Optimizing Compilers
7th Lecture

Bernhard Scholz
Institut f. Computersprachen
Argentinierstr. 8
scholz@complang.tuwien.ac.at

Outline

Machine-independent optimizations based on DFA

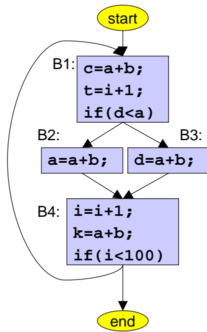
- Common Sub-Expression Elimination
- Copy Propagation

Common Sub-Expression Eliminiaton (CSE)

Goal:
Eliminate common sub-exressions

Approach:
1. Global Analysis: DFA for analyzing which common sub-expressions are available at a program point
2. Transformation: replace sub-expressions by first computation

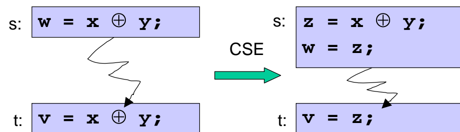
Example



- Common sub-expressions: a+b, i+1
- Which sub-expression compute the same value?

Approach

- Given a statement s: `w = x ⊕ y;` where `x ⊕ y` is available at s, the computation within s can be eliminated.
- Find statements of the form t: `v = x ⊕ y;` such that the path from s to t does not compute `x ⊕ y` or define `x` or `y`.



DFA Equations

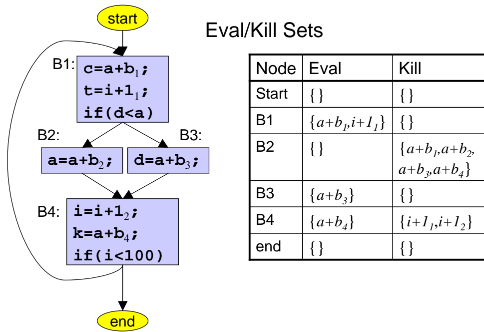
- Gen/Kill-Equations (i.e. forward & intersection)

  $AeOut(start) = \{\}$

  $AeIn(n) = \cap_{p \in preds(n)} AeOut(p)$

  $AeOut(n) = [In(n) - Kill(n)] \cup Eval(n)$
- DFA sets
  - *AeIn*(*n*): set of available expressions at entry
  - *AeOut*(*n*): set of available expressions at exit
- Local sets
  - *Eval*(*n*): set of expressions evaluated in block *n*
  - *Kill*(*n*): set of expressions killed in block *n*

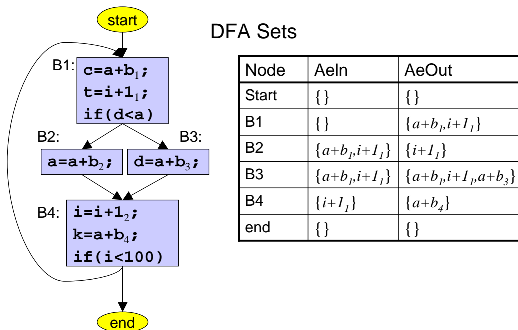
Example (cont'd)



Eval/Kill Sets

Node	Eval	Kill
Start	{ }	{ }
B1	$\{a+b_1, i+1_1\}$	{ }
B2	{ }	$\{a+b_1, a+b_2, a+b_3, a+b_4\}$
B3	$\{a+b_3\}$	{ }
B4	$\{a+b_4\}$	$\{i+1_1, i+1_2\}$
end	{ }	{ }

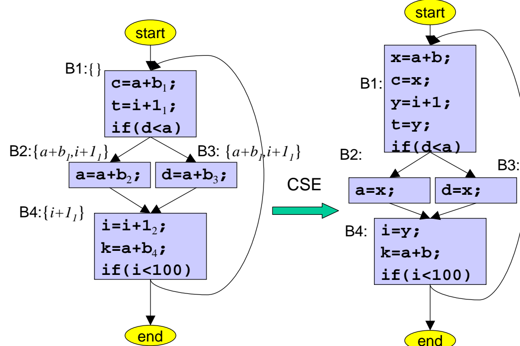
Example (cont'd)



DFA Sets

Node	AeIn	AeOut
Start	{ }	{ }
B1	{ }	$\{a+b_1, i+1_1\}$
B2	$\{a+b_1, i+1_1\}$	$\{i+1_1\}$
B3	$\{a+b_1, i+1_1\}$	$\{a+b_1, i+1_1, a+b_3\}$
B4	$\{i+1_1\}$	$\{a+b_4\}$
end	{ }	{ }

Example (cont'd)

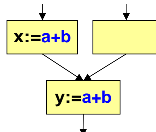


Implementation

- Every computation is potentially reused
  - consider computations which occur at least twice for keeping compile-time small.
- Watch out!
  - Two instances of computations in a function might compute different things
- Passes:
  1. Compute Eval/Kill sets (consider precedence)
  2. Run gen/kill solver (forward & intersection problem)
  3. Perform transformation (scan through basic block)
     - at least one instance must reach another instance
- Iterative approach to eliminate all CSE!

