Semantics of an Intermediate Language for Program Transformation

Sigurd Schneider

Master Thesis Proposal Talk
Advisors: Prof. Dr. Sebastian Hack, Prof. Dr. Gert Smolka

Saarland University
Graduate School of Computer Science

November 11, 2011

Overview

Motivation

- Static Single Assignment Form (SSA Form)
  - standard (libfirm, llvm, hotspot, gcc)
  - eases specification and implementation of program transformations
- Does SSA form ease the correctness proofs of program transformations, too?

Static Single Assignment Form

- Alpern, Cytron, Ferrante, Rosen, Wegman, Zadeck [6, 1, 2]
- developed in the context of data flow analysis
- SSA condition
  - every name has exactly one assignment in the program
  - join points of control flow are mediated by phi nodes (e.g. loops)

Example

Program

```
1 // n input
2 entry:   i = 1;
3 loop :   if n = 0 then
4            return i
5          else
6            i = n * i;
7            n = n - 1;
8            goto loop
```

Example

Program

```
1 // n input
2 entry:   i = 1;
3 loop :   if n = 0 then
4            return i
5          else
6            i = n * i;
7            n = n - 1;
8            goto loop
```

Program with variables renamed (broken)

```
1 // n0 input
2 entry: i0 = 1;
3 loop : if n0 = 0 then
4          return i0
5        else
6          i1 = n0 * i0;
7          n1 = n0 - 1;
8          body:goto loop
```

Example

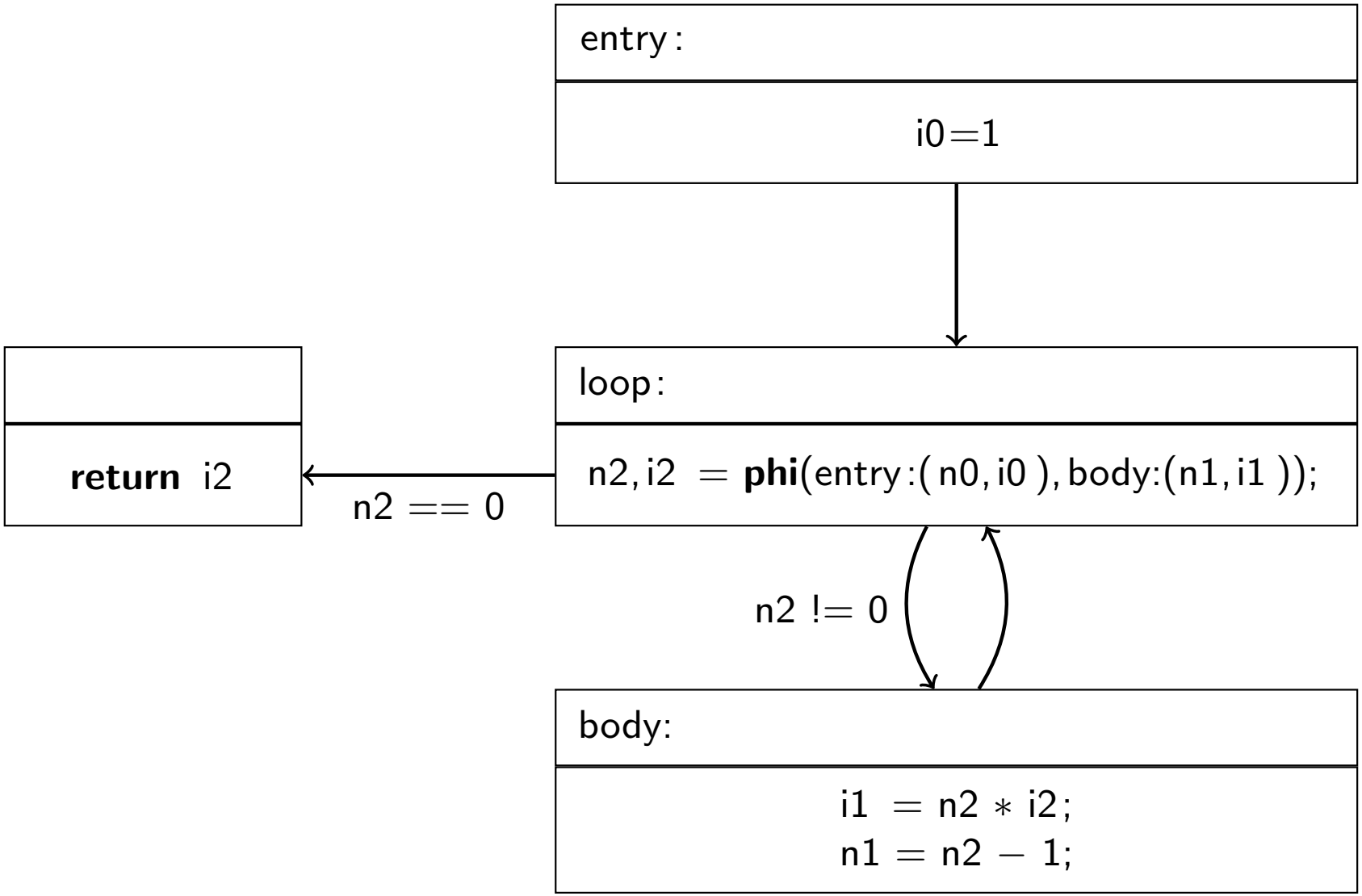
Program

```
1 // n input
2 entry:   i = 1;
3 loop :   if n = 0 then
4            return i
5          else
6            i = n * i;
7            n = n - 1;
8            goto loop
```

