On the use of SSA with Scripting Languages

Paul Biggar and David Gregg

Department of Computer Science and Statistics
Trinity College Dublin

Static Single-Assignment Form Seminar
Autrans, France
27th April, 2009

Motivating Example

```
1  function log ($printer, $prefix, $message) {
2     $fout = "$prefix: $message";
3     $printer->file_print ($fout);
4
5     $cout = "$prefix: $message"
6     $printer->console_print ($cout);
7  }
```

In SSA

```
1  function log ($printer_0, $prefix_0, $message_0) {
2    $fout_0 = $prefix_0 . ": " . $message_0;
3    $printer_0->file_print ($fout_0);
4
5    $cout_0 = $prefix_0 . ": " . $message_0;
6    $printer_0->console_print ($cout_0);
7  }
```

Value numbering

```
1  function log ($printer_0, $prefix_0, $message_0) {
2    $fout_0 = $prefix_0 . ": " . $message_0;
3    $printer_0->file_print ($fout_0);
4
5    $printer_0->console_print ($fout_0);
6  }
```

Aliased parameters?

```
1  function log ($printer, $prefix, $message) {
2      ...
3  }
4
5  $p = new Printer;
6  log ($p, &$p->pre, &$p->mes);
```

References in PHP

- Java style

References in PHP

- Java style
- C++ style

References in PHP cont.

```
1 $y = 1;
2 if (...)
3     $x =& $y;
4 else
5     $x = $y;
6
7 $x = 5;
8 print $y;
```

Aliased parameters?

```
1  function log ($printer, $prefix, $message) {
2      ...
3  }
4
5  $p = new Printer;
6  log ($p, &$p->pre, &$p->mes);
```

SSA + Alias analysis

- What form of SSA to support alias analysis?

SSA + Alias analysis

- What form of SSA to support alias analysis?

```
http://www.cs.man.ac.uk/~jsinger/ssa.html
```

SSA + Alias analysis

- What form of SSA to support alias analysis?
  - Dynamic Single Assignment

Paul Feautrier. Dataflow analysis of array and scalar references. International Journal of Parallel Programming, 1991.

SSA + Alias analysis

- What form of SSA to support alias analysis?
  - ~~Dynamic Single Assignment~~
  - Cytron and Gershbein

Ron Cytron and Reid Gershbein. Efficient accommodation of may-alias information in SSA form. PLDI 1993.

SSA + Alias analysis

- What form of SSA to support alias analysis?
  - ~~Dynamic Single Assignment~~
  - ~~Cytron and Gershbein~~
  - Extended SSA Numbering

Christopher Lapkowski and Laurie J. Hendren. Extended SSA numbering: Introducing SSA properties to language with multi-level pointers. Compiler Construction, 1998.

SSA + Alias analysis

- What form of SSA to support alias analysis?
  - ~~Dynamic Single Assignment~~
  - ~~Cytron and Gershbein~~
  - ~~Extended SSA Numbering~~
  - Extended Array SSA

Stephen Fink, Kathleen Knobe, and Vivek Sarkar. Unified analysis of array and object references in strongly typed languages. Static Analysis Symposium, 2000.

SSA + Alias analysis

- What form of SSA to support alias analysis?
  - ~~Dynamic Single Assignment~~
  - ~~Cytron and Gershbein~~
  - ~~Extended SSA Numbering~~
  - ~~Extended Array SSA~~
  - Hashed SSA

Fred C. Chow, Sun Chan, Shin-Ming Liu, Raymond Lo, and Mark Streich. Effective representation of aliases and indirect memory operations in SSA form. Compiler Construction, 1996.

What is HSSA?

- Virtual variables

What is HSSA?

- Virtual variables
- Mu: may-use

What is HSSA?

- Virtual variables
- Mu: may-use
- Chi: may-def

What is HSSA?

- Virtual variables
- Mu: may-use
- Chi: may-def
- Space efficient representation

What is HSSA?

- Virtual variables
- Mu: may-use
- Chi: may-def
- Space efficient representation
- Drop indices to get out of SSA

What is HSSA?

