

Global Value Numbering

COINS's GVN

- COINS compiler has two kinds of global value numbering (GVN):
 1. Basic GVN: eliminates common sub-expressions, and
 2. GVN based on question propagation: eliminates common sub-expressions and partially redundant expressions, and hoists loop-invariant expressions out of loops.

Basic GVN

Common sub-expression elimination (CSE)
with following features:

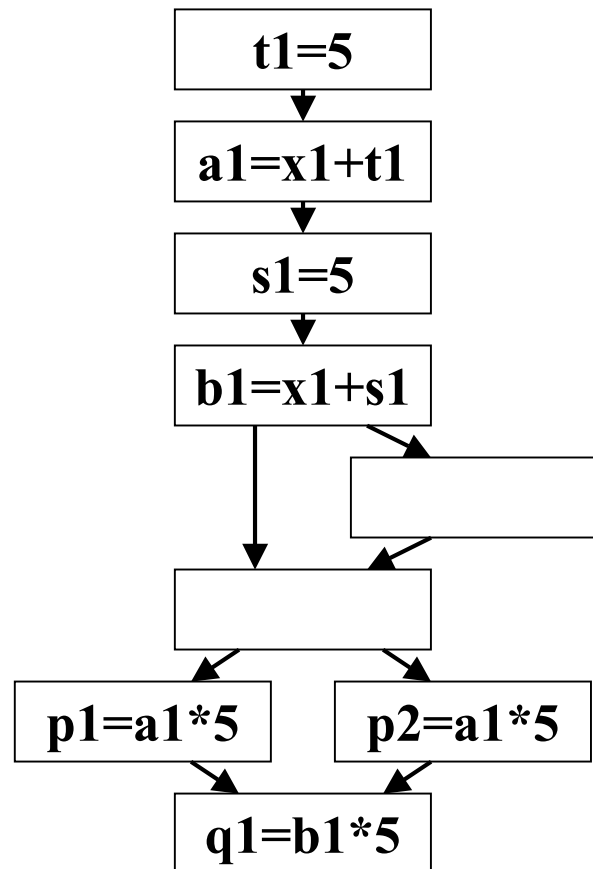
1. SSA-based approach, and
2. Efficiently propagating available expression information and copy assignment information along dominator tree.



Catching second order effects of CSE

Basic GVN

CFG

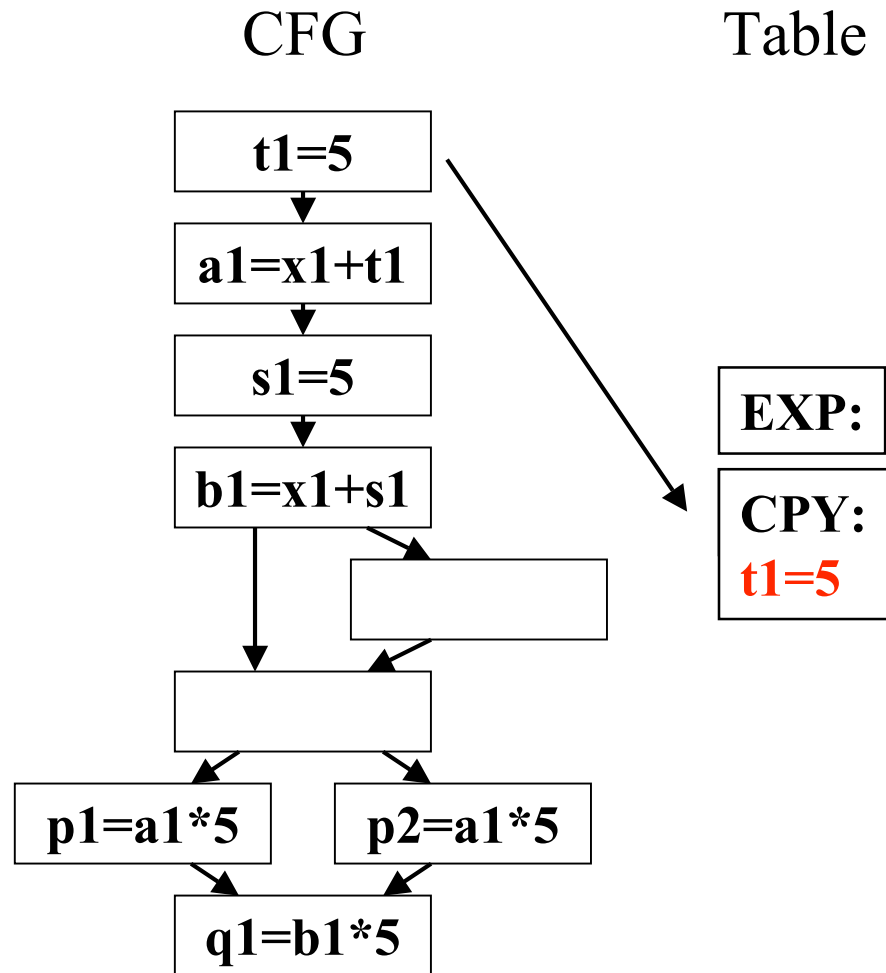


Table

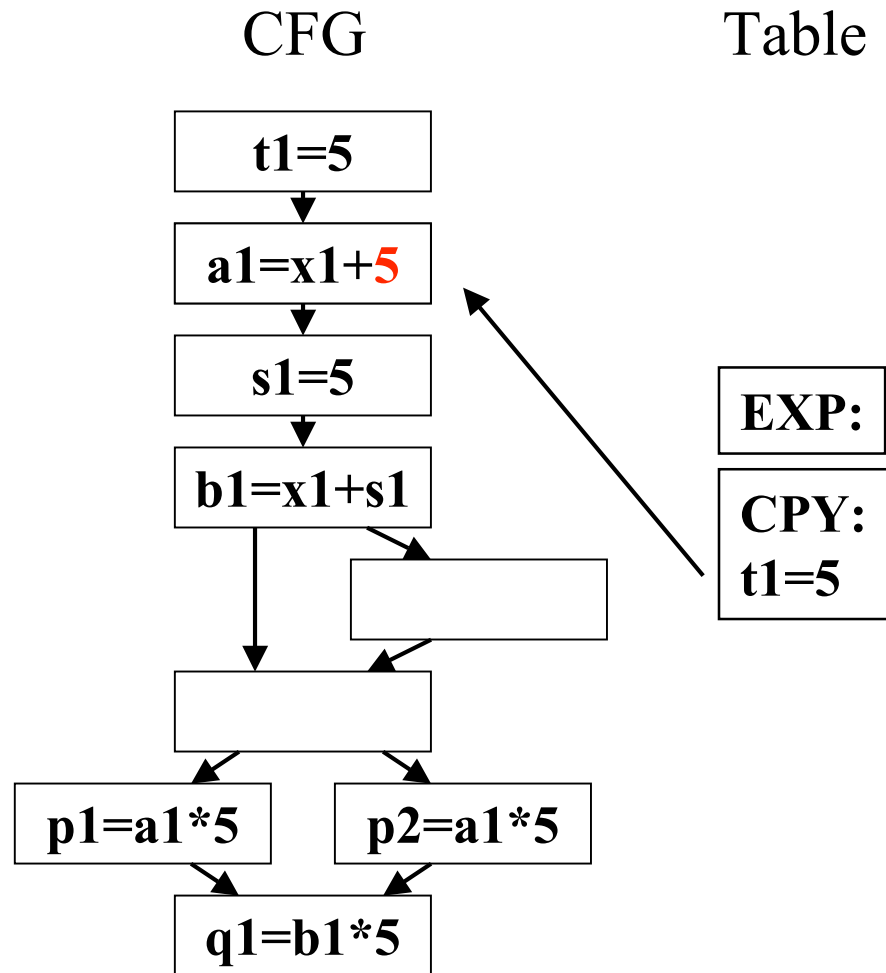
EXP:

CPY:

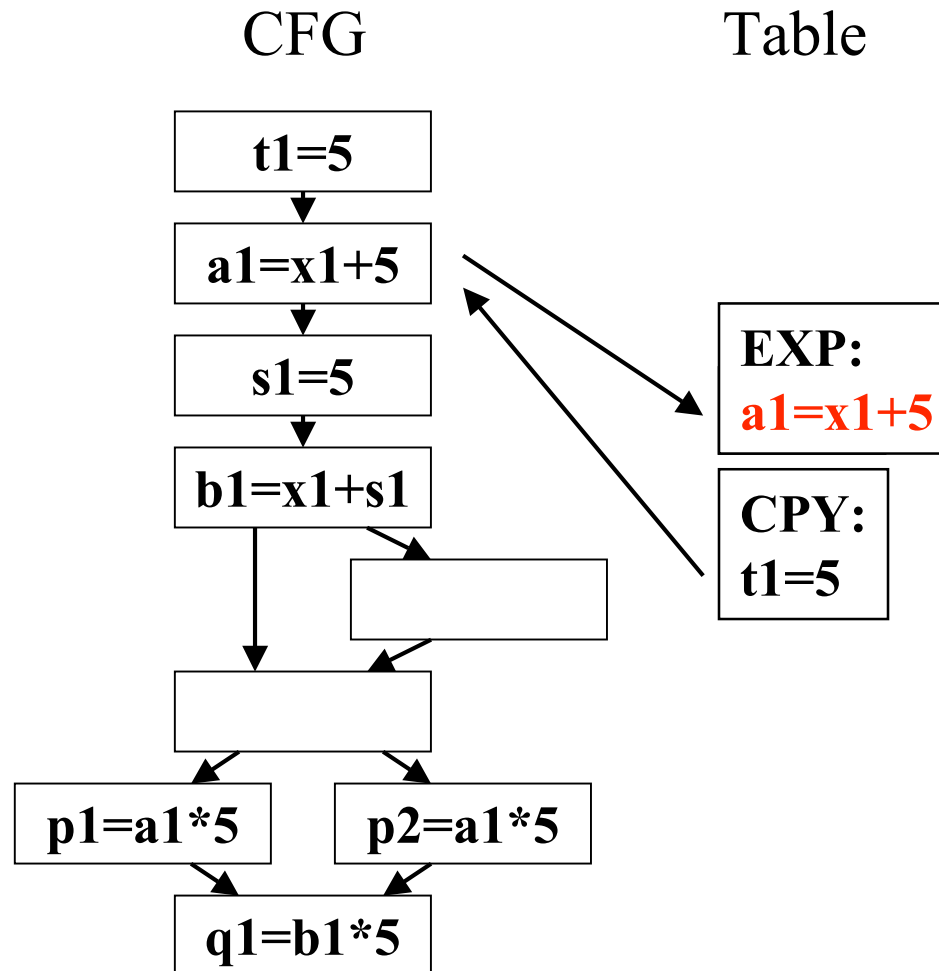
Basic GVN



Basic GVN

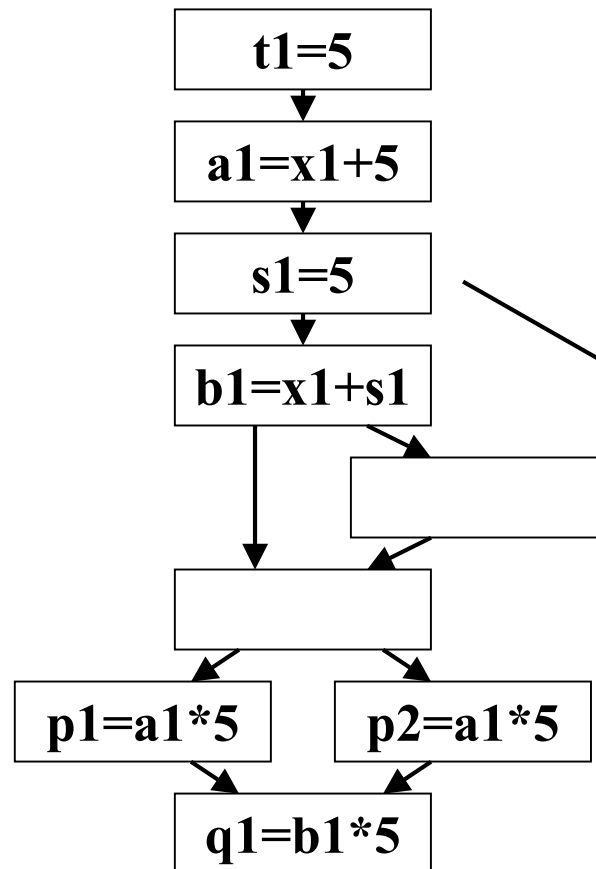


Basic GVN



Basic GVN

CFG

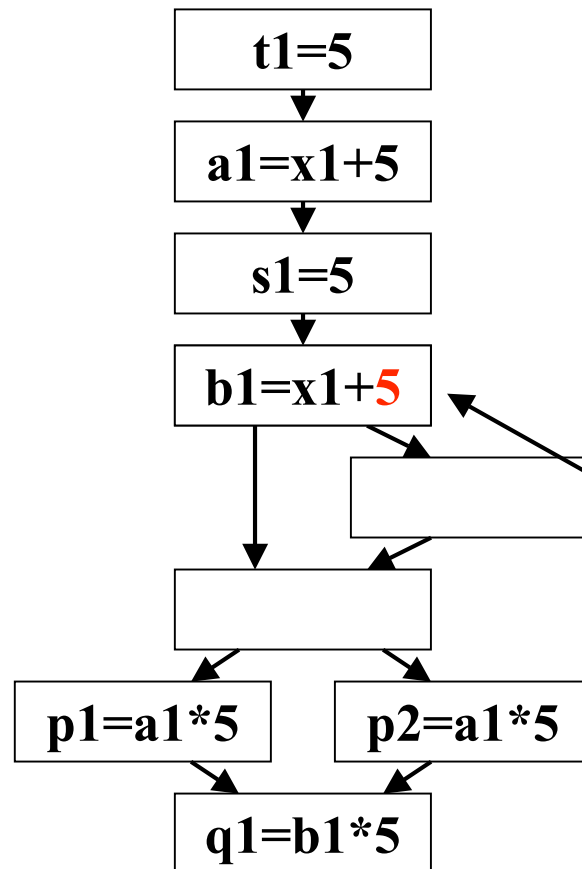


Table

EXP: $a1=x1+5$
CPY: $t1=5$ $s1=5$

Basic GVN

CFG

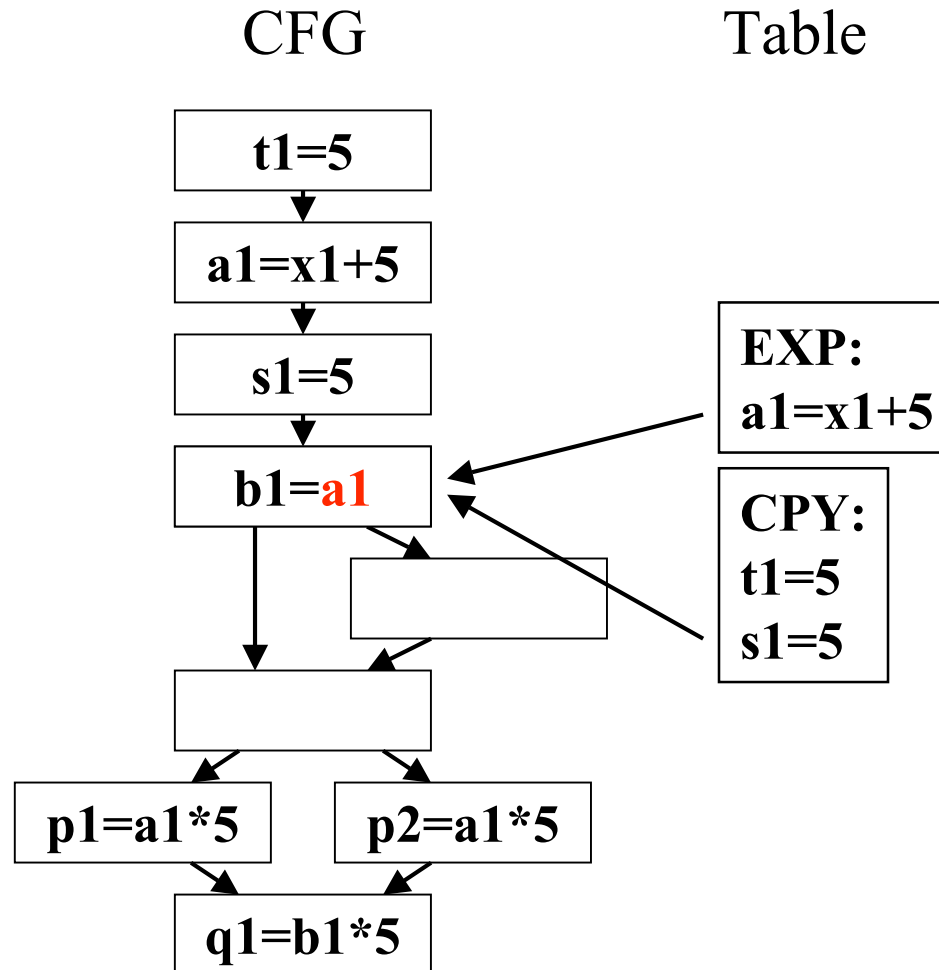


