Simple Generation of Static Single Assignment Form

John Aycock
Nigel Horspool

University of Victoria
Canada

STATIC SINGLE ASSIGNMENT FORM

- Each use of a variable corresponds to a unique definition point for that variable.
- SSA form is obtained by renaming variables and by inserting ϕ-functions at places where control flow paths merge.

EXAMPLE OF SSA FORM

```
  /* IR BEFORE */
  i ← 999
  i ← i + 1
  i ← i * j

L1:

  if (i < 0)
    goto L2

  if (i > 10)
    i ← i - 5
  else
    i ← i - 1


  goto L1
L2:
```

```
  /* IR AFTER */
  i1 ← 999
  i2 ← i1 + 1
  i3 ← i2 * j1

L1:
  i4 ← φ(i3,i7)
  if (i4 < 0)
    goto L2

  if (i4 > 10)
    i5 ← i4 - 5
  else
    i6 ← i4 - 1
  i7 ← φ(i5,i6)

  goto L1
L2:
```

INTRODUCTION

- SSA form with the minimal number of ϕ-functions is highly desirable.
- Several algorithms for ϕ-function placement exist; we present yet another algorithm.
- We believe ours is *much* simpler; it achieves the minimum number of ϕ-functions for reducible flowgraphs.
  (It may not be minimal for irreducible flowgraphs.)

OVERVIEW OF THE MINIMIZATION ALGORITHM

1. Initially create correct but non-minimal SSA form by inserting ϕ-functions almost everywhere.

2. Iteratively remove the excess ϕ-functions by performing a *minimization* phase.

```
i ← 123
j ← i * j
repeat


  write j
  if (j > 5) then


     i ← i + 1
  else


     break
  endif


until (i > 234)
```

Initial Version of Code

Brute Force Conversion to SSA Form

(Appel 98)

```
i0 ← ⊥
j0 ← ⊥
i1 ← 123
j1 ← i1 * j0
repeat
   i2 ← φ(i1, i6)
   j2 ← φ(j1, j5)
   write j2
   if (j2 > 5) then
       i3 ← φ(i2)
       j3 ← φ(j2)
       i4 ← i3 + 1
   else
       i5 ← φ(i2)
       j4 ← φ(j2)
       break
   endif
   i6 ← φ(i4)
   j5 ← φ(j3)
until (i6 > 234)
i7 ← φ(i6, i5)
j6 ← φ(j5, j4)
```

THE REDUCTION RULES

1. Delete all ϕ-assignments of the form

$$V_i \leftarrow \phi(\ldots V_i \ldots V_i \ldots)$$

2. If a ϕ-assignment has the form

$$V_i \leftarrow \phi(\ldots V_i \ldots V_j \ldots V_i \ldots V_j \ldots)$$

then delete that assignment and rename all occurrences of V_i to V_j.

[I.e. the arguments of the ϕ-function consist of 0 or more occurrences of V_i and 1 or more occurrences of another variable V_j.]

$i_0 \leftarrow \perp$
$j_0 \leftarrow \perp$
$i_1 \leftarrow 123$
$j_1 \leftarrow i_1 * j_0$
repeat
  $i_2 \leftarrow \phi(i_1, i_6)$
  $j_2 \leftarrow \phi(j_1, j_5)$
  write j_2
  if ($j_2 > 5$) then
    $i_3 \leftarrow \phi(i_2)$
    $j_3 \leftarrow \phi(j_2)$
    $i_4 \leftarrow i_3 + 1$
  else
    $i_5 \leftarrow \phi(i_2)$
    $j_4 \leftarrow \phi(j_2)$
    break
  endif
  $i_6 \leftarrow \phi(i_4)$
  $j_5 \leftarrow \phi(j_3)$
until ($i_6 > 234$)
$i_7 \leftarrow \phi(i_6, i_5)$
$j_6 \leftarrow \phi(j_5, j_4)$

0

(THE START)

$i_0 \leftarrow \perp$
$j_0 \leftarrow \perp$
$i_1 \leftarrow 123$
$j_1 \leftarrow i_1 * j_0$
repeat
  $i_2 \leftarrow \phi(i_1, i_6)$
  $j_2 \leftarrow \phi(j_1, j_5)$
  write j_2
  if ($j_2 > 5$) then
    ~~$i_3 \leftarrow \phi(i_2)$~~
    ~~$j_3 \leftarrow \phi(j_2)$~~
    $i_4 \leftarrow i_2 + 1$
  else
    ~~$i_5 \leftarrow \phi(i_2)$~~
    ~~$j_4 \leftarrow \phi(j_2)$~~
    break
  endif
  $i_6 \leftarrow \phi(i_4)$
  $j_5 \leftarrow \phi(j_2)$
until ($i_6 > 234$)
$i_7 \leftarrow \phi(i_6, i_2)$
$j_6 \leftarrow \phi(j_5, j_2)$

