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
function log ($printer, $prefix, $message) {
    $fout = "$prefix: $message";
    $printer->file_print ($fout);

    $cout = "$prefix: $message"
    $printer->console_print ($cout);
}
In SSA

```php
function log ($printer_0, $prefix_0, $message_0) {
    $fout_0 = $prefix_0 . ": " . $message_0;
    $printer_0->file_print ($fout_0);
    
    $cout_0 = $prefix_0 . ": " . $message_0;
    $printer_0->console_print ($cout_0);
}
```
function log ($printer_0, $prefix_0, $message_0) {
    $fout_0 = $prefix_0 . ": " . $message_0;
    $printer_0->file_print ($fout_0);
    $printer_0->console_print ($fout_0);
}
function log ($printer, $prefix, $message) {
    ...
}
$p = new Printer;
log ($p, &$p->pre, &$p->mes);
References in PHP

- Java style
References in PHP

- Java style
- C++ style
$y = 1;

if (...) {
    $x = & $y;
} else {
    $x = $y;
}

$x = 5;

print $y;
function log ($printer, $prefix, $message) {
    ...
}
What form of SSA to support alias analysis?
What form of SSA to support alias analysis?

http://www.cs.man.ac.uk/~jsinger/ssa.html
What form of SSA to support alias analysis?
- Dynamic Single Assignment

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.
What form of SSA to support alias analysis?

- Dynamic Single Assignment
- Cytron and Gershbein
- Extended SSA Numbering

What form of SSA to support alias analysis?

- Dynamic Single Assignment
- Cytron and Gershbein
- Extended SSA Numbering
- Extended Array SSA

What form of SSA to support alias analysis?

- Dynamic Single Assignment
- Cytron and Gershbein
- Extended SSA Numbering
- Extended Array SSA
- Hashed SSA

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
function log ($printer_0, $prefix_0, $message_0) {
    MU ($printer_0)
    $fout_0 = $prefix_0 . ": " . $message_0;

    $printer_0->file_print ($fout_0);
    $printer_1 = CHI ($printer_0);
    $prefix_1 = CHI ($prefix_0);
    $message_1 = CHI ($message_0);
    $fout_1 = CHI ($fout_0);

    MU ($printer_1)
    MU ($fout_1)
    $cout_0 = $prefix_1 . ": " . $message_1;

    $printer_0->console_print ($cout_0);
    ...
}
Conservative SSA form is very pessimistic
function bastardized_mandel ($n)
{
for ($y = 0; $y <= $n; $y++)
{
    $imc = 0.28 * ($y - 12);
    for ($x = 0; $x <= 150; $x++)
    {
        $rec = 0.28 * ($x - 40) - 0.45;
        $re = $rec;
        $im = $imc;
        $color = 10;
        $re2 = $re * $re;
        $im2 = $im * $im;
    }
}
}
C API handlers

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

Trinity College Dublin
function bastardized_mandel ($n)
{
    $y = 0;

    while (1)
    {
        if ($y > $n)
            break;

        $imc = 0.28 * ($y - 12);

        ...

        $y++;
    }

    bastardized_mandel (extension_function ());
function bastardized_mandel ($n_0) {
    $y_0 = 0;

    $y_1 = PHI ($y_0, $y_X);
    $n_1 = PHI ($n_0, $n_X)

    while (1) {
        $y_2 = CHI ($y_1);
        if ($y_2 > $n_1)
            break;

        $imc_1 = CHI ($imc_0);
        $imc_1 = 0.28 * ($y_2 - 12);
        $y_3 = CHI ($y_2);
        $imc_2 = CHI ($imc_1);

        ...
    }
}
Unknown types propagate

- local symbol table
- global symbol table
- return values
- reference parameters
- callee parameters
Def-use chains cannot be trivially obtained without analysis

*even for scalars!!*
Intra-procedural (only) analysis
Intra-procedural (only) analysis
Derive def-use chains from whole-program analysis
Intra-procedural (only) analysis

Derive def-use chains from whole-program analysis

- Abstract Execution / Interpretation
- Points-to analysis
- Conditional Constant-propagation
- Type-inference

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)
Perform analyses on “SSA” while building SSA
- Integrate SSA building into the abstract execution
- Intuitively might be possible.
Userspace handlers - syntax hides function calls.
Userspace handlers - syntax hides function calls.
Renaming not possible
SSA is hard in scripting languages
Perform propagation algorithm and alias analysis before SSA construction
Can still use SSA for other analyses
Q. What else am I an expert in?

A. Um, I suppose, maybe, scripting languages?
   - Compiler research landscape
   - (Informal) Semantics
   - Optimization and analysis techniques