- Virtual variables
- Mu: may-use
- Chi: may-def
- Space efficient representation
- Drop indices to get out of SSA
- Must be careful not to move copies across live ranges

Aliased parameters in SSA

```
1  function log ($printer_0, $prefix_0, $message_0) {
2    MU ($printer_0)
3    $fout_0 = $prefix_0 . ": " . $message_0;
4
5    $printer_0->file_print ($fout_0);
6    $printer_1 = CHI ($printer_0);
7    $prefix_1 = CHI ($prefix_0);
8    $message_1 = CHI ($message_0);
9    $fout_1 = CHI ($fout_0);
10
11   MU ($printer_1)
12   MU ($fout_1)
13   $cout_0 = $prefix_1 . ": " . $message_1;
14
15   $printer_0->console_print ($cout_0);
16   ...
17 }
```

Implication

Conservative SSA form is very pessimistic

Simpler?

```
1  function bastardized_mandel ($n)
2  {
3     for ($y = 0; $y <= $n; $y++)
4     {
5        $imc = 0.28 * ($y - 12);
6        for ($x = 0; $x <= 150; $x++)
7        {
8           $rec = 0.28 * ($x - 40) - 0.45;
9           $re = $rec;
10          $im = $imc;
11          $color = 10;
12          $re2 = $re * $re;
13          $im2 = $im * $im;
14       }
15    }
16 }
```

C API handlers

- read_property
- read_dimension
- get
- set
- cast_object
- has_property
- unset_property
- ...

Mandelbrot again

```
1  function bastardized_mandel ($n)
2  {
3    $y = 0;
4
5    while (1)
6    {
7      if ($y > $n)
8        break;
9
10     $imc = 0.28 * ($y - 12);
11     ...
12     $y++;
13   }
14 }
15
16 bastardized_mandel (extension_function ());
```

Mandelbrot in SSA

```
1  function bastardized_mandel ($n_0)
2  {
3     $y_0 = 0;
4
5     $y_1 = PHI ($y_0, $y_X)
6     $n_1 = PHI ($n_0, $n_X)
7     while (1)
8     {
9        $y_2 = CHI ($y_1);
10       if ($y_2 > $n_1)
11          break;
12
13       $imc_1 = CHI ($imc_0);
14       $imc_1 = 0.28 * ($y_2 - 12);
15       $y_3 = CHI ($y_2);
16       $imc_2 = CHI ($imc_1);
17
18       ...
19    }
20 }
```

Unknown types propagate

- local symbol table
- global symbol table
- return values
- reference parameters
- callee parameters

Implication

Def-use chains cannot be trivially obtained without analysis
even for scalars!!

SSA in phc

- Intra-procedural (only) analysis

SSA in phc

- ~~Intra-procedural (only) analysis~~
- Derive def-use chains from whole-program analysis

SSA in phc

- ~~Intra-procedural (only) analysis~~
- Derive def-use chains from whole-program analysis
  - Abstract Execution / Interpretation
  - Points-to analysis
  - Conditional Constant-propagation
  - Type-inference

Conditional Pointer Aliasing and Constant Propagation. Anthony Pioli. MS Thesis, SUNY at New Paltz Technical Report #99-102, January 1999.

Benefits of SSA

- End-to-end compiler IR

Benefits of SSA

- ~~End-to-end compiler IR~~
- Sparse propagation framework

Benefits of SSA

- ~~End-to-end compiler IR~~
- ~~Sparse propagation framework~~
- Sparse analysis framework (execution-time)

Benefits of SSA

- ~~End-to-end compiler IR~~
- ~~Sparse propagation framework~~
- Sparse analysis framework (execution-time)
- Sparse representation (memory usage)

Open research problem (I think)

- Perform analyses on “SSA” while building SSA
  - Integrate SSA building into the abstract execution
  - Intuitively might be possible.

Misc

- Userspace handlers - syntax hides function calls.

Misc

- Userspace handlers - syntax hides function calls.
- Renaming not possible

Summary

- SSA is hard in scripting languages
- Perform propagation algorithm and alias analysis before SSA construction
- Can still use SSA for other analyses

Thanks

Q.

What else am I an expert in?

A.

Um, I suppose, maybe, scripting languages?

- Compiler research landscape
- (Informal) Semantics
- Optimization and analysis techniques