1

```
i0 ← ⊥
j0 ← ⊥
i1 ← 123
j1 ← i1 * j0
repeat
   i2 ← φ(i1, i4)
   j2 ← φ(j1, j2)
   write j2
   if (j2 > 5) then
      i3 ← φ(i2)
      j3 ← φ(j2)
      i4 ← i2 + 1
   else
      i5 ← φ(i2)
      j4 ← φ(j2)
      break
   endif
   i6 ← φ(i4)
   j5 ← φ(j2)
until (i4 > 234)
i7 ← φ(i4, i2)
j6 ← φ(j2, j2)
```

2

$i_0 \leftarrow \perp$
$j_0 \leftarrow \perp$
$i_1 \leftarrow 123$
$j_1 \leftarrow i_1 * j_0$
repeat
  $i_2 \leftarrow \phi(i_1, i_4)$
  ~~$j_2 \leftarrow \phi(j_1, j_2)$~~
  write j_1
  if (j_1 > 5) then
    ~~$i_3 \leftarrow \phi(i_2)$~~
    ~~$j_3 \leftarrow \phi(j_2)$~~
    $i_4 \leftarrow i_2 + 1$
  else
    ~~$i_5 \leftarrow \phi(i_2)$~~
    ~~$j_4 \leftarrow \phi(j_2)$~~
    break
  endif
  ~~$i_6 \leftarrow \phi(i_4)$~~
  ~~$j_5 \leftarrow \phi(j_2)$~~
until (i_4 > 234)
$i_7 \leftarrow \phi(i_4, i_2)$
$j_6 \leftarrow \phi(j_1, j_1)$

3

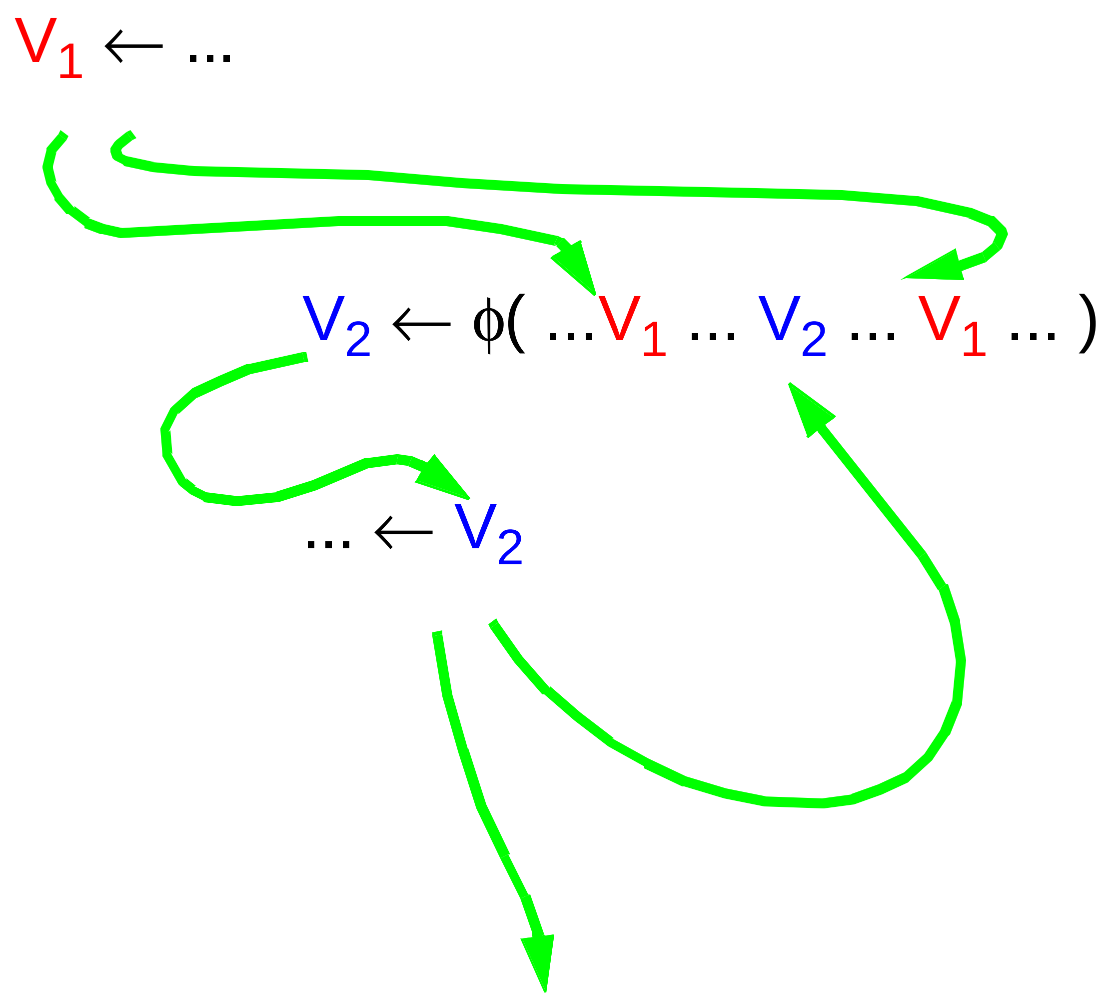
$i_0 \leftarrow \perp$
$j_0 \leftarrow \perp$
$i_1 \leftarrow 123$
$j_1 \leftarrow i_1 * j_0$
repeat
  $i_2 \leftarrow \phi(i_1, i_4)$
  ~~$j_2 \leftarrow \phi(j_1, j_2)$~~
  write j_1
  if ($j_1 > 5$) then
    ~~$i_3 \leftarrow \phi(i_2)$~~
    ~~$j_3 \leftarrow \phi(j_2)$~~
    $i_4 \leftarrow i_2 + 1$
  else
    ~~$i_5 \leftarrow \phi(i_2)$~~
    ~~$j_4 \leftarrow \phi(j_2)$~~
    break
  endif
  ~~$i_6 \leftarrow \phi(i_4)$~~
  ~~$j_5 \leftarrow \phi(j_2)$~~
until ($i_4 > 234$)
$i_7 \leftarrow \phi(i_4, i_2)$
~~$j_6 \leftarrow \phi(j_1, j_1)$~~