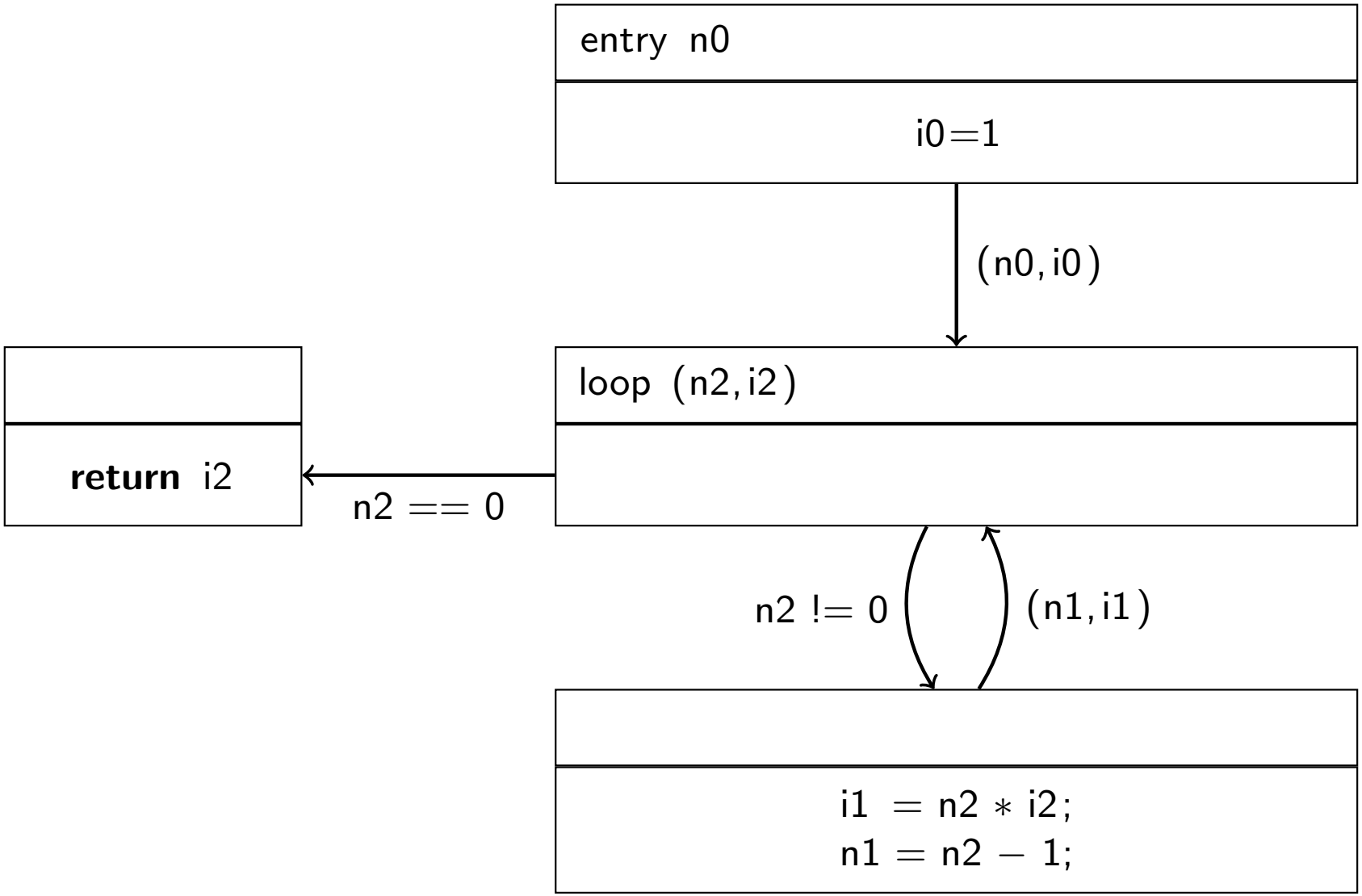
Program in SSA form

```
1  // n0 input
2  entry: i0 = 1;
3  loop : (n2,i2) =
4           phi(entry:(n0,i0),
5                body:(n1,i1));
6         if n2 = 0 then
7           return i2
8         else
9           i1 = n2 * i2;
10          n1 = n2 - 1;
11          body:goto loop
```

