Value Numbering

Мирошников Владислав
371

Value Numbering

- Метод определения эквивалентности двух вычислений и исключения одного из них
- Используется оптимизирующим компилятором
- Разбивает множество всех операций на классы конгруэнтности (эквивалентности)
- Номер класса конгруэнтности - **номер значения**

Задача:
Присвоить уникальный номер каждому значению, вырабатываемому внутри рассматриваемого участка программы

Использование

Используется для следующих оптимизаций:

- Распространение копий (copy propagation)
- Оптимизация условий
- Удаление общих подвыражений (CSE)

Local Value Numbering

- Работает в базовых блоках
- Для нумерации значений используется хэширование
- Для каждого выражения вычисляет хэш и смотрит на HashMap
- Хэш функция коммутативна относительно оператора выражения
- Для повторяющихся выражений выполняем подстановки

Пример LVN

```
a := i + 1
b := 1 + i
i := j
if i + 1 goto L1
c := i + 1
```

```
a := i + 1
b := a
i := j
t1 := i + 1
if t1 goto L1
c := t1
```

Другой пример

a ← 4 a is tagged as #1

b ← 5 b is tagged as #2

c ← a + b c (#1 + #2) is tagged as #3

d ← 5 d is tagged as #2, the same as b

e ← a + d e, being '#1 + #2' is tagged as #3

a ← 4

b ← 5

c ← a + b

d ← b

e ← c

Global Value Numbering

- Расширение LVN с базовых блоков на всю процедуру, нужен CFG
- Программа должна быть в SSA форме
- Два подхода: хэширование (аналогично LVN) и разбиение на классы конгруэнтности

GVN - хэширование

- Нужно дерево доминаторов - получается в результате преобразования к SSA
- Обходим блоки дерева в обратном порядке

```
DVNT_GVN(block b):
  for each phi node in b:
    remove and continue if meaningless or redundant
    set the value number for the remaining phi node to be the assigned variable name
    add phi node to the hash table

  for each assignment:
    get value numbers for each operand
    simplify the expression if possible
    if the expression has been computed before:
      set the value number for the assigned variable to the expression's value number
    else:
      set the value number for the expression to be the assigned variable name
      add the expression to the hash table

  for each child c of b in the control flow graph:
    replace all phi node operands in c that were computed in this block with their value numbers

  for each child c of b in the dominator tree:
    DVNT_GVN(c)


  remove all values hashed during this function call
```

Пример

```
w := 3
x := 3
y := x + 4
z := w + 4
```

```
w := 3
x := w
y := w + 4
z := y
```

Имеем отображение:

$$\{w \mapsto 1, x \mapsto 1, y \mapsto 2, z \mapsto 2\}$$

Copy propagation может удалить присваивания x и z

Если не будет в SSA форме?

```
a ← 1          a is tagged as #1
b ← 2          b is tagged as #2
c ← a + b      c is tagged as #3
b ← 3
d ← a + b      d is incorrectly tagged as #3
```

Разделение на классы конгруэнтности

- Идея: сделать две переменные конгруэнтными друг другу ⇔ если вычисления, которые их определяют, имеют одинаковые операторы (или константы) и их соответствующие операнды конгруэнтны

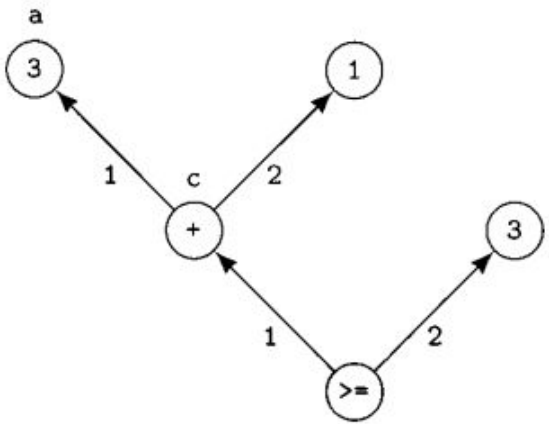
  $c \leftarrow a + 1 \text{ and } d \leftarrow b + 1$

