Constant Optimizations

Simone Campanoni
simone.campanoni@northwestern.edu
Outline

• Constant propagation

• Constant folding

• Algebraic simplification
Constant propagation: problem definition

Given a program, we would like to know for every point in that program, which variables have constant values, and which ones do not.

A variable has a constant value at a certain point in the CFG if every execution that reaches that point sees that variable holding the same constant value.

We are now going to implement constant propagation automatically and by relying only on reaching definition.

i: x = 0
j: N = N + 1
k: Z = x + N
Reaching definition summary

• Reaching definition data-flow analysis computes \( \text{IN}[i] \) and \( \text{OUT}[i] \) for every instruction \( i \).

• \( \text{IN}[i] \) (\( \text{OUT}[i] \)) includes definitions that reach just before (just after) instruction \( i \).

• Each \( \text{IN}/\text{OUT} \) set contains a mapping for every variable in the program to a “value”.
Constant propagation

• For a use of variable $v$ by instruction $n$
  $n$: $x = \ldots v \ldots$
  • If the definitions of $v$ that reach $n$
    are all of the form
    $d$: $v = c$ (//c is a generic constant)
  • then replace
    the uses of $v$ in $n$ with $c$

Do you see any problem?

```
1: int x,y
2: x = 0
3: y = 0
4: if (a > b)
5: x = x +N
6: if (b > N)
7: return y
8: return x
```


Constant propagation problem?

1: int x, y
3: y = 0
4: If (a > b)

5: x = 5

IN[3] = {}
IN[4] = {3}

IN[6] = {3, 5}
6: If (b > N)

5: x = 5

IN[5] = {3}

IN[7] = {3, 5}
7: return y

IN[8] = {3, 5}
8: return 5

Is this correct?
Undefined behavior: a funny interpretation

• **Undefined behavior** is the result of executing a program whose behavior is unpredictable

• Undefined behavior results in whatever compilers want the program being compiled to do even *to make demons fly out of your nose*
  • Undefined behavior is often referred to as *nasal demons*
Constant propagation problem?

1: int x,y
3: y = 0
4: If (a > b)

Better solutions?
- Customize reaching definitions
- New analysis

5: x = 5

6: If (b > N)

7: return y

8: return x
Constant propagation for CAT

- Undefined values enable optimizations
- What about in the CAT language?
- CATData CAT_new (int64_t value);
SSA simplifies transformations

• We learned constant propagation that relies on reaching definition
  • This transformation is correct for both SSA and non-SSA IRs

• Can we have a faster constant propagation for SSA IRs?
  • Yes
  • Let’s first apply the previous constant propagation to an SSA IR to understand how to make it faster
Constant propagation in SSA (in LLVM)

If you want the conventional CP semantics:
- Skip undef

IN[%R2]={%3, %6, %7}
Constant propagation in SSA (in LLVM)

(Equivalent to SSA)

```
%E
%3 = icmp sgt %a, %b
br %3 %T %F

%T
br %F

%F
%6 = phi [undef, %E] [5, %T]
%7 = icmp sgt %b, %N
br %7 %R1 %R2

IN[%R2]={%3, %6, %7}

%R2
return 5

%R1
return 0

(Unnecessary thanks to SSA)
```
Outline

• Constant propagation

• Constant folding

• Algebraic simplification
Constant folding

**Definition:**
This transformation evaluates constant expressions at compile time so they do not need to be computed at runtime.

```
int a = 3 + 2;
myF(a);
```

```
int a = 5;
myF(a);
```

```
%a = add 3, 2
call @myF(%a)
```

```
%a = add 3, 2
call @myF(%a)
```

```cpp
Constant folding
for C variables
```

```cpp
Constant folding
for LLVM IR variables
```

```cpp
call @myF(5)
```
Outline

• Constant propagation

• Constant folding

• Algebraic simplification
Algebraic simplification

• **Definition:**
  Algebraic simplification uses algebraic properties of operators or particular operand combinations to simplify expressions.

• **Example:**

  int b = a + 0;  
  \[ \text{Algebraic simplification} \]  
  int b = a;  

  %b = add a, 0  
  call @myF(%b)  
  \[ \text{Algebraic simplification} \]  
  call @myF(%a)
Algebraic simplification

• **Definition:**
  Algebraic simplification uses algebraic properties of operators or particular operand combinations to simplify expressions

• **Example:**

  int b = a * 1;
  Algebraic simplification

  int b = a;

  %b = mul a, 1
  Algebraic simplification

  call @myF(%b)
  Algebraic simplification

  call @myF(%a)
Constant propagation  Constant folding  Algebraic simplification

Constant Optimizations
Always have faith in your ability

Success will come your way eventually

Best of luck!