Control Flow Graph



Control Flow Graph revisited



Syntax

We assume expressions e of integer type with tuples.

Program equations

$$\begin{aligned} f_0\ x_0 &= s_0 \\ f_1\ x_1 &= s_1 \\ &\vdots \end{aligned}$$

shortly written as

$$\overline{f\ x = s}$$

Statement language

$$\begin{aligned} s, t ::= {} & x = e;\ s && \text{assignment} \\ \mid {} & \texttt{if}\ e\ s\ t && \text{conditional} \\ \mid {} & \texttt{goto}\ l\ x && \text{goto} \\ \mid {} & \texttt{return}\ x && \text{return} \end{aligned}$$

Example

Program in SSA form

```
1  // n0 input
2  entry: i0 = 1;
3  loop : (n2,i2) =
4           phi(entry:(n0,i0),
5                body:(n1,i1));
6          if n2 = 0 then
7            return i2
8          else
9            i1 = n2 * i2;
10           n1 = n2 - 1;
11           body:goto loop
```

Program equations in IL

```
1  // n0 input
2  entry n0 =
3    i0 = 1;
4    goto loop (n0,i0)
5
6  loop (n2,i2) =
7    if n2 = 0 then
8      return i2
9    else
10     i1 = n2 * i2;
11     n1 = n2 - 1;
12     goto loop (n1, i1)
```

Semantics for the IL 1

The semantics is defined on state tuples

$$\Pi \vdash \sigma, s \qquad \text{where} \qquad \begin{array}{ll} \Pi & \text{program equations } \overline{f\ x = s} \\ \sigma & \text{environment} \\ s & \text{current statement} \end{array}$$

Semantics for the IL 2

$$\textsc{Assign}\ \frac{\Pi \vdash \sigma[x := [\![e]\!]\,\sigma], s \triangleright v}{\Pi \vdash \sigma, x = e; s \triangleright v}$$