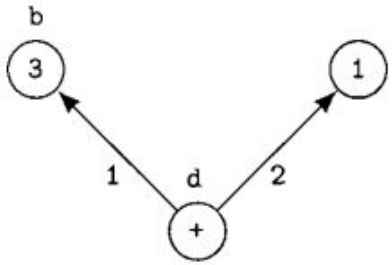
- Переводим CFG в SSA форму
- Нужно построить граф значений (Value Graph)

Пример

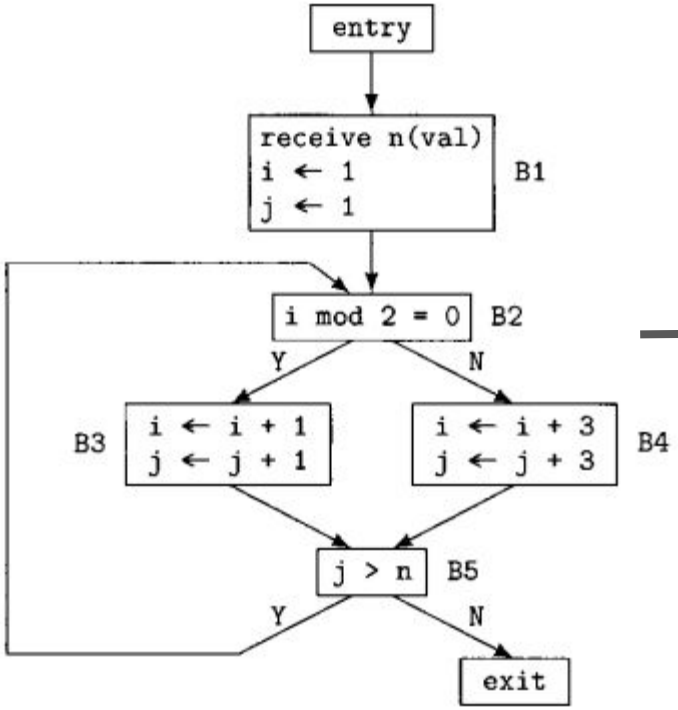
```
a ← 3
b ← 3
c ← a + 1
d ← b + 1
if c >= 3 then ...
```