Partial Redundancy Elimination

- Combines global common sub-expression elimination and loop-invariant code motion.
- *Partial Redundancy* is a computation that is done more than once on some incoming path.
- Example:

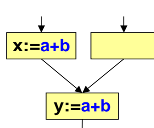


- PRE: inserts and deletes computation for minimizing partial redundancies!

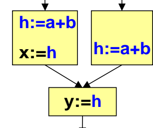
Code Motion Transformation

- Transformation
  - introduce a new temporary variable for computation
  - delete redundant computations
  - insert computations to preserve program semantics
- Example

Input Program:



Optimised Program:



Classical PRE

- Uni-directional approaches
  - Busy Code Motion / Lazy Code Motion (J. Knoop, O. Rüthing, B. Steffen)
  - Problems: critical edges
- Bi-directional approaches
  - Morel and Renvoise
  - Problems: convergence
- Importance
  - Many optimisations based on PRE
    - load/store optimisation,
    - communication optimisation, etc.

Copy Propagation(CP)

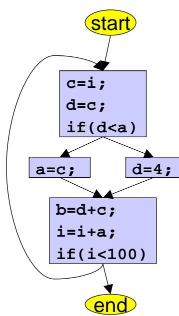
Goal:

Propagate values of copy assignments

Approach:

1. Global Analysis: DFA for analyzing whether propagation is valid.
2. Transformation: replace copy value

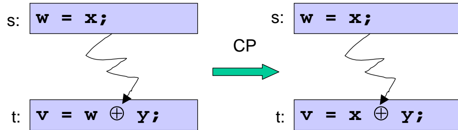
Example



- Copy-Assignments:
  c=i; a=c; d=c;
- Which copy values can be replaced?
- Which copy assignment can be eliminated?

Approach

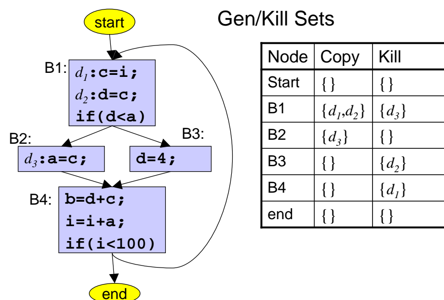
- Given a copy assignment s: `w = x;`
- Given another statement t: `v = w ⊕ y;`
- If s reaches t and no other definition of w reaches t and there is no definition of x on any path from s to t then:



DFA Equations

- Gen/Kill-Equations (i.e. forward & intersection)
  - $CoOut(start) = \{\}$
  - $CoIn(n) = \cap_{p \in preds(n)} CoOut(p)$
  - $CoOut(n) = [CoIn(n) - Kill(n)] \cup Copy(n)$
- DFA sets
  - $CoIn(n)$: set of copies at entry of n
  - $CoOut(n)$: set of copies at exit of n
- Local sets
  - $Copy(n)$: set of copies generated in n
  - $Kill(n)$: set of copies killed in n

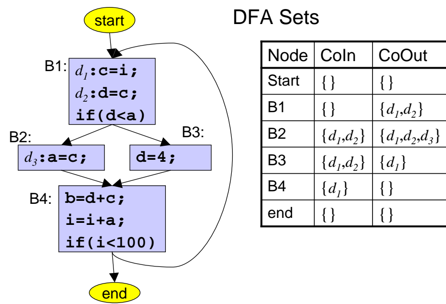
Example (cont'd)



Gen/Kill Sets

Node	Copy	Kill
Start	$\{\}$	$\{\}$
B1	$\{d_1,d_2\}$	$\{d_3\}$
B2	$\{d_3\}$	$\{\}$
B3	$\{\}$	$\{d_2\}$
B4	$\{\}$	$\{d_1\}$
end	$\{\}$	$\{\}$

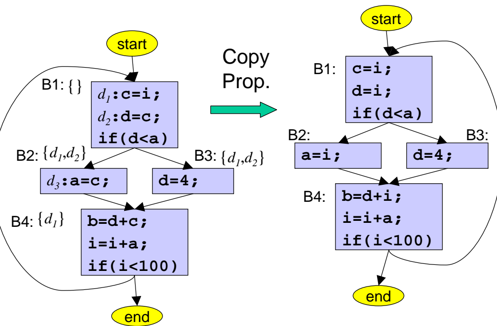
Example (cont'd)



DFA Sets

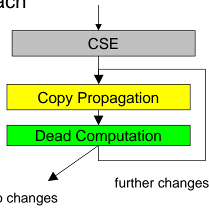
Node	CoIn	CoOut
Start	{ }	{ }
B1	{ }	$\{d_1,d_2\}$
B2	$\{d_1,d_2\}$	$\{d_1,d_2,d_3\}$
B3	$\{d_1,d_2\}$	$\{d_1\}$
B4	$\{d_1\}$	{ }
end	{ }	{ }

Example (cont'd)



Optimisation Approach

- Approach



- To capture all copies =>
  iterative approach with dead computations
- Iterate as long as everything is stable

Stop

- Next lecture: 5.6.2003, 13:45 – 14:45

This document was created with Win2PDF available at http://www.daneprairie.com.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.