Semantics for the IL 2

$$\textsc{Assign}\ \frac{\Pi \vdash \sigma[x := [\![e]\!]\,\sigma], s \triangleright v}{\Pi \vdash \sigma, x = e; s \triangleright v} \qquad \textsc{Cond}\ \frac{[\![e]\!]\,\sigma = b \quad \Pi \vdash \sigma, s_b \triangleright v}{\Pi \vdash \sigma, \texttt{if}\ e\ s_0\ s_1 \triangleright v}$$

Semantics for the IL 2

$$\textsc{Assign}\ \frac{\Pi \vdash \sigma[x := [\![e]\!]\,\sigma], s \triangleright v}{\Pi \vdash \sigma, x = e; s \triangleright v}$$

$$\textsc{Cond}\ \frac{[\![e]\!]\,\sigma = b \quad \Pi \vdash \sigma, s_b \triangleright v}{\Pi \vdash \sigma, \texttt{if}\ e\ s_0\ s_1 \triangleright v}$$

$$\textsc{Return}\ \frac{\sigma\, x = v}{\Pi \vdash \sigma, \texttt{return}\ x \triangleright v}$$

Semantics for the IL 2

$$\textsc{Assign}\ \frac{\Pi \vdash \sigma[x := [\![e]\!]\,\sigma], s \triangleright v}{\Pi \vdash \sigma, x = e; s \triangleright v} \qquad \textsc{Cond}\ \frac{[\![e]\!]\,\sigma = b \quad \Pi \vdash \sigma, s_b \triangleright v}{\Pi \vdash \sigma, \texttt{if}\ e\ s_0\ s_1 \triangleright v}$$

$$\textsc{Return}\ \frac{\sigma\, x = v}{\Pi \vdash \sigma, \texttt{return}\, x \triangleright v} \qquad \textsc{Goto}\ \frac{\overline{f\ x = s} \vdash \emptyset[x_i := \sigma\, y], s_i \triangleright v}{\overline{f\ x = s} \vdash \sigma, \texttt{goto}\, f_i\ y \triangleright v}$$

Contextual Equivalence (Morris [5])

"Two programs are contextually equivalent if, whenever they are each inserted into a hole in a larger program of integer type, the resulting programs either both converge or both diverge." [3]

$$C ::= [] \mid x = e;\ C \mid \mathtt{if}_1\ e\ C\ s \mid \mathtt{if}_2\ e\ s\ C$$

$$\forall C\ v,\quad \Pi \vdash\ \emptyset, C[s] \triangleright v\ \leftrightarrow\ \emptyset, C[s'] \triangleright v$$

Statement Equivalence

$$\Pi \vdash s \approx s' \iff \forall \sigma\, v,\quad \Pi \vdash \sigma, s \triangleright v \;\leftrightarrow\; \sigma, s' \triangleright v$$

Congruence

$$\frac{\Pi \vdash s \approx s'}{\Pi \vdash C[s] \approx C[s']}$$

Completeness

$$\frac{\forall C\, v,\quad \Pi \vdash\ \emptyset, C[s] \triangleright v \;\leftrightarrow\; \emptyset, C[s'] \triangleright v}{\Pi \vdash s \approx s'}$$

Program Equivalence

$$\Pi \vdash f \approx g \iff \Pi \vdash \mathtt{goto}\, f\, x \approx \mathtt{goto}\, g\, x$$

Extend

$$\frac{\Pi \vdash s \approx s' \quad g \text{ fresh}}{\Pi, g\, x = t \vdash s \approx s'}$$

Extract

$$\forall \sigma,\ \Pi|_g \vdash \sigma, \mathtt{goto}\, g\, x \triangleright v \ \leftrightarrow\ \Pi \vdash \sigma, \mathtt{goto}\, g\, x \triangleright v$$

where $\Pi|_g$ denotes the set equation names possibly reachable from g.

