_x, \cdots, C_1, \cdots) = C_1 \cup R_x \\
\vdots \\
\mathsf{F}_{\mathsf{R}(\mathsf{y})}(\cdots, R_y, \cdots, C_1, C_2, \cdots, C_5, C_6, \cdots) = \\
\quad R_y \cup \{a \in C_2 \mid \texttt{fn y => y}^4 \in C_1\} \\
\qquad \cup \{a \in C_6 \mid \texttt{fn y => y}^4 \in C_5\}
\end{array}
$$

How to solve the constraints

- Initialisation

  $X_1 := \emptyset; \cdots; X_{11} := \emptyset;$

writing $X_1, \cdots, X_{11}$ for $\mathsf{R}(\mathsf{x}), \cdots, \mathsf{R}(\mathsf{g}), \mathsf{C}(1), \cdots, \mathsf{C}(7)$

- Iteration

  while $X_j \neq \mathsf{F}_{\mathsf{X_j}}(X_1, \cdots, X_{11})$ for some j

  do

  $\quad X_j := \mathsf{F}_{\mathsf{X_j}}(X_1, \cdots, X_{11})$

$\boxed{\text{The algorithm terminates and computes the least fixed point of F}}$

In practice we propagate smaller contributions than $\mathsf{F}_{\mathsf{X_j}}$, e.g. a constraint at a time.

The example constraint system

R	x	y	f	g
0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
1	$\emptyset$	$\emptyset$	`fn x =>` $.^3$	$\emptyset$
2	$\emptyset$	$\emptyset$	`fn x =>` $.^3$	`fn y =>` $.^4$
3	$\emptyset$	$\emptyset$	`fn x =>` $.^3$	`fn y =>` $.^4$
4	$\emptyset$	$\emptyset$	`fn x =>` $.^3$	`fn y =>` $.^4$
5	`fn y =>` $.^4$	$\emptyset$	`fn x =>` $.^3$	`fn y =>` $.^4$
6	`fn y =>` $.^4$	$\emptyset$	`fn x =>` $.^3$	`fn y =>` $.^4$

C	1	2	3	4	5	6	7
0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
2	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
3	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	`fn x =>` $.^3$	$\emptyset$	$\emptyset$
4	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	`fn x =>` $.^3$	`fn y =>` $.^4$	$\emptyset$
5	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	`fn x =>` $.^3$	`fn y =>` $.^4$	$\emptyset$
6	`fn y =>` $.^4$	$\emptyset$	$\emptyset$	$\emptyset$	`fn x =>` $.^3$	`fn y =>` $.^4$	$\emptyset$

The constraints:

$\{\texttt{fn x =>}\ .^3\} \subseteq \mathsf{R}(\texttt{f})$ (1)

$\{\texttt{fn y =>}\ .^4\} \subseteq \mathsf{R}(\texttt{g})$ (2)

$\mathsf{R}(\texttt{x}) \subseteq \mathsf{C}(1)$ (6)

$\mathsf{R}(\texttt{y}) \subseteq \mathsf{C}(4)$

$\mathsf{R}(\texttt{f}) \subseteq \mathsf{C}(5)$ (3)

$\mathsf{R}(\texttt{g}) \subseteq \mathsf{C}(6)$ (4)

$(\texttt{fn y =>}\ .^4) \in \mathsf{C}(1) \Rightarrow \mathsf{C}(2) \subseteq \mathsf{R}(\texttt{y})$

$(\texttt{fn y =>}\ .^4) \in \mathsf{C}(1) \Rightarrow \mathsf{C}(4) \subseteq \mathsf{C}(3)$

$(\texttt{fn x =>}\ .^3) \in \mathsf{C}(1) \Rightarrow \mathsf{C}(2) \subseteq \mathsf{R}(\texttt{x})$

$(\texttt{fn x =>}\ .^3) \in \mathsf{C}(1) \Rightarrow \mathsf{C}(3) \subseteq \mathsf{C}(3)$

$(\texttt{fn y =>}\ .^4) \in \mathsf{C}(5) \Rightarrow \mathsf{C}(6) \subseteq \mathsf{R}(\texttt{y})$

$(\texttt{fn y =>}\ .^4) \in \mathsf{C}(5) \Rightarrow \mathsf{C}(4) \subseteq \mathsf{C}(7)$

$(\texttt{fn x =>}\ .^3) \in \mathsf{C}(5) \Rightarrow \mathsf{C}(6) \subseteq \mathsf{R}(\texttt{x})$ (5)

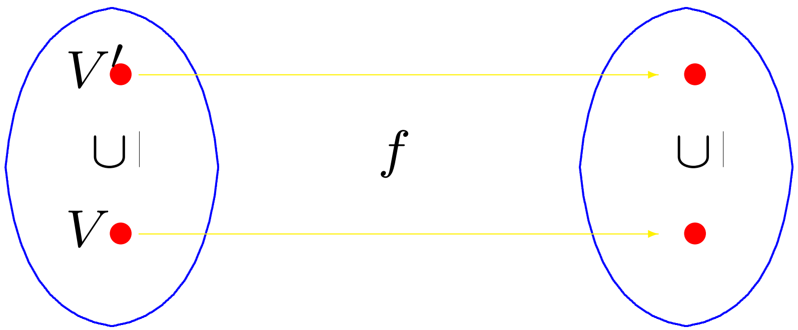
$(\texttt{fn x =>}\ .^3) \in \mathsf{C}(5) \Rightarrow \mathsf{C}(3) \subseteq \mathsf{C}(7)$

Why does it work? (1)

A function $f : \mathcal{P}(S) \to \mathcal{P}(S)$ is a monotone function if

$$V \subseteq V' \quad \Rightarrow \quad f(V) \subseteq f(V')$$

(the larger the argument — the larger the result)



Why does it work? (2)

A set L equipped with an ordering $\subseteq$ satisfies the Ascending Chain Condition if all chains

$$V_0 \subseteq V_1 \subseteq V_2 \subseteq V_3 \subseteq \cdots$$

stabilise, that is, if there exists some n such that $V_n = V_{n+1} = V_{n+2} = \cdots$

If S is a finite set then $\mathcal{P}(S)$ equipped with the subset ordering $\subseteq$ satisfies the Ascending Chain Condition — the chains cannot grow forever since each element is a subset of a finite set.

Fact

For a given program $\mathbf{Abstr}$ will be a finite set so $\mathcal{P}(\mathbf{Abstr})$ with the subset ordering satisfies the Ascending Chain Condition.