Value Graph

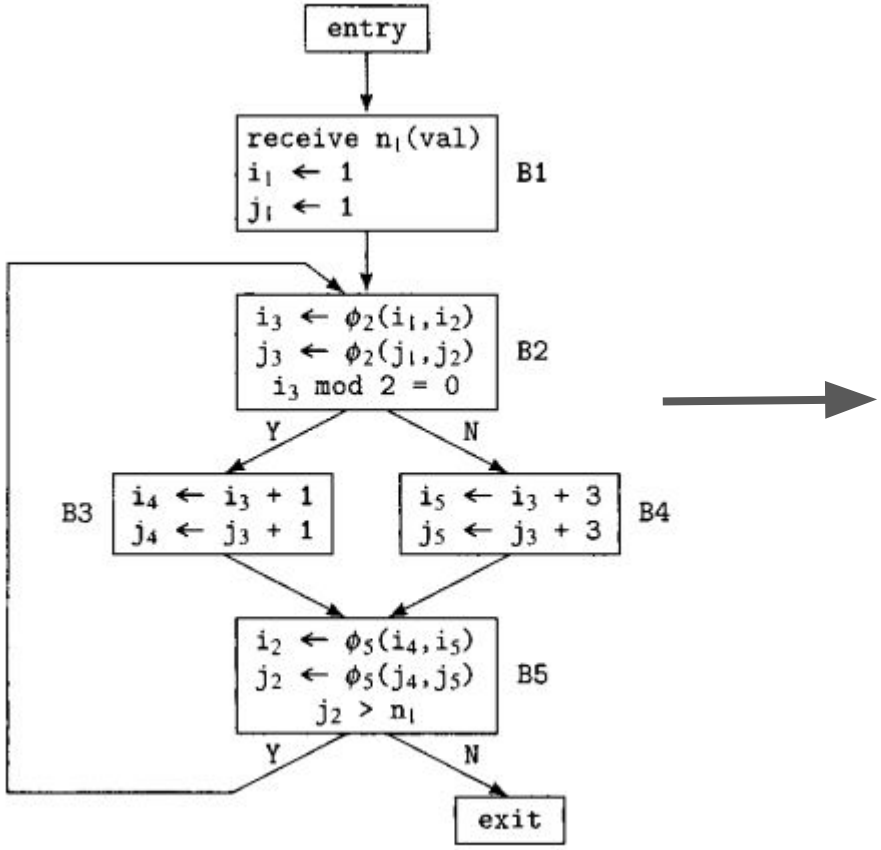




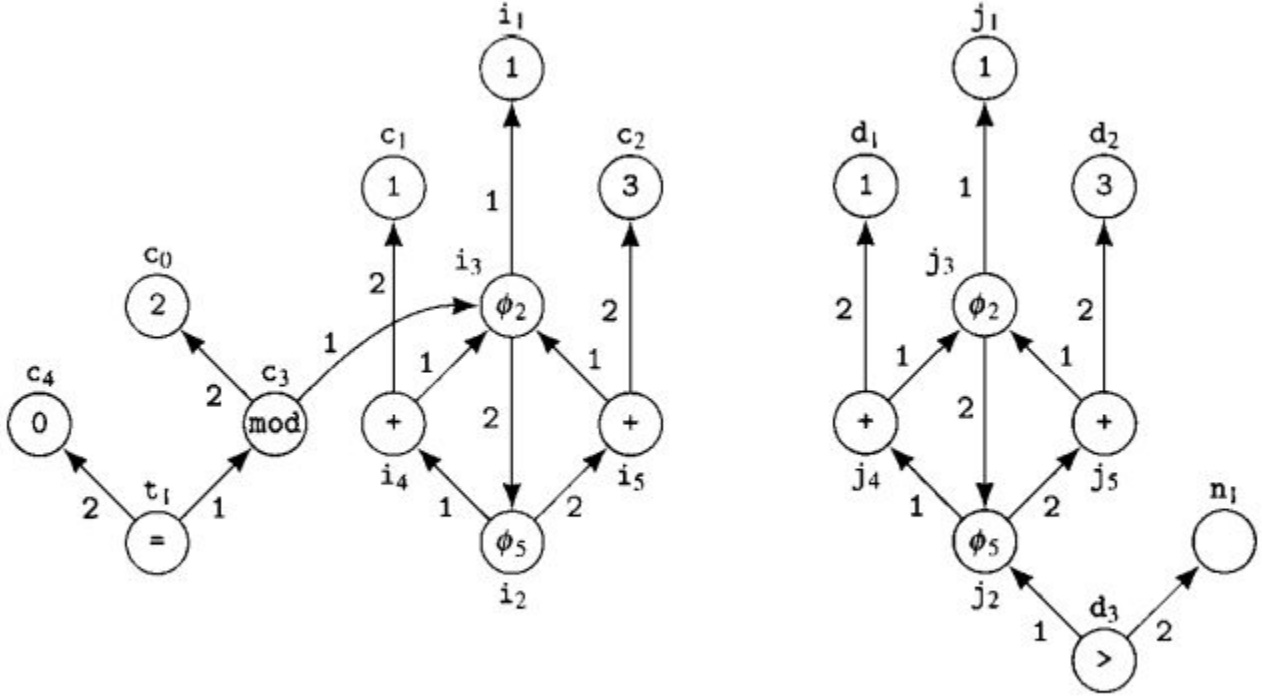
Другой пример



Example flowgraph for global value numbering.



Minimal SSA form for the flowgraph in Figure 12.18.



) Value graph for the code in Figure 12.19.

GVN и мощнее, и слабее других оптимизаций

```
read(i)
j ← i + 1
k ← i
l ← k + 1
```

(a)

```
i ← 2
j ← i * 2
k ← i + 2
```

(b)

```
    read(i)
    l ← 2 * i
    if i > 0 goto L1
    j ← 2 * i
    goto L2
L1: k ← 2 * i
L2:
```

(c)