Table

EXP:
 $a1=x1+5$

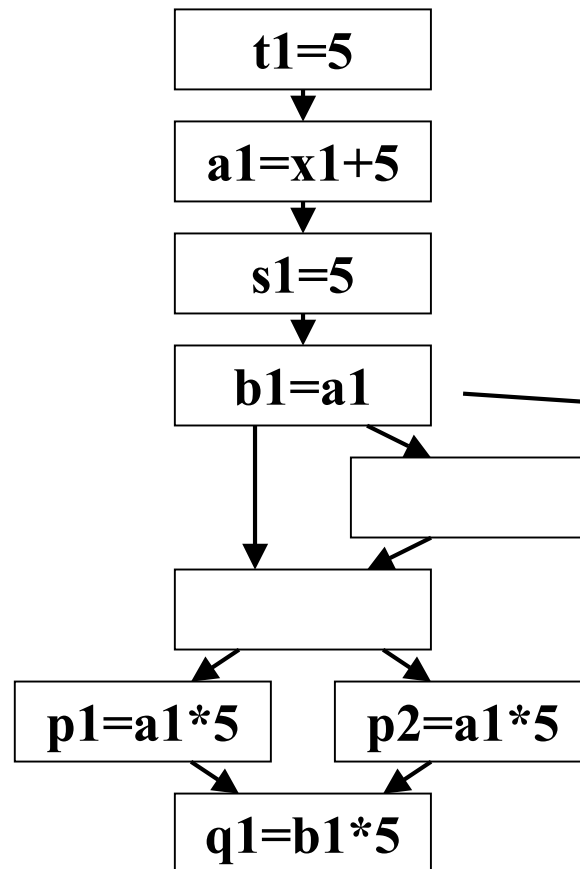
CPY:
 $t1=5$
 $s1=5$

Basic GVN



Basic GVN

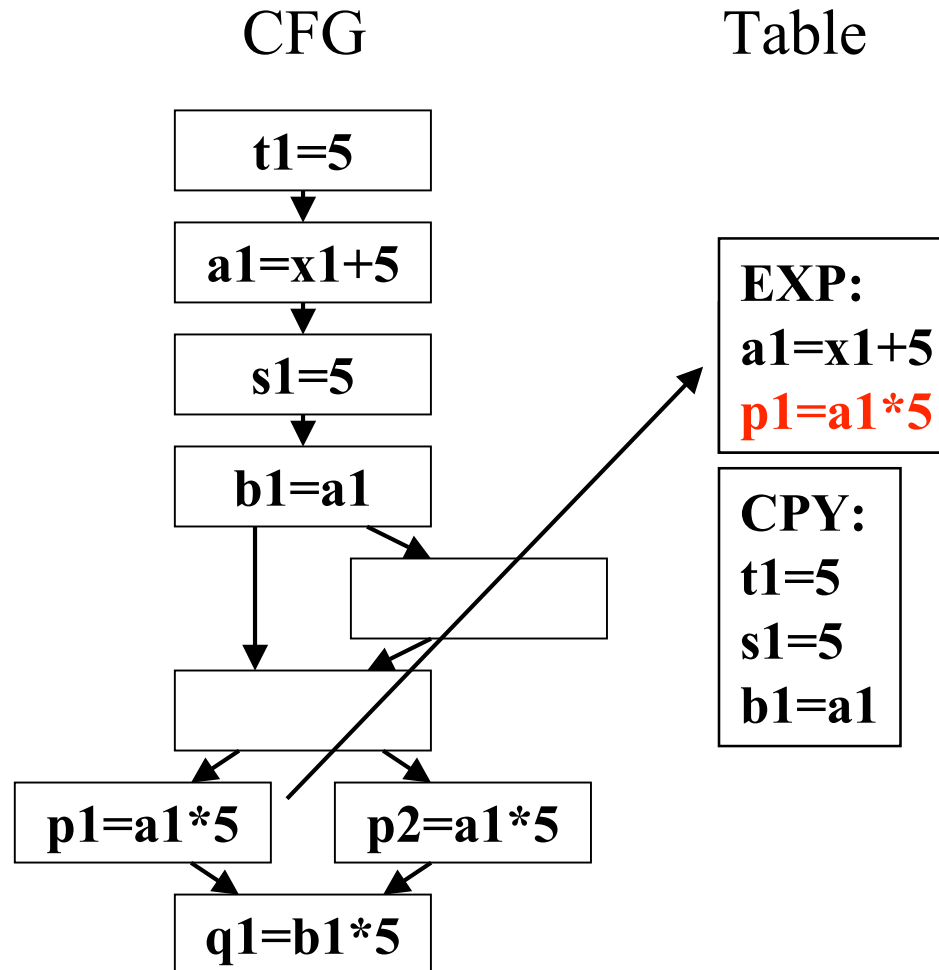
CFG



Table

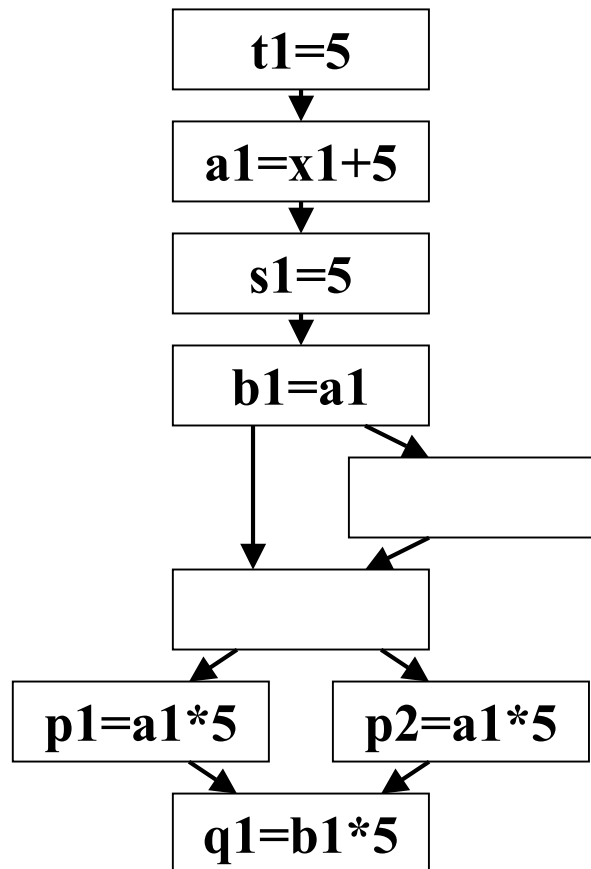
EXP: $a1=x1+5$
CPY: $t1=5$ $s1=5$ $b1=a1$