$f\ x = y = x; \texttt{return}\ y$ is equivalent to $f\ x = \texttt{return}\ x$

$$\textsc{Extend}\ \frac{\Pi \vdash s \approx s' \quad g\ \text{fresh}}{\Pi, g\ x = t \vdash s \approx s'}$$

$f\,x = y = x; \texttt{return}\,y$ is equivalent to $f\,x = \texttt{return}\,x$

$$\textsc{Extend}\ \frac{\Pi \vdash s \approx s' \quad g\ \text{fresh}}{\Pi, g\,x = t \vdash s \approx s'}$$

$$\Pi := \begin{array}{l} f\,x = y = x; \texttt{return}\,y \\ g\,x = \texttt{return}\,x \end{array}$$

$f\ x = y = x; \texttt{return}\ y$ is equivalent to $f\ x = \texttt{return}\ x$

$$\textsc{Extend}\ \frac{\Pi \vdash s \approx s' \quad g\ \text{fresh}}{\Pi, g\ x = t \vdash s \approx s'}$$

$$\Pi := \begin{array}{l} f\ x = y = x; \texttt{return}\ y \\ g\ x = \texttt{return}\ x \end{array}$$

$$\frac{\dfrac{}{\overline{f\ x = s} \vdash s_i \approx s_j}}{\overline{f\ x = s} \vdash \texttt{goto}\ f_i\ x \approx \texttt{goto}\ f_j\ x}$$

$f\,x = y = x; \texttt{return}\,y$ is equivalent to $f\,x = \texttt{return}\,x$

$$\textsc{Extend}\ \frac{\Pi \vdash s \approx s' \quad g \text{ fresh}}{\Pi, g\,x = t \vdash s \approx s'}$$

$$\Pi := \begin{array}{l} f\,x = y = x; \texttt{return}\,y \\ g\,x = \texttt{return}\,x \end{array}$$

$$\frac{\overline{f\ x = s} \vdash s_i \approx s_j}{\overline{f\ x = s} \vdash \texttt{goto}\,f_i\,x \approx \texttt{goto}\,f_j\,x}$$

$$\emptyset \vdash \quad y = x; \texttt{return}\,y \ \approx\ \texttt{return}\,x$$

$f\,x = y = x; \texttt{return}\,y$ is equivalent to $f\,x = \texttt{return}\,x$

$$\textsc{Extend}\ \frac{\Pi \vdash s \approx s' \quad g \text{ fresh}}{\Pi, g\,x = t \vdash s \approx s'}$$

$$\Pi := \begin{array}{l} f\,x = y = x; \texttt{return}\,y \\ g\,x = \texttt{return}\,x \end{array}$$

$$\frac{\dfrac{\overline{f\ x = s} \vdash s_i \approx s_j}{\overline{f\ x = s} \vdash \texttt{goto}\,f_i\,x \approx \texttt{goto}\,f_j\,x}}{\emptyset \vdash y = x; \texttt{return}\,y \approx \texttt{return}\,x}$$

$\emptyset \vdash y = x; \texttt{return}\,y \approx \texttt{return}\,x$	Assumption
$\Pi \vdash y = x; \texttt{return}\,y \approx \texttt{return}\,x$	$\textsc{Extend}$
$\Pi \vdash \texttt{goto}\,f\,x \approx \texttt{goto}\,g\,x$	Lemma
$\Pi \vdash f \approx g$	Def. $\approx$

How to translate from an imperative language to IL?

Appel/Kelsey embedding [4]

"Any SSA program can be syntactically translated into an equivalent continuation passing style program"

- Via Appel/Kelsey embedding, if in SSA form
- How can we investigate the SSA construction in our setting?