4

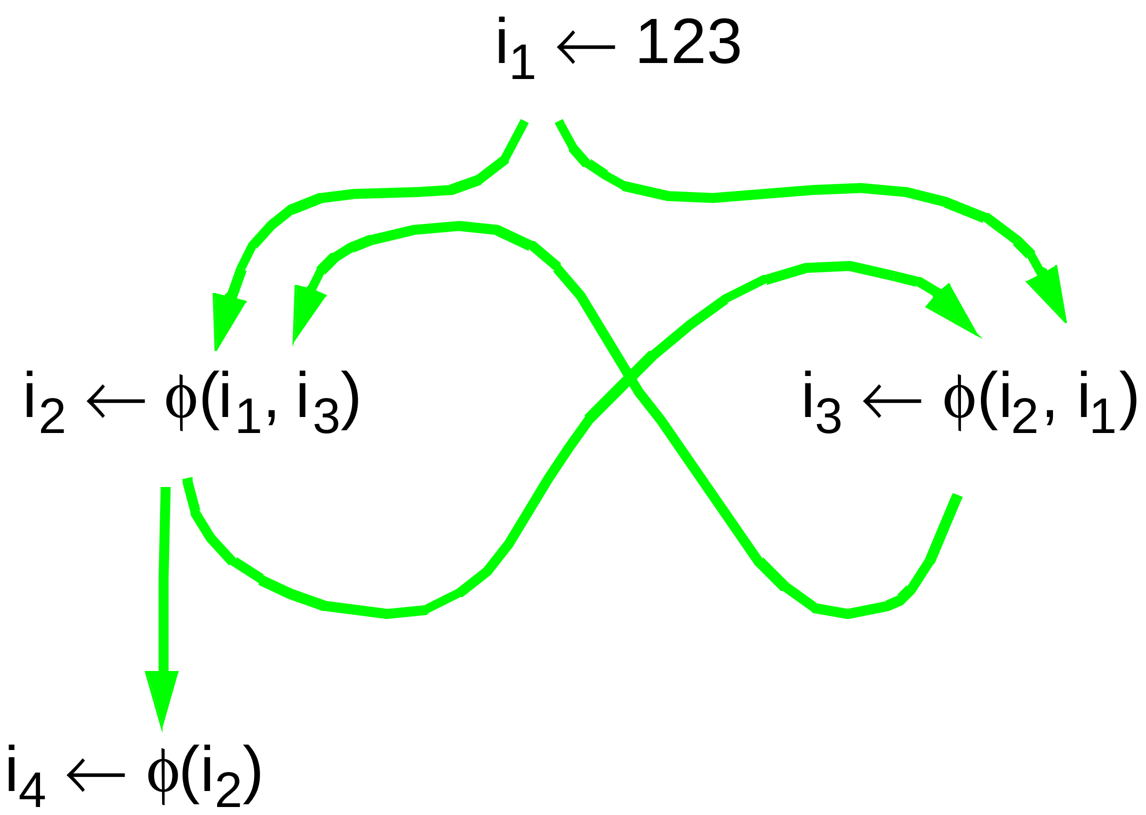
(THE FINISH)