Basic GVN



Basic GVN

CFG



Table

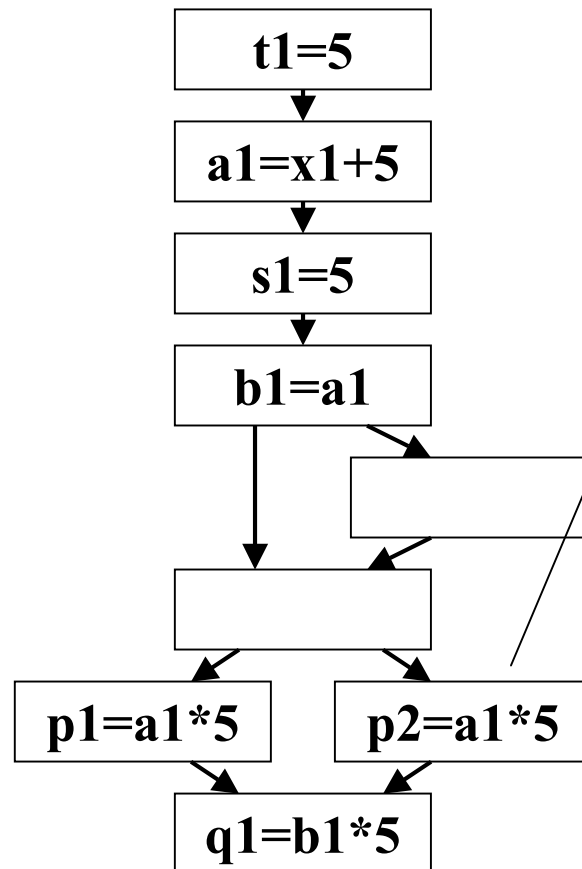
undo

EXP:
 $a1=x1+5$

CPY:
 $t1=5$
 $s1=5$
 $b1=a1$

Basic GVN

CFG



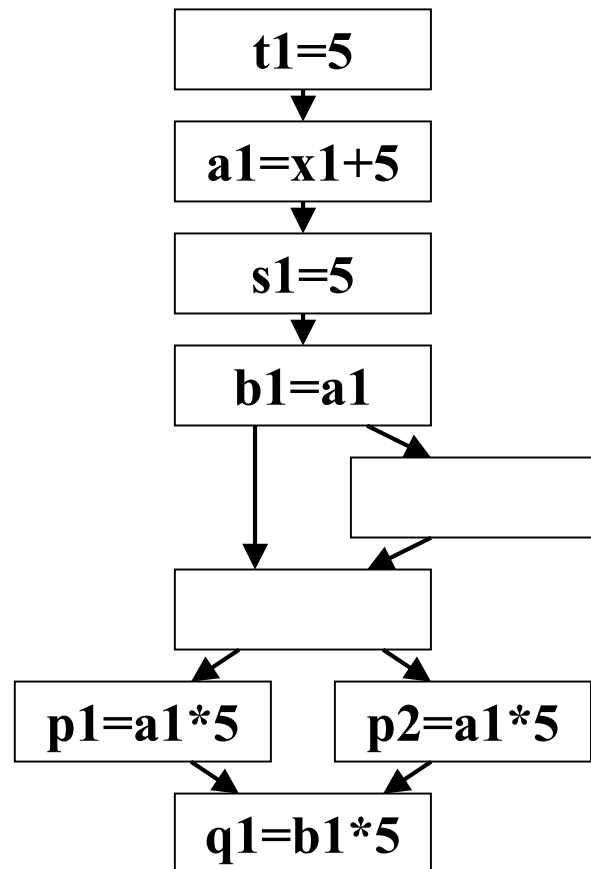
Table

EXP:
 $a1=x1+5$
 $p2=a1*5$

CPY:
 $t1=5$
 $s1=5$
 $b1=a1$

Basic GVN

CFG



Table

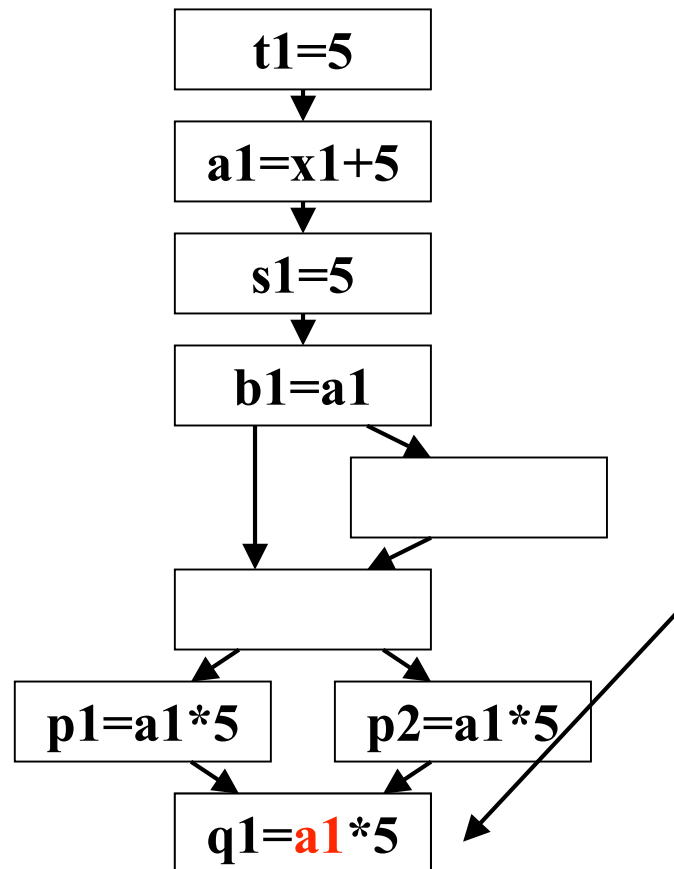
undo

EXP:
 $a1=x1+5$

CPY:
 $t1=5$
 $s1=5$
 $b1=a1$

Basic GVN

CFG

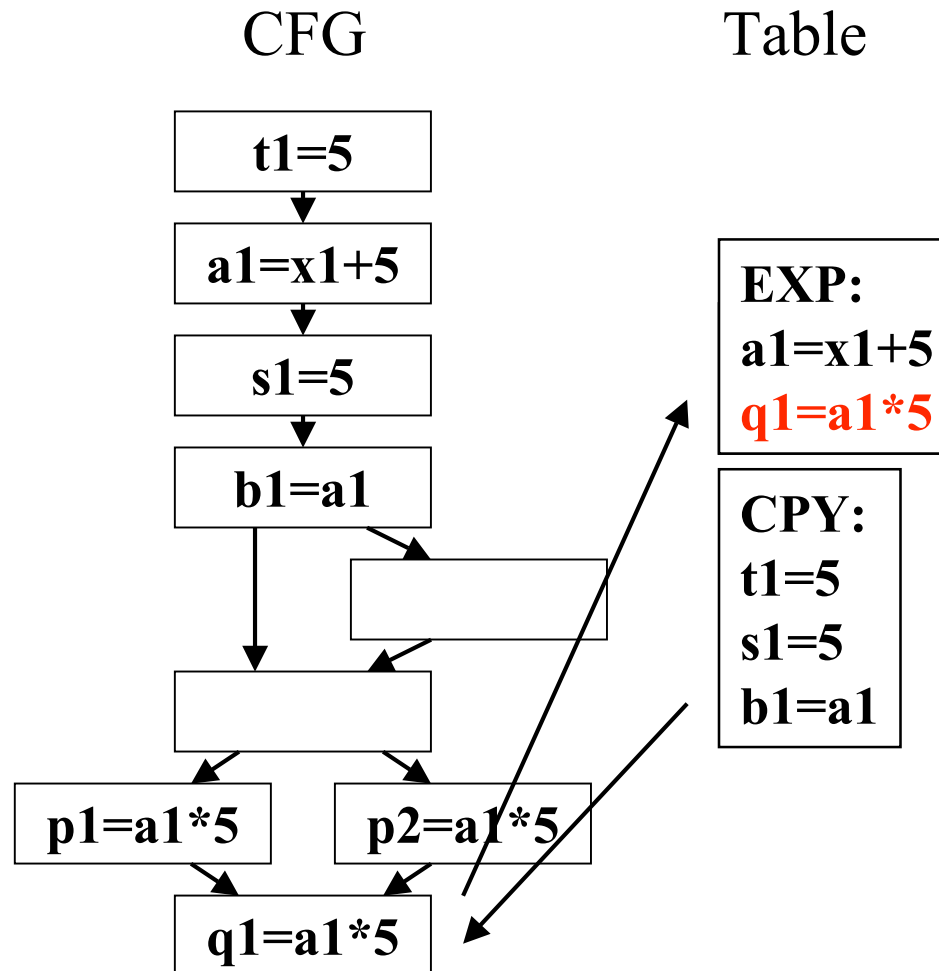


Table

EXP:
 $a1=x1+5$

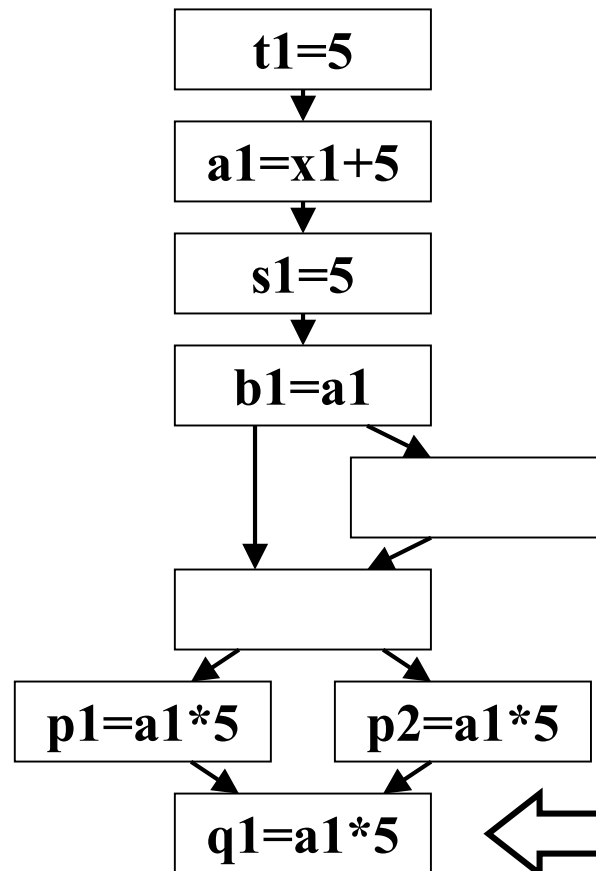
CPY:
 $t1=5$
 $s1=5$
 $b1=a1$

Basic GVN



Basic GVN

CFG



Table