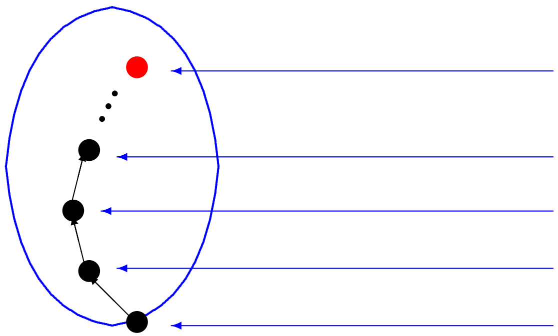
Why does it work? (3)

Let $f : \mathcal{P}(S) \rightarrow \mathcal{P}(S)$ be a monotone function. Then

$$\emptyset \subseteq f(\emptyset) \subseteq f^2(\emptyset) \subseteq f^3(\emptyset) \subseteq \cdots$$

Assume that S is a finite set; then the Ascending Chain Condition is satisfied. This means that the chain cannot grow infinitely so there exists n such that $f^n(\emptyset) = f^{n+1}(\emptyset) = \cdots$

$f^n(\emptyset)$ is the least fixed point of f



Correctness of the algorithm

- Initialisation
  $X_1 := \emptyset; \cdots; X_{11} := \emptyset;$
  Invariant: $\vec{X} \subseteq \mathsf{F}^n(\vec{\emptyset})$ since $\vec{X} = \vec{\emptyset}$ is the least element

- Iteration
  while $X_j \neq \mathsf{F}_{\mathsf{X}_j}(X_1, \cdots, X_{11})$ for some j

  do assume $\vec{X}$ is $\vec{X}'$ and $\vec{X}' \subseteq \mathsf{F}^n(\vec{\emptyset})$

  $X_j := \mathsf{F}_{\mathsf{X}_j}(X_1, \cdots, X_{11})$

  then $\vec{X} \subseteq \mathsf{F}(\vec{X}') \subseteq \mathsf{F}^{n+1}(\vec{\emptyset}) = \mathsf{F}^n(\vec{\emptyset})$ when $\mathsf{lfp}(\mathsf{F}) = \mathsf{F}^n(\vec{\emptyset})$

If the algorithm terminates then it computes the least fixed point of F

The algorithm terminates because $X_j \subset \mathsf{F}_{\mathsf{X}_j}(X_1, \cdots, X_{11})$ is only possible finitely many times since $\mathcal{P}(\mathbf{AbsExp})^{11}$ satisfies the Ascending Chain Condition

Abstract Interpretation

- **Technique:** Abstract Interpretation
- **Example:** Reaching Definitions analysis
  - idea
  - collecting semantics
  - Galois connections
  - Inducing the analysis

Abstract Interpretation

$$\boxed{\begin{array}{c}\text{program}\\ \text{analysis}\\ \text{old}\end{array}} \xrightarrow{\ ?\ } \boxed{\begin{array}{c}\text{program}\\ \text{analysis}\\ \text{new}\end{array}}$$

- We have the analysis old: it has already been proved correct but it is inefficient, or maybe even uncomputable

- We want the analysis new: it has to be correct as well as efficient!

- Can we develop new from old?

  abstract interpretation !

Example: Collecting Semantics and Reaching Definitions

$$\boxed{\begin{array}{c}\text{collecting}\\\text{semantics}\\\text{CS}\end{array}} \xrightarrow{\text{calculate}} \boxed{\begin{array}{c}\text{reaching}\\\text{definitions}\\\text{RD}\end{array}}$$

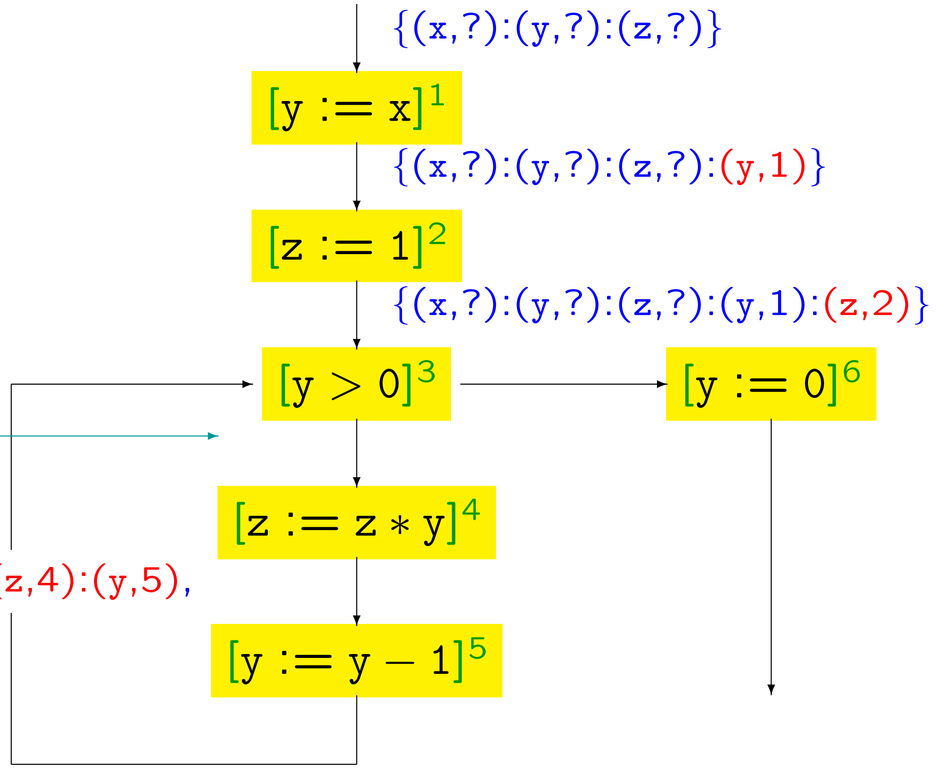
The collecting semantics CS

- collects the set of traces that can reach a given program point
- has an easy correctness proof
- is uncomputable

The reaching definitions analysis RD is as before

Example: Collecting Semantics

Collect the set of traces that reach a given program point ℓ



$\{(x,?):(y,?):(z,?):(y,1):(z,2),$

$(x,?):(y,?):(z,?):(y,1):(z,2):(z,4):(y,5),$

$(x,?):(y,?):(z,?):(y,1):(z,2):(z,4):(y,5):(z,4):(y,5),$

$\cdots\}$

How to proceed

As before:

- extract a set of equations defining the possible sets of traces
- compute the least fixed point of the set of equations

And furthermore:

- prove the correctness: the set of traces computed by the analysis is a superset of the possible traces

Two kinds of equations

$$
\begin{tikzcd}
{} \arrow[d, "\mathsf{CS}_\circ(\ell)"] \\
{[x := a]^\ell} \arrow[d, "\mathsf{CS}_\bullet(\ell)"] \\
{}
\end{tikzcd}
\qquad
\begin{tikzcd}
{[\ldots]^{\ell_1}} \arrow[dr, "\mathsf{CS}_\bullet(\ell_1)"' near start] & & {[\ldots]^{\ell_2}} \arrow[dl, "\mathsf{CS}_\bullet(\ell_2)" near start, "\mathsf{CS}_\circ(\ell)" near end] \\
& {[\ldots]^{\ell}} &
\end{tikzcd}
$$

$$\{\mathit{trace} : (x, \ell) \mid \mathit{trace} \in \mathsf{CS}_\circ(\ell)\} = \mathsf{CS}_\bullet(\ell)$$

$$\mathsf{CS}_\bullet(\ell_1) \cup \mathsf{CS}_\bullet(\ell_2) = \mathsf{CS}_\circ(\ell)$$

Flow through assignments and tests

$[\mathtt{y} := \mathtt{x}]^1$;

$[\mathtt{z} := 1]^2$;

while $[\mathtt{y} > 0]^3$ **do**

$[\mathtt{z} := \mathtt{z} * \mathtt{y}]^4$;

$[\mathtt{y} := \mathtt{y} - 1]^5$

od;

$[\mathtt{y} := 0]^6$

$$\mathsf{CS}_\bullet(1) = \{\mathit{trace} : (\mathtt{y}, 1) \mid \mathit{trace} \in \mathsf{CS}_\circ(1)\}$$

$$\mathsf{CS}_\bullet(2) = \{\mathit{trace} : (\mathtt{z}, 2) \mid \mathit{trace} \in \mathsf{CS}_\circ(2)\}$$

$$\mathsf{CS}_\bullet(3) = \mathsf{CS}_\circ(3)$$

$$\mathsf{CS}_\bullet(4) = \{\mathit{trace} : (\mathtt{z}, 4) \mid \mathit{trace} \in \mathsf{CS}_\circ(4)\}$$

$$\mathsf{CS}_\bullet(5) = \{\mathit{trace} : (\mathtt{y}, 5) \mid \mathit{trace} \in \mathsf{CS}_\circ(5)\}$$

$$\mathsf{CS}_\bullet(6) = \{\mathit{trace} : (\mathtt{y}, 6) \mid \mathit{trace} \in \mathsf{CS}_\circ(6)\}$$

6 equations in
$\mathsf{CS}_\circ(1), \cdots, \mathsf{CS}_\bullet(6)$

Flow along the control

$$\mathsf{CS}_\circ(1) = \{(x,?) : (y,?) : (z,?)\}$$

$[y := x]^1;$

$$\mathsf{CS}_\circ(2) = \mathsf{CS}_\bullet(1)$$

$[z := 1]^2;$

$$\mathsf{CS}_\circ(3) = \mathsf{CS}_\bullet(2) \cup \mathsf{CS}_\bullet(5)$$

$\texttt{while}\ [y > 0]^3\ \texttt{do}$

$$\mathsf{CS}_\circ(4) = \mathsf{CS}_\bullet(3)$$

$\quad [z := z * y]^4;$

$$\mathsf{CS}_\circ(5) = \mathsf{CS}_\bullet(4)$$

$\quad [y := y - 1]^5$

$\texttt{od};$

$$\mathsf{CS}_\circ(6) = \mathsf{CS}_\bullet(3)$$

$[y := 0]^6$

6 equations in $\mathsf{CS}_\circ(1), \cdots, \mathsf{CS}_\bullet(6)$

Summary of Collecting Semantics

$$\mathsf{CS}_\bullet(1) = \{\mathit{trace} : (\mathrm{y}, 1) \mid \mathit{trace} \in \mathsf{CS}_\circ(1)\}$$
$$\mathsf{CS}_\bullet(2) = \{\mathit{trace} : (\mathrm{z}, 2) \mid \mathit{trace} \in \mathsf{CS}_\circ(2)\}$$
$$\mathsf{CS}_\bullet(3) = \mathsf{CS}_\circ(3)$$
$$\mathsf{CS}_\bullet(4) = \{\mathit{trace} : (\mathrm{z}, 4) \mid \mathit{trace} \in \mathsf{CS}_\circ(4)\}$$
$$\mathsf{CS}_\bullet(5) = \{\mathit{trace} : (\mathrm{y}, 5) \mid \mathit{trace} \in \mathsf{CS}_\circ(5)\}$$
$$\mathsf{CS}_\bullet(6) = \{\mathit{trace} : (\mathrm{y}, 6) \mid \mathit{trace} \in \mathsf{CS}_\circ(6)\}$$

$$\mathsf{CS}_\circ(1) = \{(\mathrm{x}, ?) : (\mathrm{y}, ?) : (\mathrm{z}, ?)\}$$
$$\mathsf{CS}_\circ(2) = \mathsf{CS}_\bullet(1)$$
$$\mathsf{CS}_\circ(3) = \mathsf{CS}_\bullet(2) \cup \mathsf{CS}_\bullet(5)$$
$$\mathsf{CS}_\circ(4) = \mathsf{CS}_\bullet(3)$$
$$\mathsf{CS}_\circ(5) = \mathsf{CS}_\bullet(4)$$
$$\mathsf{CS}_\circ(6) = \mathsf{CS}_\bullet(3)$$

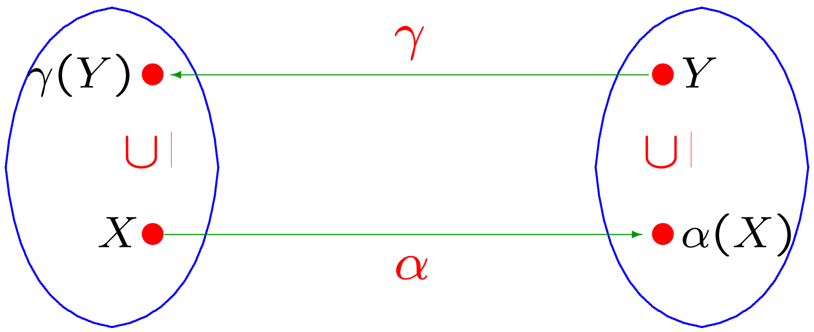
- 12 sets: $\mathsf{CS}_\circ(1), \cdots, \mathsf{CS}_\bullet(6)$
  all being subsets of $\mathbf{Trace}$
- 12 equations:
  $\mathsf{CS}_j = \mathsf{G}_j(\mathsf{CS}_\circ(1), \cdots, \mathsf{CS}_\bullet(6))$
- one function:
  $\mathsf{G} : \mathcal{P}(\mathbf{Trace})^{12} \to \mathcal{P}(\mathbf{Trace})^{12}$
- we want the least fixed point of G — but it is uncomputable!