Imperative Semantics

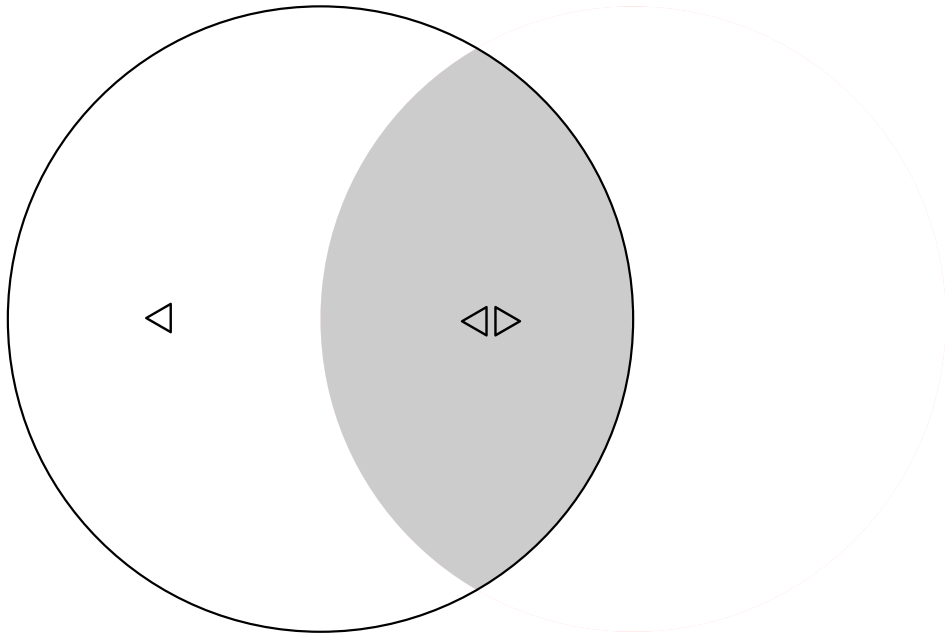
$$\textsc{Goto}\ \frac{\overline{f\ x = s}\ \vdash\ \sigma[x_i := \sigma\ y], s_i \lhd v}{\overline{f\ x = s}\ \vdash\ \sigma, \mathtt{goto}\ f_i\ y \lhd v}$$

$$\textsc{Subset}\ \frac{\Pi \vdash \sigma, s \rhd v}{\Pi \vdash \sigma, s \lhd v}$$

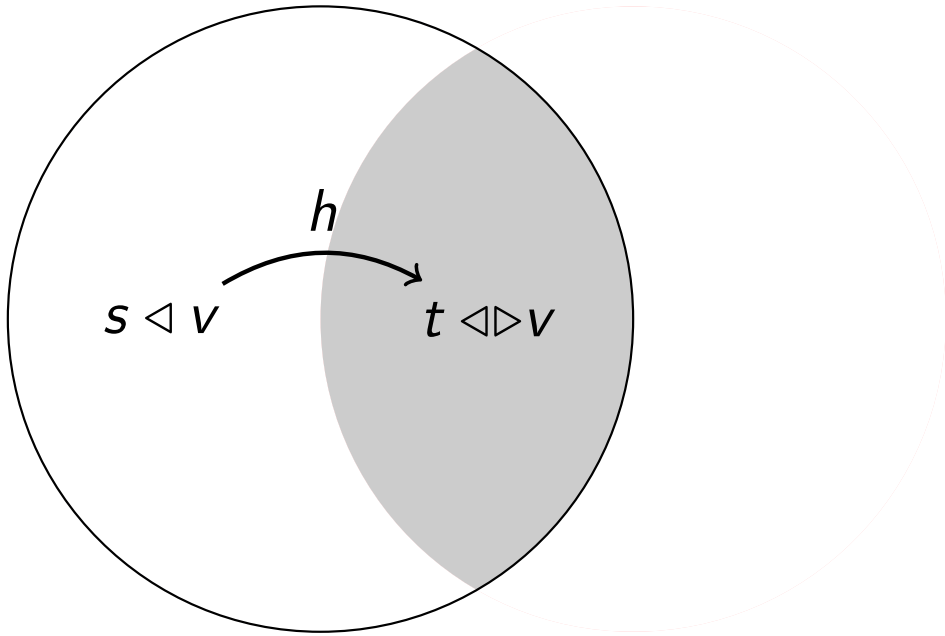
Counter-example for Superset property:

$$\begin{aligned} f\ x &= \mathtt{return}\ y \vdash \emptyset[y := 1], \mathtt{goto}\ f\ y \lhd 1 \\ f\ x &= \mathtt{return}\ y \vdash \emptyset[y := 1], \mathtt{goto}\ f\ y \not\rhd 1 \end{aligned}$$

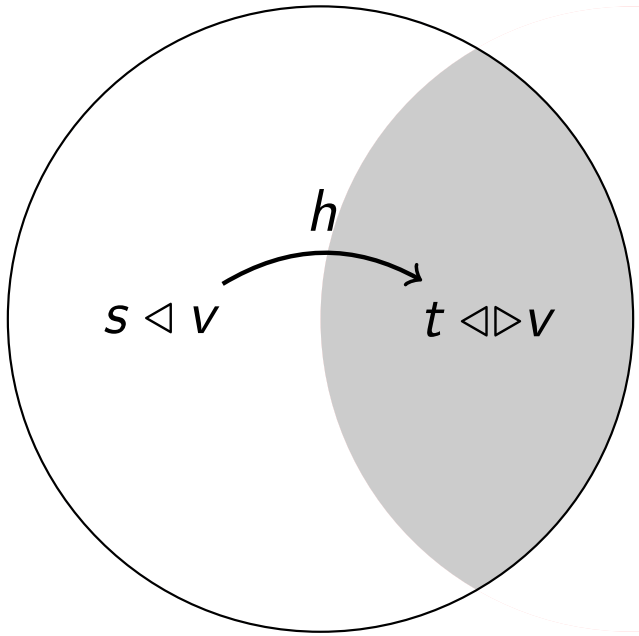
Imperative and Functional Semantics



Imperative and Functional Semantics

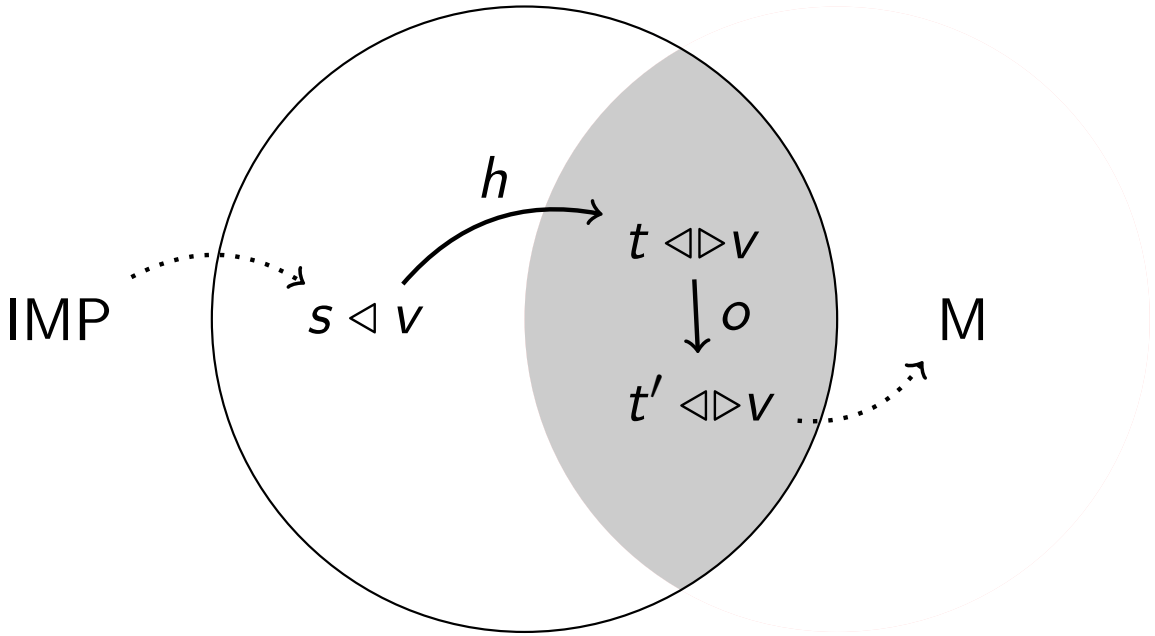


Imperative and Functional Semantics



essentially h is an SSA construction algorithm

Conclusion



h : SSA construction algorithm

o : optimizing program transformation

Thank you!