and subsequent uses of j_6 are renamed to j_1

Is The Rule Correct?



IRREDUCIBLE FLOWGRAPHS?



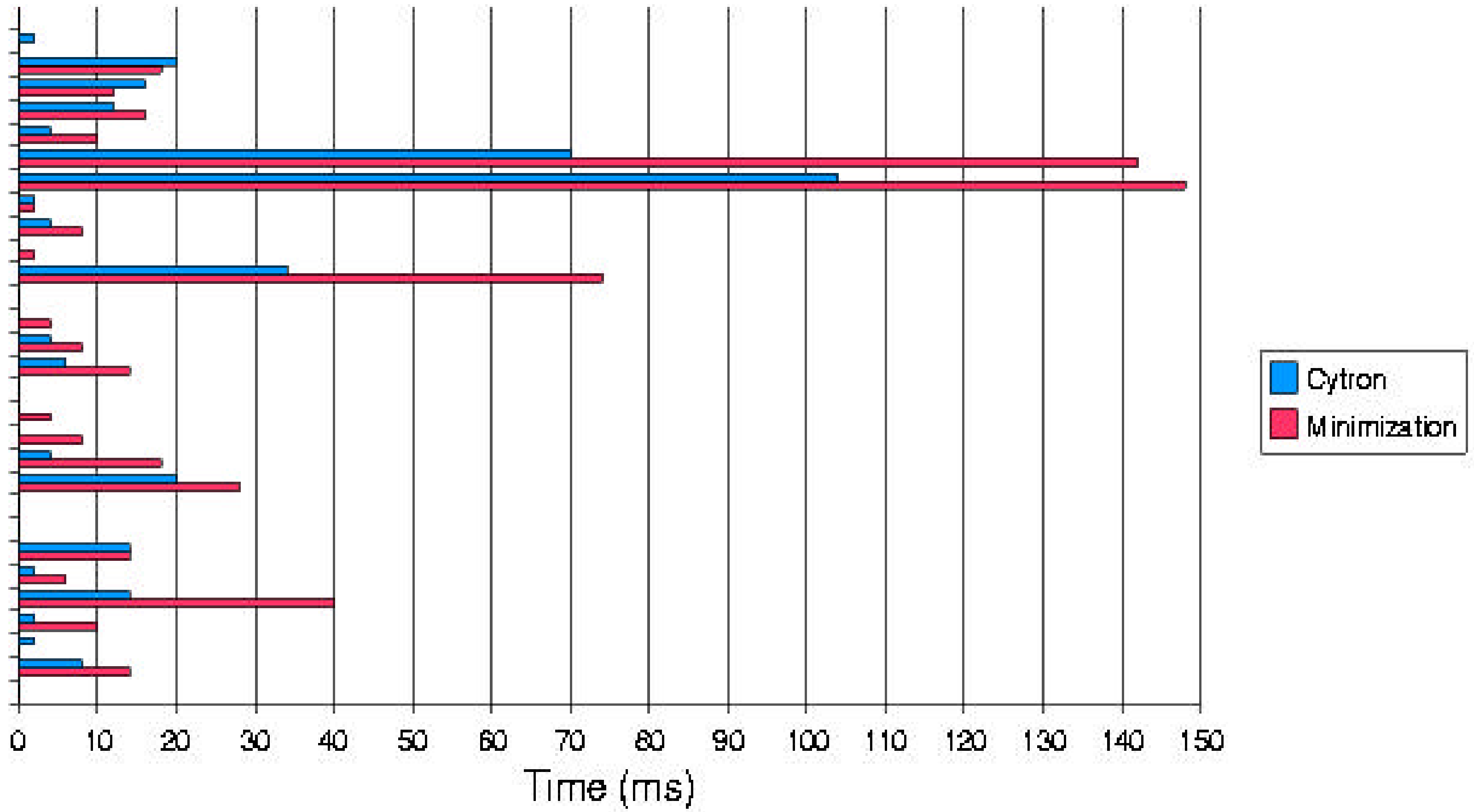
We clearly miss the simplification here.