EXP:
 $a1=x1+5$
 $q1=a1*5$

CPY:
 $t1=5$
 $s1=5$
 $b1=a1$

← missed CSE

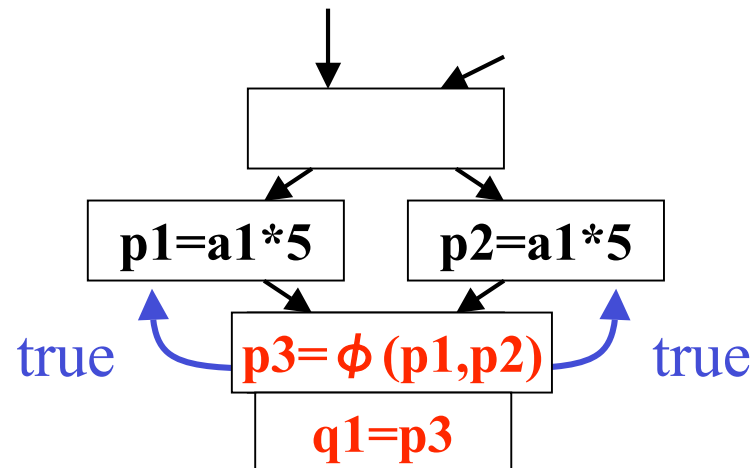
Extension Using Question Propagation (1)

Basic idea:

checking **whether expression e is redundant**
on demand

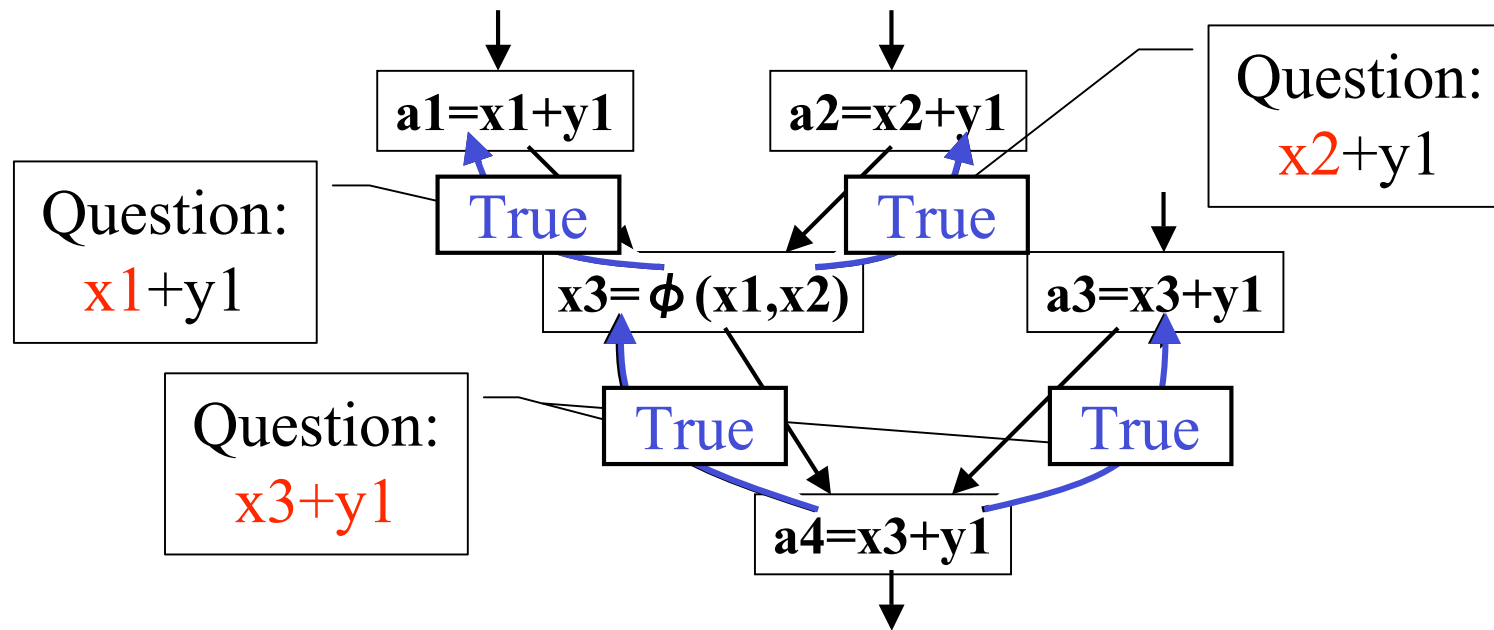


Backwardly propagating **the question**



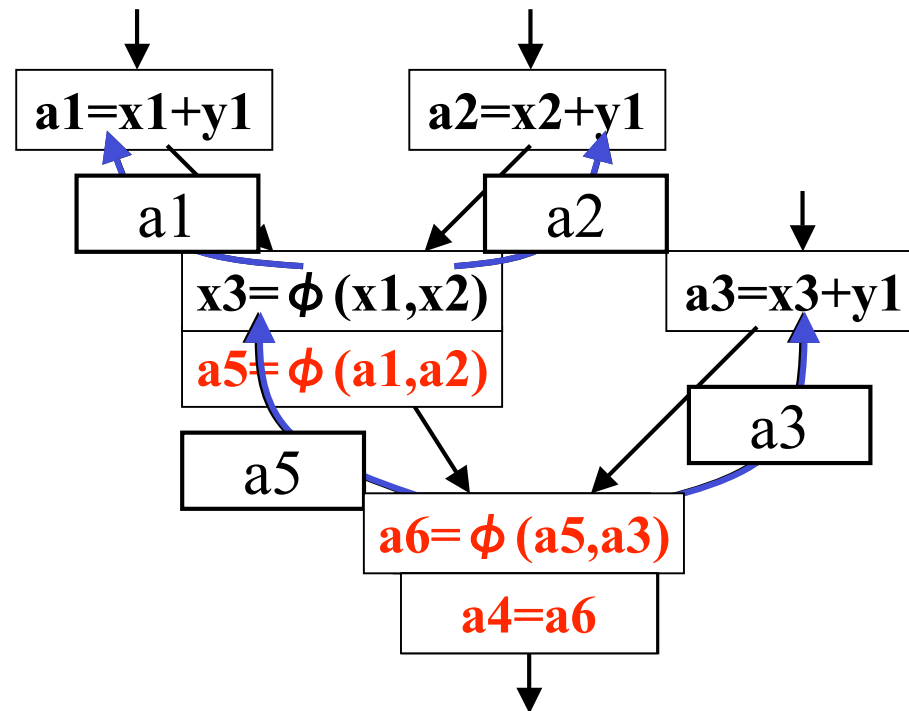
Question Propagation

- Propagating questions across ϕ -functions



Question Propagation

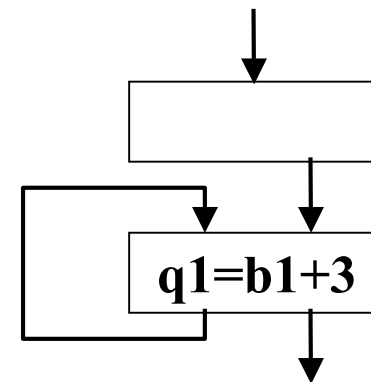
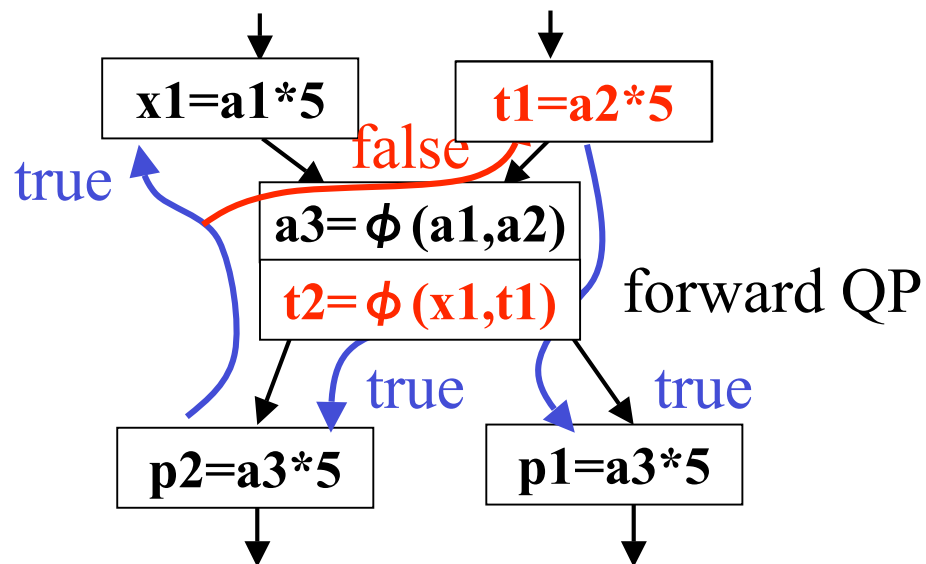
- Eliminating redundant expressions