Example: Inducing an analysis

Galois Connections

A Galois connection between two sets is a pair of (α, γ) of functions between the sets satisfying

$$X \subseteq \gamma(Y) \quad \Leftrightarrow \quad \alpha(X) \subseteq Y$$



$\mathcal{P}(\mathbf{Trace})$ collecting semantics

$\mathcal{P}(\mathbf{Var} \times \mathbf{Lab})$ reaching definitions

α: abstraction function
γ: concretisation function

Semantically Reaching Definitions

For a single trace:

trace: $(\mathtt{x},?):(\mathtt{y},?):(\mathtt{z},?):(\mathtt{y},1):(\mathtt{z},2)$

SRD(*trace*): $\{(\mathtt{x},?),(\mathtt{y},1),(\mathtt{z},2)\}$

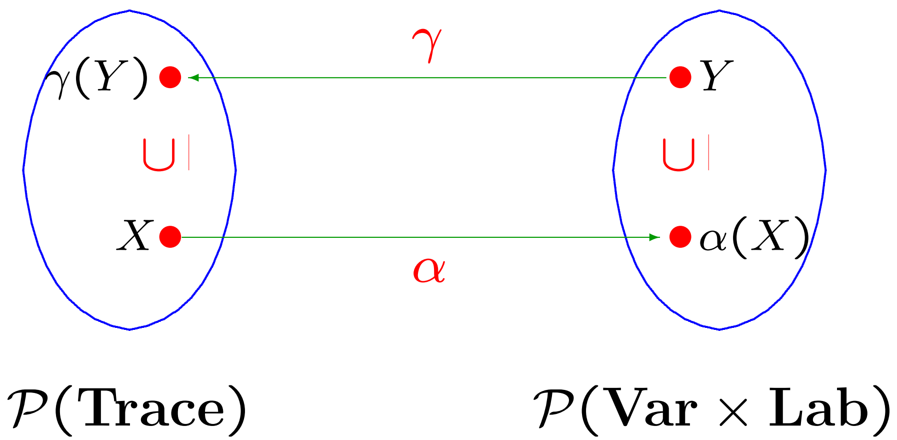
For a set of traces:

$X \in \mathcal{P}(\mathbf{Trace})$: $\{(\mathtt{x},?):(\mathtt{y},?):(\mathtt{z},?):(\mathtt{y},1):(\mathtt{z},2),$
$(\mathtt{x},?):(\mathtt{y},?):(\mathtt{z},?):(\mathtt{y},1):(\mathtt{z},2):(\mathtt{z},4):(\mathtt{y},5)\}$

SRD(X): $\{(\mathtt{x},?),(\mathtt{y},1),(\mathtt{z},2),(\mathtt{z},4),(\mathtt{y},5)\}$

Galois connection for Reaching Definitions analysis

$$\begin{array}{rcl} \alpha(X) & = & \mathsf{SRD}(X) \\ \gamma(Y) & = & \{trace \mid \mathsf{SRD}(trace) \subseteq Y\} \end{array}$$



Galois connection:

$$X \subseteq \gamma(Y) \quad \Leftrightarrow \quad \alpha(X) \subseteq Y$$

Inducing the Reaching Definitions analysis (1)

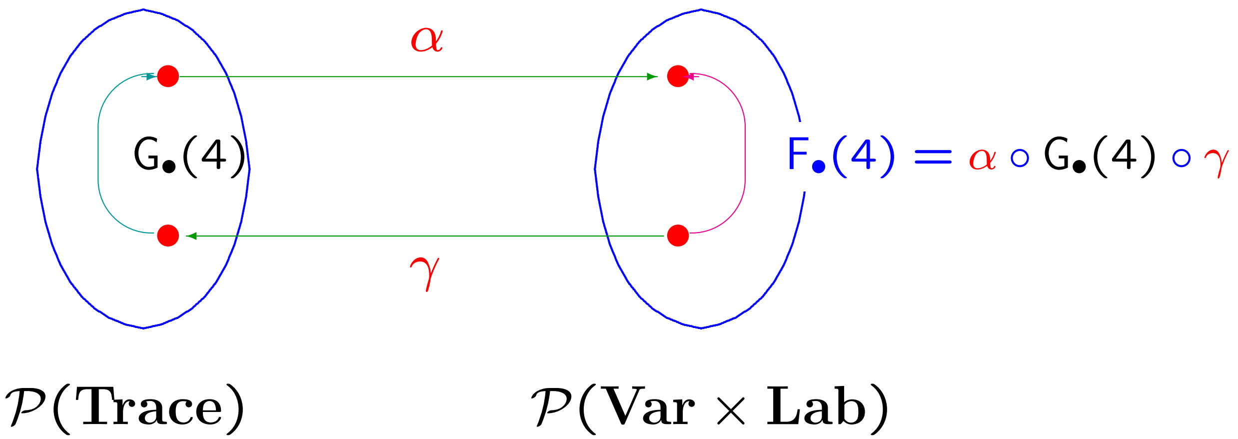
Known:

- $G_\bullet(4)$ defined on $\mathcal{P}(\mathbf{Trace})$
- the Galois connection (α, γ)

Calculate:

- $F_\bullet(4)$ defined on $\mathcal{P}(\mathbf{Var} \times \mathbf{Lab})$

  as $F_\bullet(4) = \alpha \circ G_\bullet(4) \circ \gamma$



Inducing the Reaching Definitions analysis (2)

$$
\begin{array}{rcl}
\mathsf{RD}_\bullet(4) & = & \mathsf{F}_\bullet(4)(\cdots,\mathsf{RD}_\circ(4),\cdots) \\
& = & \alpha(\mathsf{G}_\bullet(4)(\gamma(\cdots,\mathsf{RD}_\circ(4),\cdots))) \quad \text{using } \mathsf{F}_\bullet(4) = \alpha\circ\mathsf{G}_\bullet(4)\circ\gamma \\
& = & \alpha(\{tr:(\mathrm{z},4) \mid tr\in\gamma(\mathsf{RD}_\circ(4))\}) \\
& & \text{using } \mathsf{G}_\bullet(4)(\cdots,\mathsf{CS}_\circ(4),\cdots) = \{tr:(\mathrm{z},4) \mid tr\in\mathsf{CS}_\circ(4)\} \\
& = & \mathsf{SRD}(\{tr:(\mathrm{z},4) \mid tr\in\gamma(\mathsf{RD}_\circ(4))\}) \quad \text{using } \alpha = \mathsf{SRD} \\
& = & (\mathsf{SRD}(\{tr \mid tr\in\gamma(\mathsf{RD}_\circ(4))\})\backslash\{(\mathrm{z},\ell) \mid \ell\in\mathbf{Lab}\})\cup\{(\mathrm{z},4)\} \\
& = & (\alpha(\gamma(\mathsf{RD}_\circ(4)))\backslash\{(\mathrm{z},\ell) \mid \ell\in\mathbf{Lab}\})\cup\{(\mathrm{z},4)\} \quad \text{using } \alpha = \mathsf{SRD} \\
& = & (\mathsf{RD}_\circ(4)\backslash\{(\mathrm{z},\ell) \mid \ell\in\mathbf{Lab}\})\cup\{(\mathrm{z},4)\} \quad \text{using } \alpha\circ\gamma = id
\end{array}
$$

just as before!

