

Compiler Optimisation

4-from-ssa – Conversion from SSA (addendum)

Hugh Leather

IF 1.18a

hleather@inf.ed.ac.uk

Institute for Computing Systems Architecture
School of Informatics
University of Edinburgh

2017

Introduction

Things to watch out for when converting from SSA.

- Effect of optimisation
- Critical edges
- Lost copy problem
- Swap problem

Effect of Optimisation

Optimisations can prevent conversion by just merging variables

Example

```
a = x + y
```

```
b = x + y
```

```
a = 17
```

```
c = x + y
```

Just a basic block

Effect of Optimisation

Optimisations can prevent conversion by just merging variables

Example

$$a_0 = x_0 + y_0$$

$$b_0 = x_0 + y_0$$

$$a_1 = 17$$

$$c_0 = x_0 + y_0$$

Convert to SSA.
Note that b_0 and c_0 are copies of a_0

Effect of Optimisation

Optimisations can prevent conversion by just merging variables

Example

$$a_0 = x_0 + y_0$$

$$b_0 = a_0$$

$$a_1 = 17$$

$$c_0 = a_0$$

Optimise the redundant expressions.
What will happen if we merge variables
now?

Effect of Optimisation

Optimisations can prevent conversion by just merging variables

Example

$a = x + y$

$b = a$

$a = 17$

$c = a$ $(x+y)$

If we merge a_0 and a_1 back into a , then c gets the wrong value

So, keep variables, use copies in predecessors of ϕ nodes¹

¹As in lecture-3.

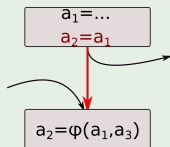
Critical Edges

Copies on predecessors difficult with *critical edges*.

Critical Edge

A CFG edge whose destination has multiple predecessors and whose source has multiple successors.

Example



Source has multiple successors: a copy in the source means all of its successors get the copy. If the copy is live into them then potential semantics change.

Destination has multiple predecessors: If there was only one, we could put the copy in the destination and probably wouldn't need the phi node anyway

Lost copy problem

- Most SSA algorithms *split* critical edges
- Next example shows necessary splitting to prevent lost copy

Lost copy problem

Example

```
i = 1
```

```
y = i
```

```
i = i+1
```

```
z = y + ..
```

A simple loop

Convert to SSA

Lost copy problem

Example

$i_0 = 1$

$i_1 = \phi(i_0, i_2)$

$y_0 = i_1$

$i_2 = i_1 + 1$

$z_0 = y_0 + \dots$

Converted to SSA

y_0 now redundant

Optimisation: *Replace uses with i_1 and remove definition*

Lost copy problem

Example

$i_0 = 1$

$i_1 = \phi(i_0, i_2)$

$i_2 = i_1 + 1$

$z_0 = i_1 + ..$

y_0 removed

*Try to convert from
SSA*

*Place copies without
splitting*

Lost copy problem

Example

$$i_0 = 1$$

$$i_1 = i_0$$

$$i_1 = \phi(i_0, i_2)$$

$$i_2 = i_1 + 1$$

$$i_1 = i_2$$

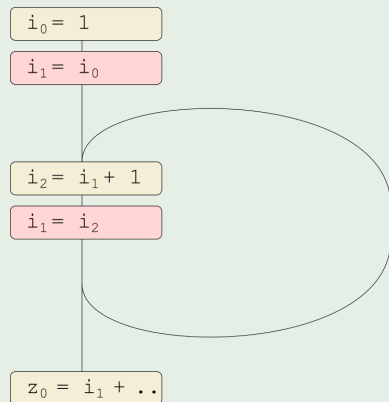
$$z_0 = i_1 + \dots$$

Copies placed

Now remove ϕ

Lost copy problem

Example



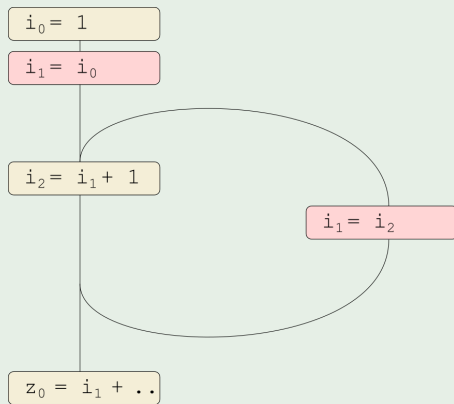
Note: Back edge is **critical** and i_1 is live in to loop exit

Does z_0 use the same version of i_1 as before the copy?

Instead, split loop's back edge

Lost copy problem

Example



Edge split keeps semantics

Extra jump can be expensive inside hot loops

Instead, use temporaries to remember correct values

Lost copy problem

Example

$i_0 = 1$

$i_1 = i_0$

$i_2 = i_1 + 1$

$t = i_1$

$i_1 = i_2$

$z_0 = t + \dots$

Extra temporary
in place

Swap problem

- ϕ nodes execute simultaneously in parallel
 - i.e. All read their operands at once, before any assignments
- Copies do not
 - Naive conversion with copies can cause incorrect behaviour

Example

Simultaneous phi,
swap values

$$x_1 = \phi(x_0, y_1)$$

$$y_1 = \phi(y_0, x_1)$$

Naive copy,
swap lost²

$$x_1 = y_1$$


$$y_1 = x_1$$

Temporary inserted

$$t = x_1$$

$$x_1 = y_1$$

$$y_1 = t$$

²Assume $x_1 = x_0, y_1 = y_0$ placed in another block. 

PPar CDT Advert

EPSRC Centre for Doctoral Training in Pervasive Parallelism

- 4-year programme:
MSc by Research + PhD
- Research-focused:
Work on your thesis topic
from the start
- Collaboration between:
 - ▶ University of Edinburgh's
School of Informatics
 - * Ranked top in the UK by
2014 REF
 - ▶ Edinburgh Parallel Computing
Centre
 - * UK's largest supercomputing
centre
- Research topics in software,
hardware, theory and
application of:
 - ▶ Parallelism
 - ▶ Concurrency
 - ▶ Distribution
- Full funding available
- Industrial engagement
programme includes
internships at leading
companies

The biggest revolution
in the technological
landscape for fifty years

Now accepting applications!
Find out more and apply at:

pervasiveparallelism.inf.ed.ac.uk



THE UNIVERSITY of EDINBURGH
informatics

EPSRC

Engineering and Physical Sciences
Research Council