Extension Using Question Propagation (2)

Furthermore, QP enables followings:

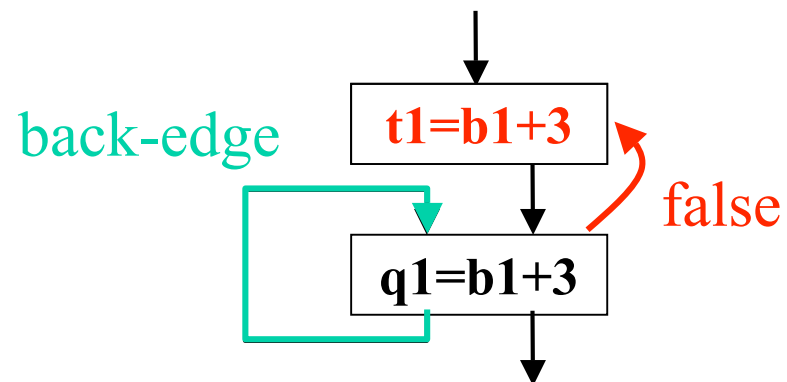
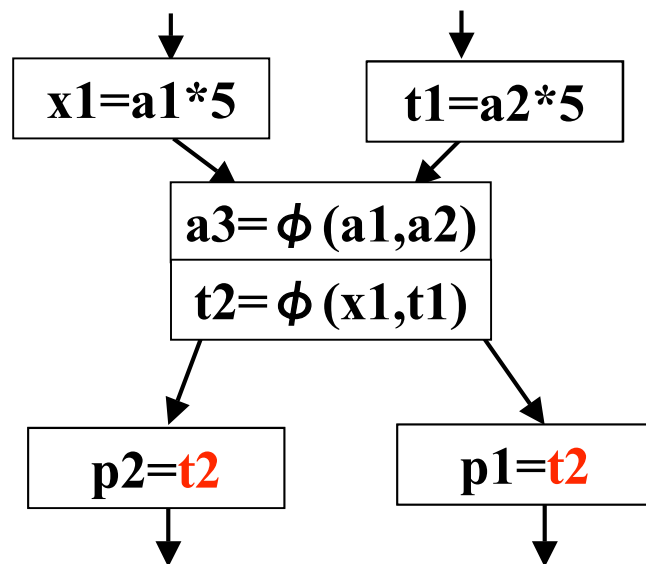
1. Eliminating partially redundant expressions, and
2. Hoisting loop-invariant expressions out of loops.



Extension Using Question Propagation (2)

Furthermore, QP enables followings:

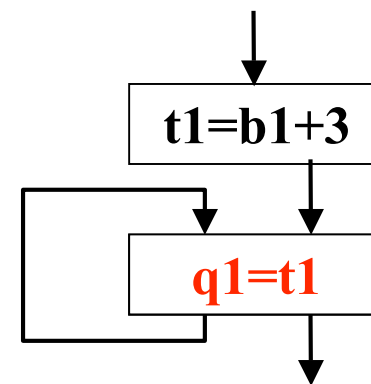
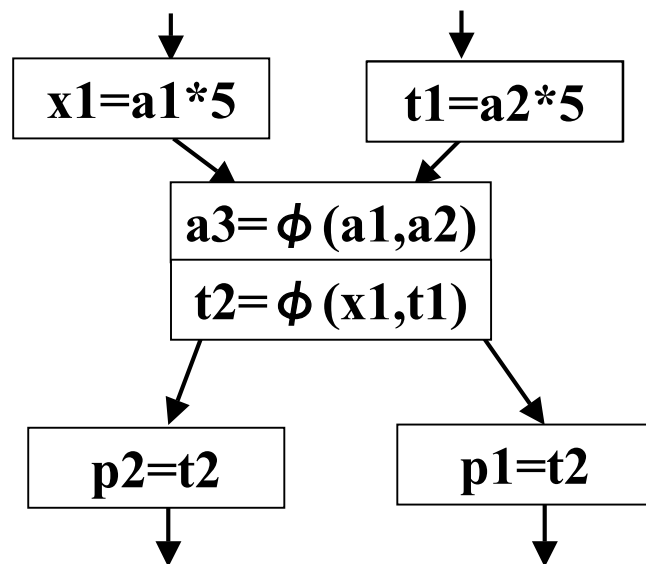
1. Eliminating partially redundant expressions, and
2. Hoisting loop-invariant expressions out of loops.



Extension Using Question Propagation (2)

Furthermore, QP enables followings:

1. Eliminating partially redundant expressions, and
2. Hoisting loop-invariant expressions out of loops.



Efficient Propagation

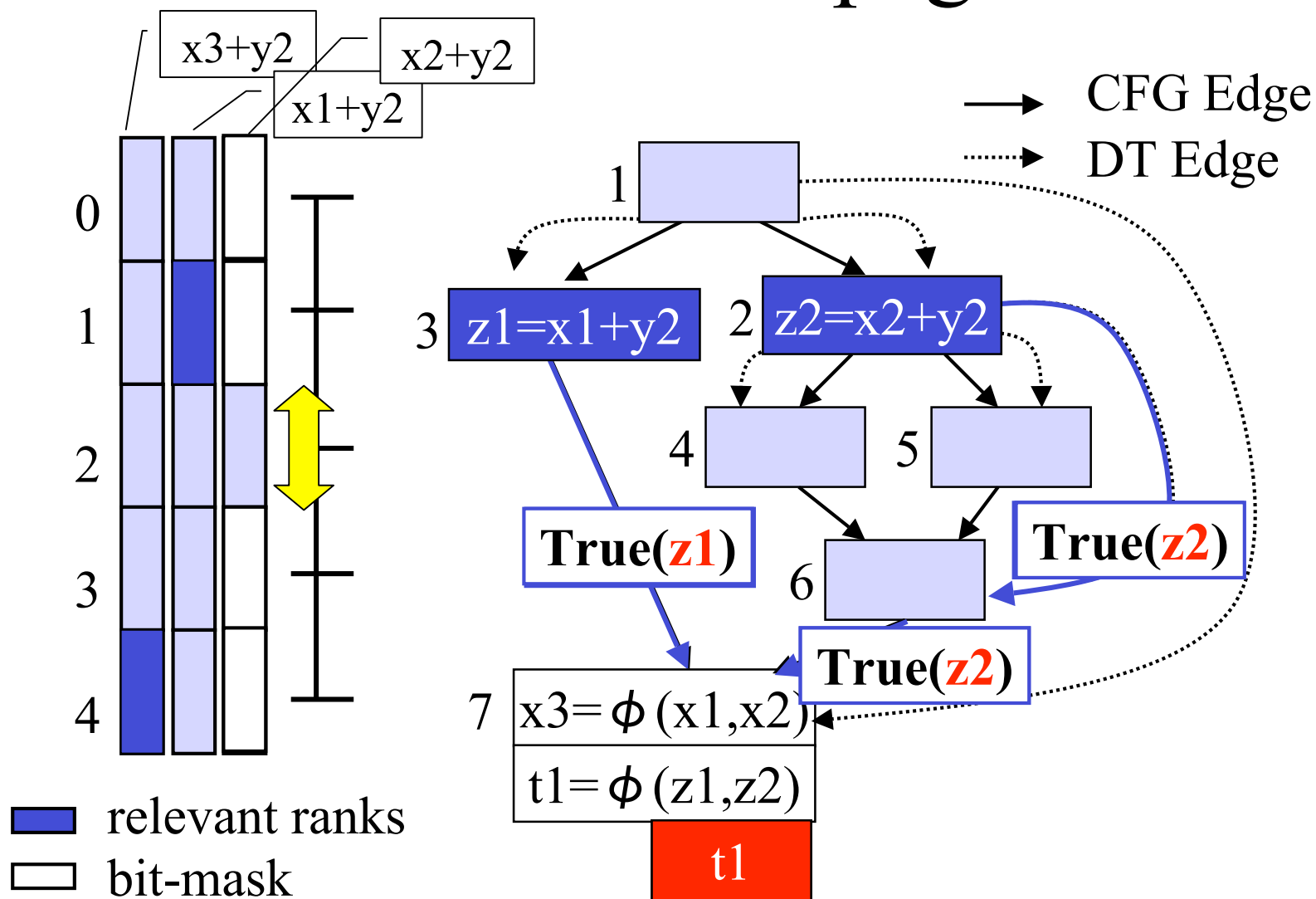
Short-circuit from current node n to its immediate dominator d if region R between n and d does not include any followings:

1. question's expression e
2. ϕ -function defining e 's operands, and
3. destination of up-edge except back-edge.