Type and Effect Systems

- **Technique:** Annotated Type Systems

- **Example:** Reaching Definitions analysis

  - idea

  - annotated base types

  - annotated type constructors

The while language

- syntax of statements:

$$
\begin{array}{rcl}
S & ::= & [x := a]^\ell \mid S_1; S_2 \\
 & \mid & \texttt{if } [b]^\ell \texttt{ then } S_1 \texttt{ else } S_2 \texttt{ fi} \\
 & \mid & \texttt{while } [b]^\ell \texttt{ do } S \texttt{ od}
\end{array}
$$

- semantics:

  statements map states to states

- types:

  Σ is the type of states;

  all statements S have type $\Sigma \to \Sigma$ written $\vdash S : \Sigma \to \Sigma$

Annotated base types

type of initial state

type of final state(s)

$$\vdash \quad S : \quad \Sigma \;\rightarrow\; \Sigma$$

Analysis:

$$\vdash \quad S : \quad \mathsf{RD}_1 \;\rightarrow\; \mathsf{RD}_2$$

analysis information for initial state

analysis information for final state(s)

Idea: $\mathsf{RD}_1 \subseteq \mathsf{RD}_\circ(\mathit{init}(S))$
$\Downarrow$
$\forall \ell \in \mathit{final}(S) :$
$\mathsf{RD}_\bullet(\ell) \subseteq \mathsf{RD}_2$

Annotated type system (1)

$$\vdash [x := a]^{\ell} : \underbrace{\mathsf{RD}}_{\text{before}} \;\rightarrow\; \underbrace{(\mathsf{RD} \setminus \{(x,\ell') \mid \ell' \in \mathbf{Lab}\}) \cup \{(x,\ell)\}}_{\text{after}}$$

$$\frac{\vdash S_1 : \mathsf{RD}_1 \rightarrow \mathsf{RD}_2 \qquad \vdash S_2 : \mathsf{RD}_2 \rightarrow \mathsf{RD}_3}{\vdash S_1; S_2 : \underbrace{\mathsf{RD}_1}_{\text{before}} \rightarrow \underbrace{\mathsf{RD}_3}_{\text{after}}} \qquad \frac{\text{assumptions}}{\text{conclusion}}$$

Implicit: the analysis information at the exit of S_1
equals the analysis information at the entry of S_2

Annotated type system (2)

$$\frac{\vdash S_1 : \mathsf{RD}_1 \rightarrow \mathsf{RD}_2 \qquad \vdash S_2 : \mathsf{RD}_1 \rightarrow \mathsf{RD}_2}{\vdash \texttt{if } [b]^\ell \texttt{ then } S_1 \texttt{ else } S_2 \texttt{ fi} : \mathsf{RD}_1 \rightarrow \mathsf{RD}_2}$$

Implicit: the two branches have the same analysis information at their respective entry and exit points

$$\frac{\vdash S : \mathsf{RD} \rightarrow \mathsf{RD}}{\vdash \texttt{while } [b]^\ell \texttt{ do } S \texttt{ od} : \mathsf{RD} \rightarrow \mathsf{RD}}$$

Implicit: the occurrences of RD express an invariance i.e. a fixed point property!

Annotated type system (3)

The subsumption rule:

$$\frac{\vdash S : \mathsf{RD}_1' \to \mathsf{RD}_2'}{\vdash S : \mathsf{RD}_1 \to \mathsf{RD}_2} \quad \text{if } \mathsf{RD}_1 \subseteq \mathsf{RD}_1' \text{ and } \mathsf{RD}_2' \subseteq \mathsf{RD}_2$$

The rule is essential for the rules for conditional and iteration to work

- $\mathsf{RD}_1 \subseteq \mathsf{RD}_1'$: strengthen the analysis information for the initial state

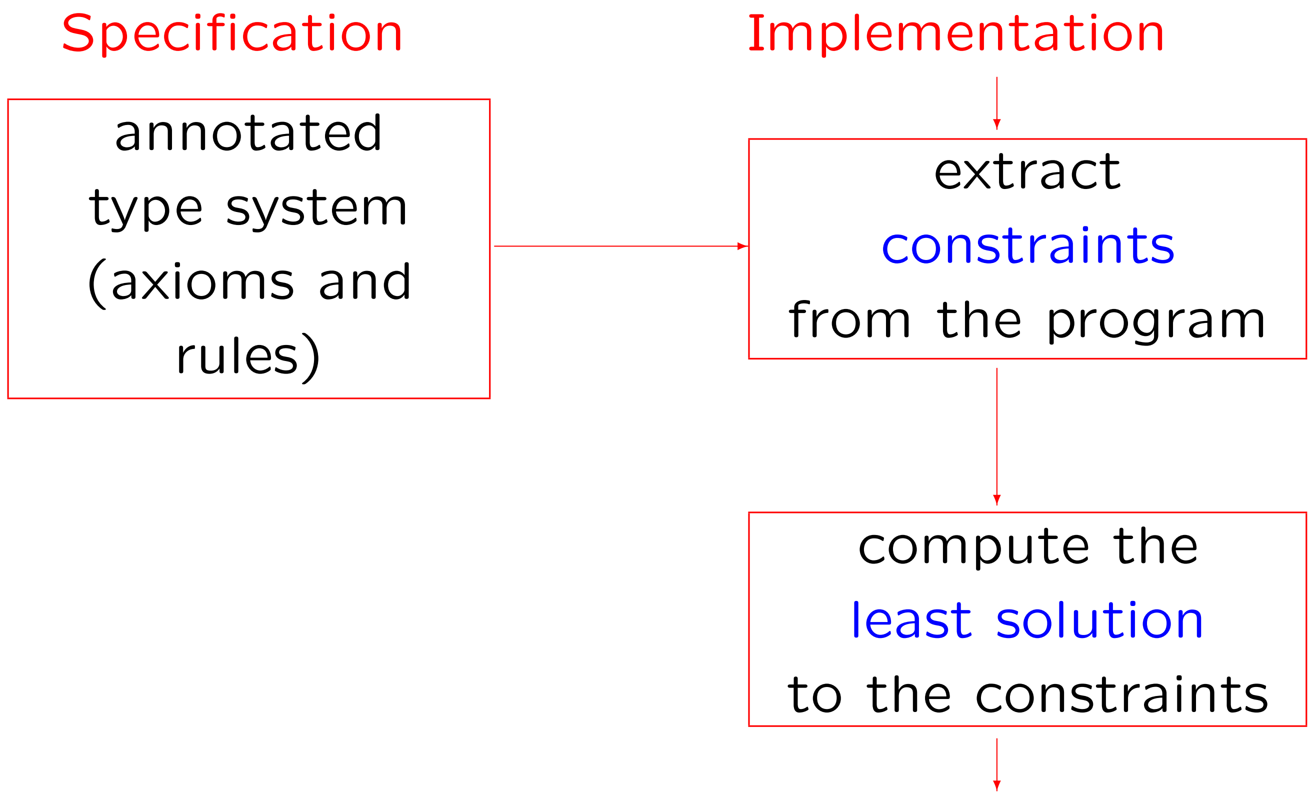
- $\mathsf{RD}_2' \subseteq \mathsf{RD}_2$: weaken the analysis information for the final states

Example inference in the annotated type system

Abbreviation: $\mathsf{RD} = \{(x,?),(y,1),(y,5),(z,2),(z,4)\}$

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdash [\texttt{z:=z*y}]^4 : \ \mathsf{RD} \to \{(x,?),(y,1),(y,5),(z,4)\} \\ \vdash [\texttt{y:=y-1}]^5 : \{(x,?),(y,1),(y,5),(z,4)\} \to \{(x,?),(y,5),(z,4)\}\end{array}}{\vdash [\texttt{z:=z*y}]^4;\ [\texttt{y:=y-1}]^5 : \ \mathsf{RD} \to \{(x,?),(y,5),(z,4)\}}}{\begin{array}{r}\vdash [\texttt{z:=z*y}]^4;\ [\texttt{y:=y-1}]^5 : \ \mathsf{RD} \to \mathsf{RD} \\ \text{using } \{(x,?),(y,5),(z,4)\} \subseteq \mathsf{RD}\end{array}}}{\vdash \texttt{while } [\texttt{y>1}]^3 \texttt{ do } [\texttt{z:=z*y}]^4;\ [\texttt{y:=y-1}]^5 \texttt{ od}: \ \mathsf{RD} \to \mathsf{RD}}}{\vdots}}{\begin{array}{c}\vdash [\texttt{y:=x}]^1;\ [\texttt{z:=1}]^2;\ \texttt{while } [\texttt{y>1}]^3 \texttt{ do } [\texttt{z:=z*y}]^4;\ [\texttt{y:=y-1}]^5 \texttt{ od};\ [\texttt{y:=0}]^6 : \\ \{(x,?),(y,?),(z,?)\} \to \{(x,?),(y,6),(z,2),(z,4)\}\end{array}}$$

How to automate the analysis



Change of abstraction level: annotated type constructors

- **Until now:**

  given a statement and a specific entry information $\mathsf{RD}_\circ$ we determine the specific exit information $\mathsf{RD}_\bullet$

- **Now:**

  given a statement we determine how entry information is transformed into exit information

Annotated type constructors

type of initial state

type of final state(s)

$$\vdash \quad S : \quad \Sigma \;\longrightarrow\; \Sigma$$

Analysis:

$$\vdash \quad S : \quad \Sigma \;\xrightarrow[\mathsf{RD}]{\mathsf{X}}\; \Sigma$$

reaching definitions that might be produced by S

set of variables definitely assigned by S

Annotated type constructors (1)

$$\vdash [x := a]^\ell : \Sigma \xrightarrow[\{(x,\ell)\}]{\{x\}} \Sigma$$

$\{x\}$: variables definitely assigned

$\{(x,\ell)\}$: potential reaching definitions

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

$X_1 \cup X_2$: variables definitely assigned

$(\mathsf{RD}_1 \setminus X_2) \cup \mathsf{RD}_2$:
potential reaching definitions

Annotated type constructors (2)

$$\frac{\vdash S_1 : \Sigma \xrightarrow[\mathsf{RD}_1]{X_1} \Sigma \qquad \vdash S_2 : \Sigma \xrightarrow[\mathsf{RD}_2]{X_2} \Sigma}{\vdash \texttt{if } [b]^\ell \texttt{ then } S_1 \texttt{ else } S_2 \texttt{ fi} : \Sigma \xrightarrow[\mathsf{RD}_1 \cup \mathsf{RD}_2]{X_1 \cap X_2} \Sigma}$$

$X_1 \cap X_2$:
variables definitely assigned

$\mathsf{RD}_1 \cup \mathsf{RD}_2$:
potential reaching definitions

$$\frac{\vdash S : \Sigma \xrightarrow[\mathsf{RD}]{X} \Sigma}{\vdash \texttt{while } [b]^\ell \texttt{ do } S \texttt{ od} : \Sigma \xrightarrow[\mathsf{RD}]{\emptyset} \Sigma}$$

$\emptyset$: variables definitely assigned

RD : potential reaching definitions

Annotated type constructors (3)

Subsumption rule:

$$\frac{\vdash S : \Sigma \xrightarrow[\mathsf{RD}]{X} \Sigma}{\vdash S : \Sigma \xrightarrow[\mathsf{RD}']{X'} \Sigma} \quad \begin{array}{ll} \text{if } X' \subseteq X & \text{(variables definite assigned)} \\ \text{and } \mathsf{RD} \subseteq \mathsf{RD}' & \text{(potential reaching definitions)} \end{array}$$

the rule can be omitted!

Example inference in the annotated type system

$$\vdash\ [\texttt{z:=z*y}]^4\colon\ \Sigma \xrightarrow[\{(\mathrm{z},4)\}]{\{\mathrm{z}\}} \Sigma$$

$$\vdash\ [\texttt{y:=y-1}]^5\colon\ \Sigma \xrightarrow[\{(\mathrm{y},5)\}]{\{\mathrm{y}\}} \Sigma$$

$$\vdash\ [\texttt{z:=z*y}]^4;\ [\texttt{y:=y-1}]^5\colon\ \Sigma \xrightarrow[\{(\mathrm{y},5),(\mathrm{z},4)\}]{\{\mathrm{y},\mathrm{z}\}} \Sigma$$

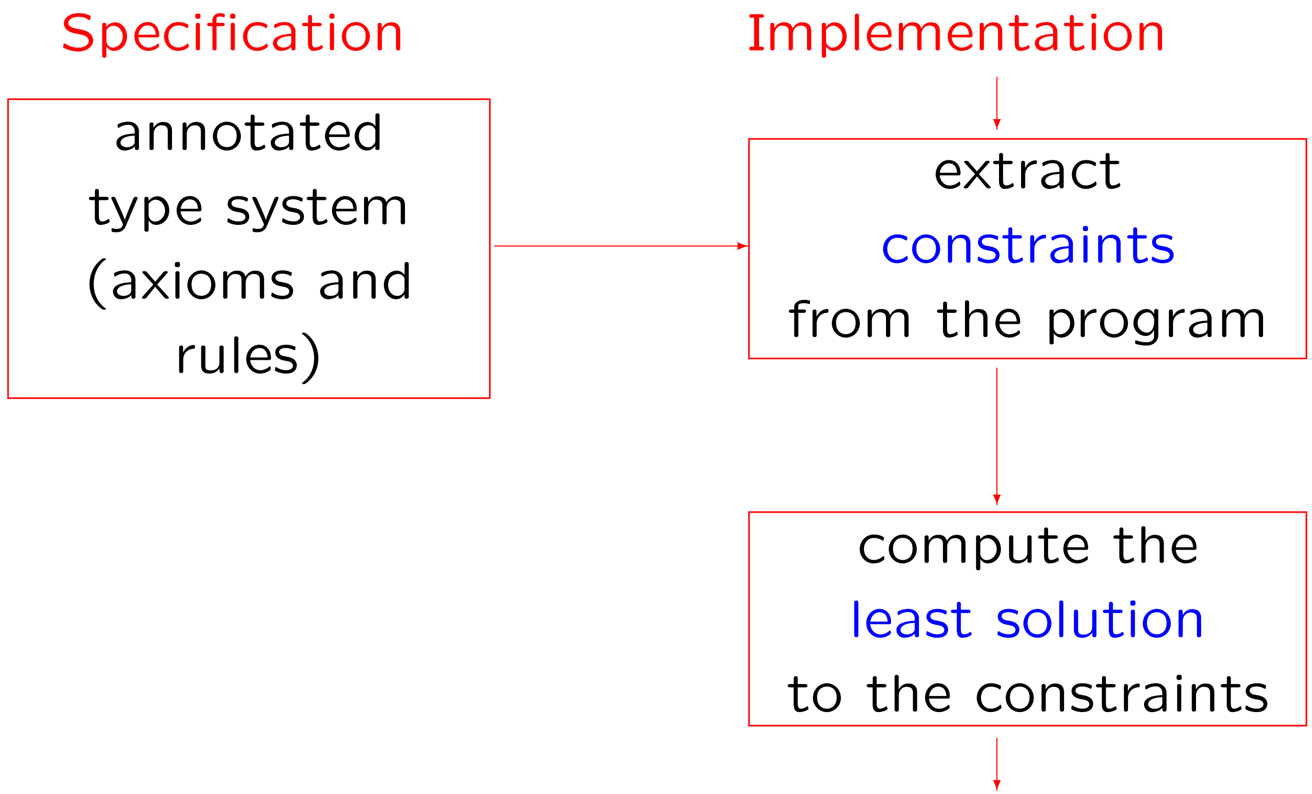
$$\vdash\ \texttt{while}\ [\texttt{y>1}]^3\ \texttt{do}\ [\texttt{z:=z*y}]^4;\ [\texttt{y:=y-1}]^5\ \texttt{od}\colon\ \Sigma \xrightarrow[\{(\mathrm{y},5),(\mathrm{z},4)\}]{\emptyset} \Sigma$$

$$\vdots$$

$$\vdash [\texttt{y:=x}]^1;\ [\texttt{z:=1}]^2;\ \texttt{while}\ [\texttt{y>1}]^3\ \texttt{do}\ [\texttt{z:=z*y}]^4;\ [\texttt{y:=y-1}]^5\ \texttt{od};\ [\texttt{y:=0}]^6\colon$$

$$\Sigma \xrightarrow[\{(\mathrm{y},6),(\mathrm{z},2),(\mathrm{z},4)\}]{\{\mathrm{y},\mathrm{z}\}} \Sigma$$

How to automate the analysis



Type and Effect Systems

- **Technique:** Effect systems
- **Example:** Call Tracking analysis
  - idea
  - simple type system
  - effect system

The fun language

- syntax of expressions

$$e ::= x \mid \mathtt{fn}_\pi\ x\ \mathtt{=>}\ e \mid e_1\ e_2 \mid \cdots$$

$\boxed{\pi \text{ names the function abstraction}}$

- types

$$\tau ::= \mathtt{int} \mid \mathtt{bool} \mid \tau_1 \rightarrow \tau_2$$

f has type $\tau_1 \rightarrow \tau_2$ means that
  - f expects a parameter of type τ_1
  - f returns a value of type τ_2

Call Tracking analysis

```
let f = fn_F x => x 7
    g = fn_G y => y
    h = fn_H z => z
in f g + f (h g)
```

$$\begin{array}{ccc} \downarrow & & \downarrow \\ \texttt{g 7} & & \texttt{f g} \\ & & \downarrow \\ & & \texttt{g 7} \end{array}$$

Aim: For each function application, which function abstractions might be applied during its execution?

function applications	function abstractions that might be applied during its execution
x 7	G, H
f g	F, G
h g	H, G
f (h g)	F, H, G

Simple types

$$
\begin{array}{llll}
\texttt{let f = fn}_{F}\texttt{ x => x 7} & \longleftarrow & \texttt{f}: (\texttt{int} \to \texttt{int}) \to \texttt{int} \\
\quad\texttt{g = fn}_{G}\texttt{ y => y} & \longleftarrow & \texttt{g}: \texttt{int} \to \texttt{int} \\
\quad\texttt{h = fn}_{H}\texttt{ z => z} & \longleftarrow & \texttt{h}: (\texttt{int} \to \texttt{int}) \to (\texttt{int} \to \texttt{int}) \\
\texttt{in f g + f (h g)} & \longleftarrow & \texttt{int}
\end{array}
$$

Simple type system

- type environment: Γ gives types to the variables (like R)

- an expression e has type τ relative to type environment Γ (like C)

  $\Gamma \vdash e : \tau$

A simple type system

$$\Gamma \vdash x : \tau_x \qquad \text{if } \Gamma(x) = \tau_x$$

$$\frac{\Gamma[x \mapsto \tau_x] \vdash e : \tau}{\Gamma \vdash \mathtt{fn}_\pi\ x\ \mathtt{=>}\ e : \tau_x \to \tau}$$

guess: τ_x is the type of the argument x

$$\frac{\Gamma \vdash e_1 : \tau_2 \to \tau, \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau}$$

the type of the formal parameter equals that of the actual parameter

Effect systems

- Call Tracking analysis:
  For each function application, which function abstractions might be applied during its execution?

- Idea:
  Annotate the function arrows with sets of names of function abstractions that might be applied when calling the function.

$$\tau_1 \xrightarrow{\varphi} \tau_2$$

the type of functions from τ_1 to τ_2 that might call functions with names in φ.