MINIMALITY?

- Obviously not minimal for irreducible flowgraphs, but the placement of ϕ-functions is still correct.
- A proof of minimality is shown in the paper.

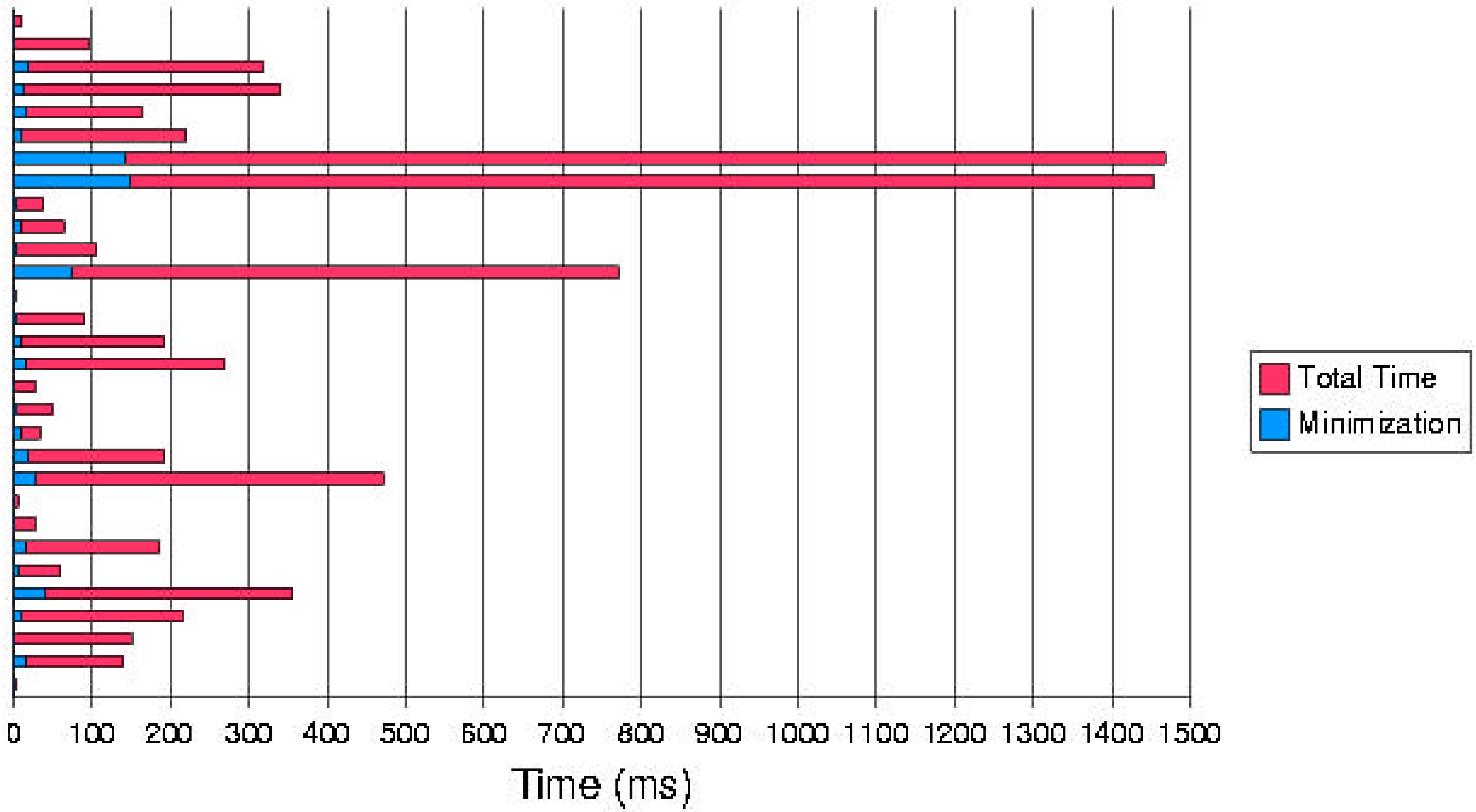
PERFORMANCE

Time Comparison with Cytron's Approach



DOES SPEED MATTER?

SSA generation time compared to whole compilation



SUMMARY

- The minimization approach to SSA generation is competitive in speed.
- Even when the simple approach is slower, the effect on the overall compilation time is insignificant.