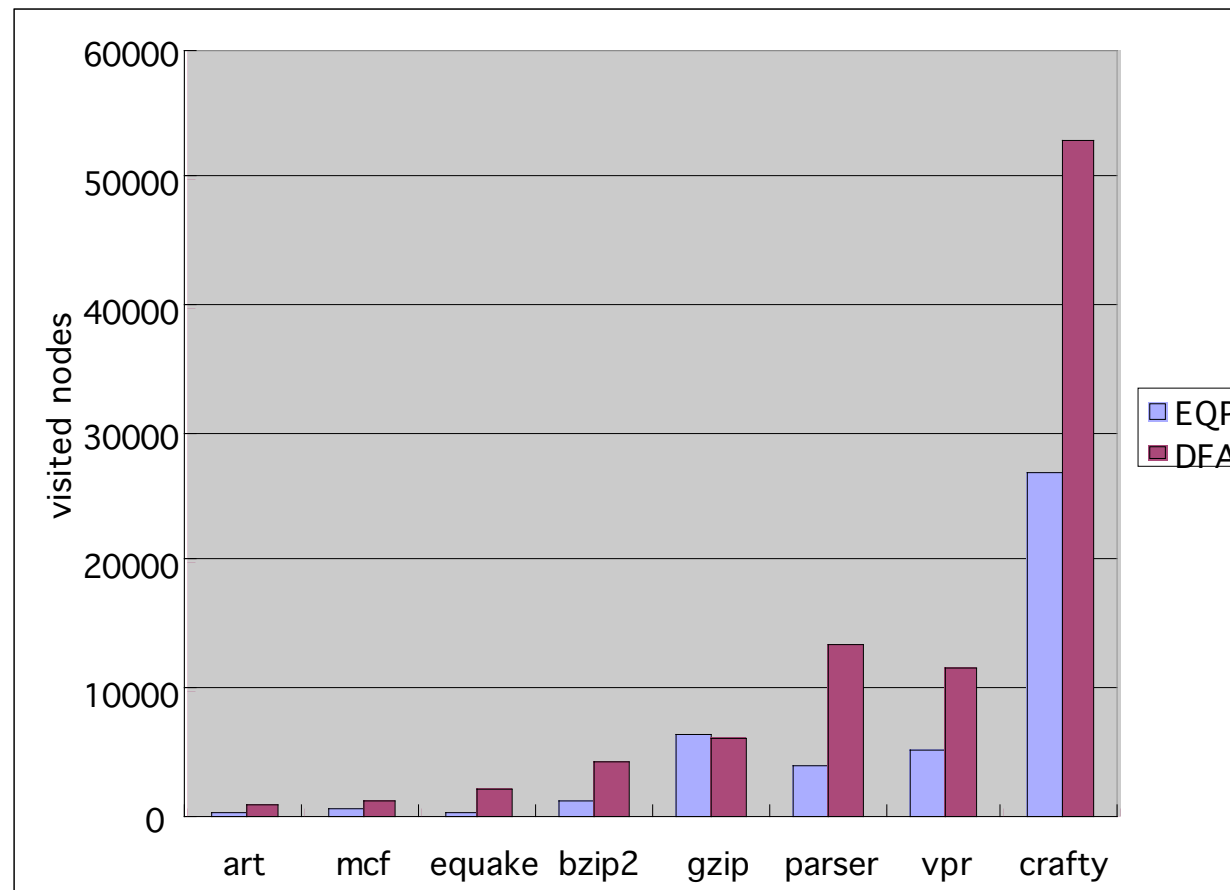
Efficiently checking using *rank* like depth

Efficient Propagation



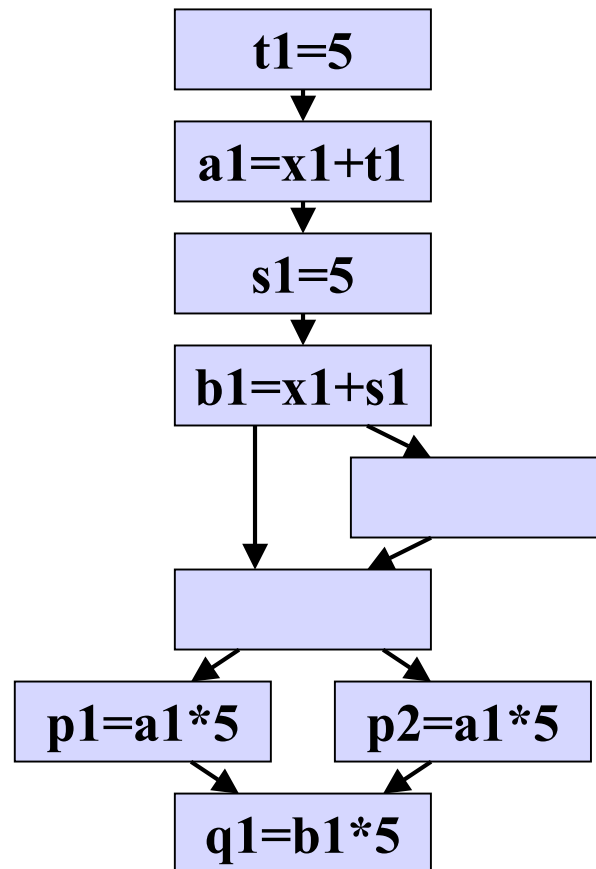
Comparing EQP with dataflow analysis

- Analyzing availabilities of all expressions



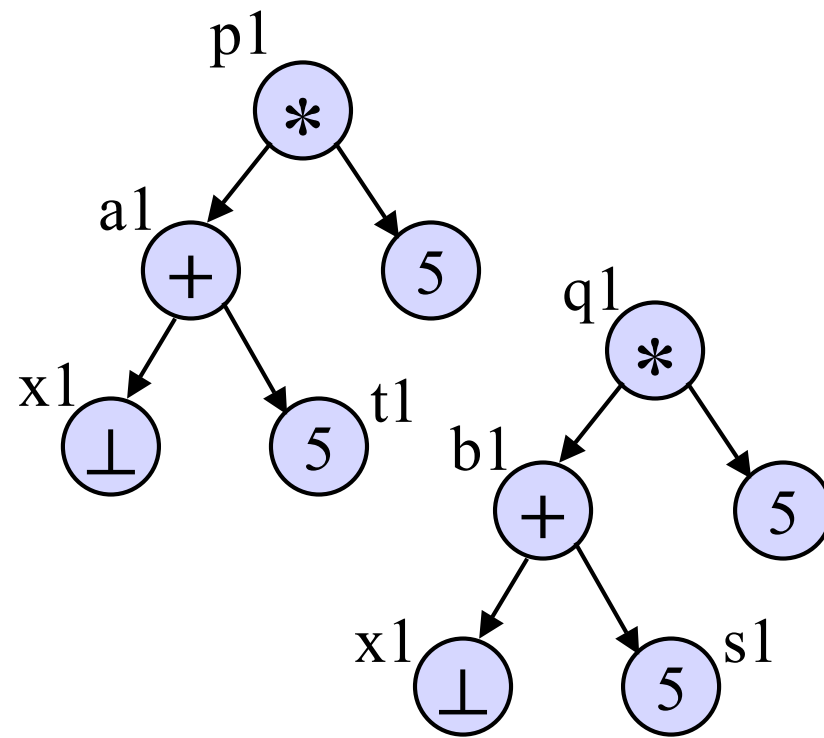
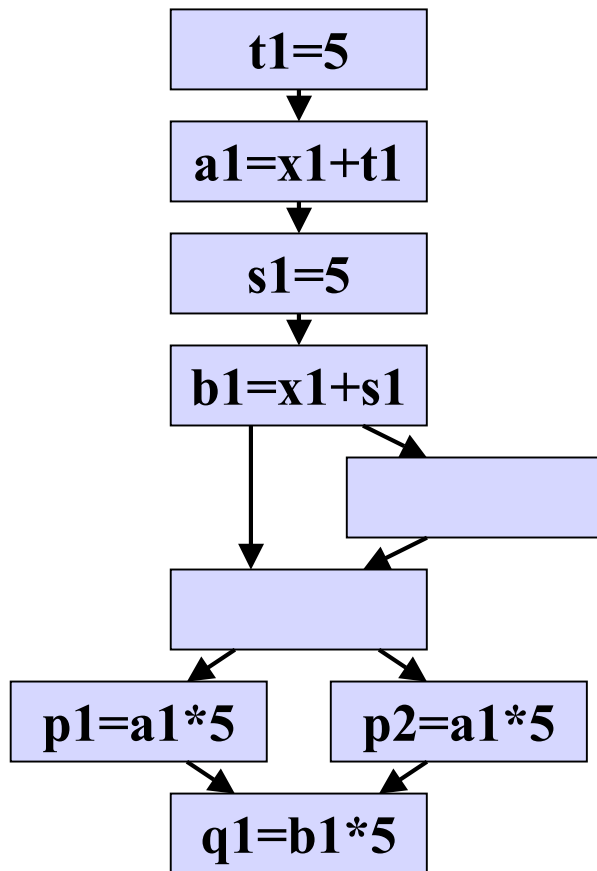
Previous Works