Example: Types and Effects

$$
\begin{array}{lcl}
\texttt{let f = fn}_{\mathsf{F}}\ \texttt{x => x 7} & \longleftarrow & \texttt{f}:\ (\texttt{int} \xrightarrow{\{\mathsf{G}\}} \texttt{int}) \xrightarrow{\{\mathsf{F},\mathsf{G}\}} \texttt{int} \\
\quad\ \texttt{g = fn}_{\mathsf{G}}\ \texttt{y => y} & \longleftarrow & \texttt{g}:\ \texttt{int} \xrightarrow{\{\mathsf{G}\}} \texttt{int} \\
\quad\ \texttt{h = fn}_{\mathsf{H}}\ \texttt{z => z} & \longleftarrow & \texttt{h}:\ (\texttt{int} \xrightarrow{\{\mathsf{G}\}} \texttt{int}) \xrightarrow{\{\mathsf{H}\}} (\texttt{int} \xrightarrow{\{\mathsf{G}\}} \texttt{int}) \\
\texttt{in f g + f (h g)} & \longleftarrow & \texttt{int} \quad \& \quad \underbrace{\{\mathsf{F},\mathsf{G},\mathsf{H}\}}_{\substack{\text{the effect} \\ \text{of executing} \\ \texttt{f g + f (g h)}}}
\end{array}
$$

The effect system

$$\Gamma \vdash x : \tau_x \ \&\ \emptyset \qquad \text{if } \Gamma(x) = \tau_x$$

variables have no effect

$$\frac{\Gamma[x \mapsto \tau_x] \vdash e : \tau \ \&\ \varphi}{\Gamma \vdash \mathtt{fn}_\pi\ x\ \mathtt{=>}\ e : \tau_x \xrightarrow{\varphi \cup \{\pi\}} \tau \ \&\ \emptyset}$$

the latent effect consists of
–that of the function body
–the function itself

the function abstraction itself has no effect

$$\frac{\Gamma \vdash e_1 : \tau_2 \xrightarrow{\varphi} \tau \ \&\ \varphi_1 \qquad \Gamma \vdash e_2 : \tau_2 \ \&\ \varphi_2}{\Gamma \vdash e_1\ e_2 : \tau \ \&\ \varphi_1 \cup \varphi_2 \cup \varphi}$$

the overall effect comes from
–evaluating the function
–evaluating the argument
–evaluating the function application: the latent effect!

The effect system

The subsumption rule:

$$\frac{\Gamma \vdash e : \tau \;\&\; \varphi}{\Gamma \vdash e : \tau \;\&\; \varphi \;\cup\; \varphi'}$$

the names of functions that may be applied

Example (1)

```
let f = fnF x => x 7
    g = fnG y => y
    h = fnH z => z
in f g + f (h g)
```

$$\frac{\Gamma[x \mapsto \tau_x] \vdash e : \tau \;\&\; \varphi}{\Gamma \vdash \mathtt{fn}_\pi\ x\ \mathtt{=>}\ e : \tau_x \xrightarrow{\varphi\ \cup\ \{\pi\}} \tau \;\&\; \emptyset}$$

$$\frac{\Gamma \vdash e_1 : \tau_2 \xrightarrow{\varphi} \tau \;\&\; \varphi_1 \qquad \Gamma \vdash e_2 : \tau_2 \;\&\; \varphi_2}{\Gamma \vdash e_1\ e_2 : \tau \;\&\; \varphi_1 \cup \varphi_2 \cup \varphi}$$

$$\frac{\dfrac{[\mathtt{x} \mapsto \mathtt{int} \xrightarrow{\{\mathsf{G}\}} \mathtt{int}] \vdash \mathtt{x} : \mathtt{int} \xrightarrow{\{\mathsf{G}\}} \mathtt{int} \;\&\; \emptyset \qquad [\mathtt{x} \mapsto \mathtt{int} \xrightarrow{\{\mathsf{G}\}} \mathtt{int}] \vdash \mathtt{7} : \mathtt{int} \;\&\; \emptyset}{[\mathtt{x} \mapsto \mathtt{int} \xrightarrow{\{\mathsf{G}\}} \mathtt{int}] \vdash \mathtt{x}\ \mathtt{7} : \mathtt{int} \;\&\; \{\mathsf{G}\}}}{[\ ] \vdash \mathtt{fn}_\mathsf{F}\ \mathtt{x}\ \mathtt{=>}\ \mathtt{x}\ \mathtt{7} : (\mathtt{int} \xrightarrow{\{\mathsf{G}\}} \mathtt{int}) \xrightarrow{\{\mathsf{F},\mathsf{G}\}} \mathtt{int} \;\&\; \emptyset}$$

Example (2)

```
let f = fnF x => x 7
    g = fnG y => y
    h = fnH z => z
in f g + f (h g)
```

$$\frac{\Gamma[x \mapsto \tau_x] \vdash e : \tau \;\&\; \varphi}{\Gamma \vdash \mathtt{fn}_\pi\ x \Rightarrow e : \tau_x \xrightarrow{\varphi \,\cup\, \{\pi\}} \tau \;\&\; \emptyset}$$

$$\frac{\Gamma \vdash e_1 : \tau_2 \xrightarrow{\varphi} \tau \;\&\; \varphi_1 \qquad \Gamma \vdash e_2 : \tau_2 \;\&\; \varphi_2}{\Gamma \vdash e_1\ e_2 : \tau \;\&\; \varphi_1 \cup \varphi_2 \cup \varphi}$$

$$\dfrac{\Gamma \vdash \mathtt{f} : (\mathtt{int} \xrightarrow{\{\mathsf{G}\}} \mathtt{int}) \xrightarrow{\{\mathsf{F},\mathsf{G}\}} \mathtt{int} \;\&\; \emptyset \qquad \dfrac{\begin{array}{c}\Gamma \vdash \mathtt{h} : (\mathtt{int} \xrightarrow{\{\mathsf{G}\}} \mathtt{int}) \xrightarrow{\{\mathsf{H}\}} (\mathtt{int} \xrightarrow{\{\mathsf{G}\}} \mathtt{int}) \;\&\; \emptyset \\ \Gamma \vdash \mathtt{g} : \mathtt{int} \xrightarrow{\{\mathsf{G}\}} \mathtt{int} \;\&\; \emptyset\end{array}}{\Gamma \vdash \mathtt{h}\ \mathtt{g} : \mathtt{int} \xrightarrow{\{\mathsf{G}\}} \mathtt{int} \;\&\; \{\mathsf{H}\}}}{\Gamma \vdash \mathtt{f}\ (\mathtt{h}\ \mathtt{g}) : \mathtt{int} \;\&\; \{\mathsf{F},\mathsf{G},\mathsf{H}\}}$$

How to automate the analysis