[1] Bowen Alpern, Mark N. Wegman, and F. Kenneth Zadeck. "Detecting Equality of Variables in Programs". In: *POPL*. 1988, pp. 1–11.

[2] Ron Cytron et al. "An Efficient Method of Computing Static Single Assignment Form". In: *POPL*. 1989, pp. 25–35.

[3] Andrew D. Gordon. "A Tutorial on Co-induction and Functional Programming". In: *Glasgow Functional Programming Workshop*. Springer, 1994, pp. 78–95.

[4] Richard A. Kelsey. "A correspondence between continuation passing style and static single assignment form". In: *SIGPLAN Not.* 30 (3 Mar. 1995), pp. 13–22. ISSN: 0362-1340.

[5] J.H. Morris. "Lambda-calculus models of programming languages". PhD Thesis. MIT, 1968.

[6] Barry K. Rosen, Mark N. Wegman, and F. Kenneth Zadeck. "Global Value Numbers and Redundant Computations". In: *POPL*. 1988, pp. 12–27.

Example

Program

```
1  // n input
2  entry:   i = 1;
3  loop :   if n = 0 then
4             return i
5           else
6             i = n * i;
7             n = n - 1;
8             goto loop
```

Program in SSA form

```
1  // n input
2  entry =
3    i = 1;
4    goto loop ()
5
6  loop _ =
7    if n = 0 then
8      return i
9    else
10     i = n * i;
11     n = n - 1;
12     goto loop ()
```

Conjecture

$$\eta, \overline{f\ x = s} \vdash \sigma, t$$

$$\textsc{Goto}\ \frac{\eta, \overline{f\ x = s} \vdash \eta[x_n := \sigma\ y], s_n \triangleright v}{\eta, \overline{f\ x = s} \vdash \sigma, \mathtt{goto}\ f_n\ y \triangleright v}$$

$$\textsc{Goto}\ \frac{\overline{f\ x = s} \vdash \sigma[x_i := \sigma\ y], s_i \triangleleft v}{\overline{f\ x = s} \vdash \sigma, \mathtt{goto}\ f_i\ y \triangleleft v}$$

Conjecture

For all σ, s, Π in SSA form,

$$\sigma, \Pi \vdash \sigma, s \triangleright v \iff \sigma, \Pi \vdash \sigma, s \triangleleft v$$

Example

Goal: $f\,x = y = x; \texttt{return}\,y$ is equivalent to $f\,x = \texttt{return}\,x$.

$$\textsc{Extract}$$
$$\forall\sigma,\ \Pi|_g \vdash \sigma, \texttt{goto}\,g\,x \triangleright v \ \leftrightarrow\ \Pi \vdash \sigma, \texttt{goto}\,g\,x \triangleright v$$

$$\Pi := \begin{array}{l} f\,x = y = x; \texttt{return}\,y \\ g\,x = \texttt{return}\,x \end{array}$$

By $\textsc{Extract}$ and Def. $\approx$, we get for all σ,

$$\begin{array}{lll} & g\,x = \texttt{return}\,x \vdash \sigma, \texttt{goto}\,g\,x \triangleright v & \\ \leftrightarrow & \Pi \vdash \sigma, \texttt{goto}\,g\,x \triangleright v & \textsc{Extract} \\ \leftrightarrow & \Pi \vdash \sigma, \texttt{goto}\,f\,x \triangleright v & \Pi \vdash f \approx g \\ \leftrightarrow & f\,x = y = x; \texttt{return}\,y \vdash \sigma, \texttt{goto}\,f\,x \triangleright v & \textsc{Extract} \end{array}$$