- Congruence among SSA graphs



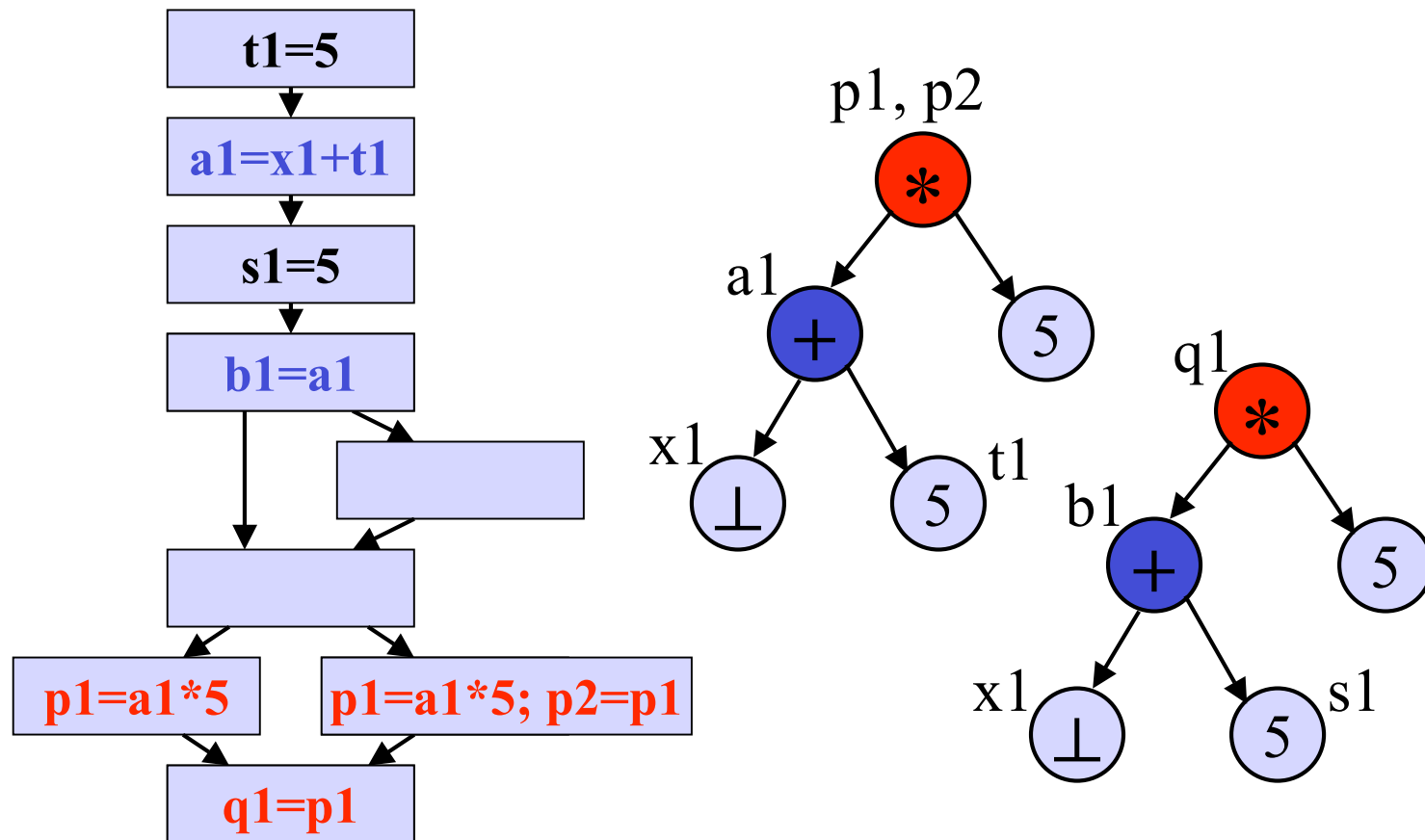
Previous Works

- Congruence among SSA graphs



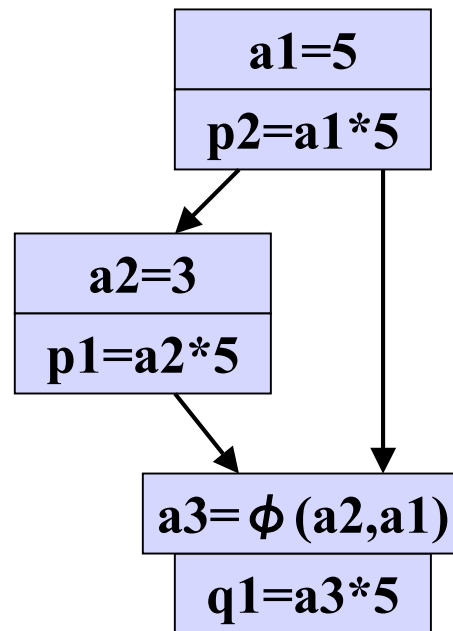
Previous Works

- Congruence among SSA graphs



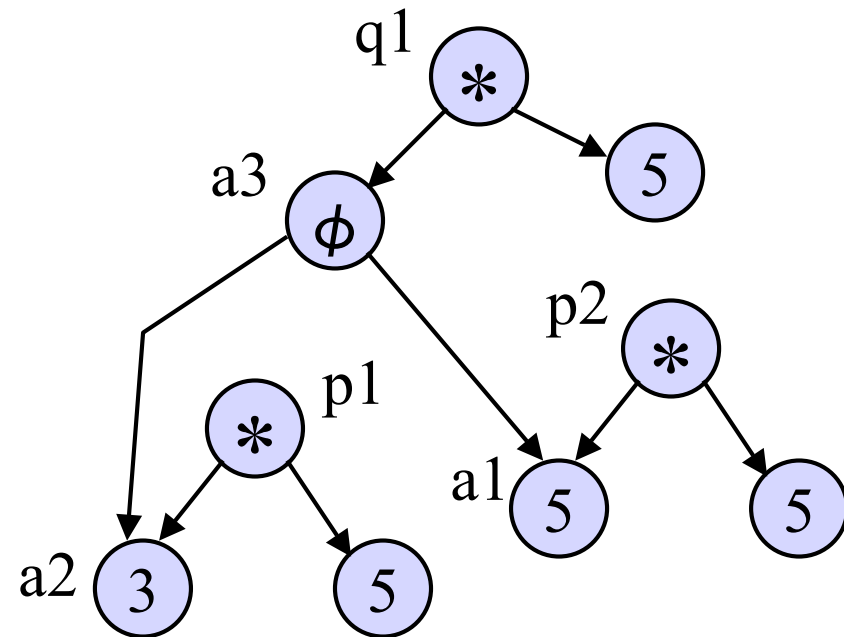
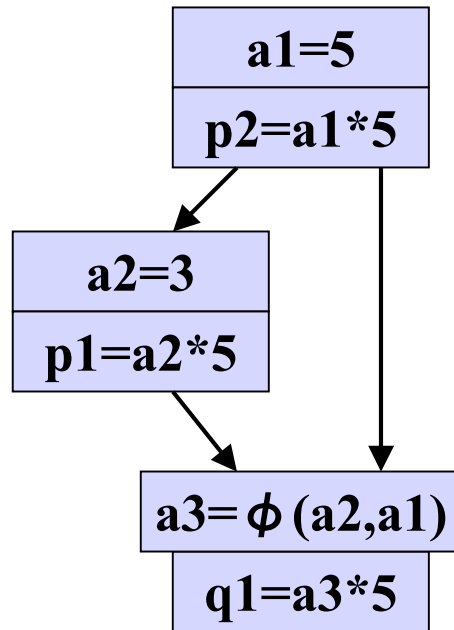
Previous Works

- Congruence among SSA graphs
missed optimization:



Previous Works

- Congruence among SSA graphs :
missed optimization:



Previous Works

- Congruence among SSA graphs :
missed optimization:

