CS738: Advanced Compiler Optimizations

Constant Propagation

Amey Karkare
karkare@cse.iitk.ac.in

http://www.cse.iitk.ac.in/~karkare/cs738
Department of CSE, IIT Kanpur

Agenda

- Using data flow analysis to identify "constant expressions" in a program
- Identify similarity/differences with bit-vector data flow analyses discussed earlier
- Other properties of constant propagation

Constant Propagation

- CP: Replace expressions that evaluate to same constant "c" every time they are executed, by the value "c"

DF Framework for CP

- Domain

DF Framework for CP

- Domain
  - For a single variable v of type τ, all possible constants of type τ

DF Framework for CP

- Domain
  - For a single variable *v* of type τ, all possible constants of type τ
- Semilattice

DF Framework for CP

- Domain
  - For a single variable *v* of type τ, all possible constants of type τ
- Semilattice
  - What is $\bigwedge$?

DF Framework for CP

- Domain
  - For a single variable *v* of type τ, all possible constants of type τ
- Semilattice
  - What is $\bigwedge$?
  - What is $\top$?

DF Framework for CP

- Domain
  - For a single variable *v* of type τ, all possible constants of type τ
- Semilattice
  - What is $\bigwedge$?
  - What is $\top$?
  - What is $\bot$?

Special Values for CP

- *NAC*: not a constant

Special Values for CP

- *NAC*: not a constant
  - If variable is inferred not to be a constant

Special Values for CP

- *NAC*: not a constant
  - If variable is inferred not to be a constant
  - Multiple (different valued) defs, non-const defs, assigned an "un-interpreted" value, . . .

Special Values for CP

- *NAC*: not a constant
  - If variable is inferred not to be a constant
  - Multiple (different valued) defs, non-const defs, assigned an “un-interpreted” value, . . .
- *Undef*: No definition of the variable is seen yet - nothing known!

NAC vs *Undef*

- *NAC* $\Rightarrow$ *too many* definitions seen for a variable v to declare v is **NOT** a constant

NAC vs *Undef*

- *NAC* $\Rightarrow$ *too many* definitions seen for a variable *v* to declare *v* is **NOT** a constant
- *Undef* $\Rightarrow$ *too few* definitions seen to declare anything about the variable

NAC vs *Undef*

- *NAC* $\Rightarrow$ *too many* definitions seen for a variable v to declare v is **NOT** a constant
- *Undef* $\Rightarrow$ *too few* definitions seen to declare anything about the variable
- $\top$ is *Undef*; $\perp$ is *NAC*

CP Meet $\bigwedge$

- Recall the requirement

$$\top \bigwedge x = x$$

$$\bot \bigwedge x = \bot$$

CP Meet $\bigwedge$

- Recall the requirement

$$\top \bigwedge x = x$$

$$\bot \bigwedge x = \bot$$

$$Undef \bigwedge c = c$$

CP Meet $\bigwedge$

- Recall the requirement

$$\top \bigwedge x = x$$

$$\bot \bigwedge x = \bot$$

$$\begin{aligned}
&Undef \bigwedge c = c \\
&NAC \bigwedge c = NAC
\end{aligned}$$

CP Meet $\bigwedge$

- Recall the requirement

$$\top \bigwedge x = x$$

$$\bot \bigwedge x = \bot$$

$$Undef \bigwedge c = c$$
$$NAC \bigwedge c = NAC$$
$$c_1 \bigwedge c_2 = NAC \text{ when } c_1 \neq c_2$$

CP Meet $\bigwedge$

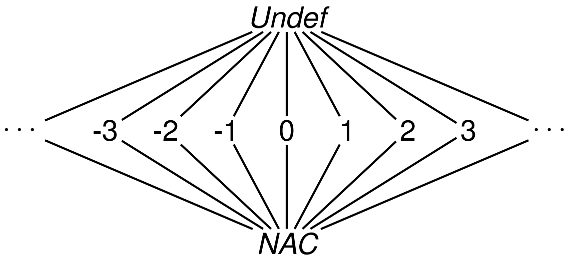
- Recall the requirement

$$\top \bigwedge x = x$$

$$\perp \bigwedge x = \perp$$

$$\begin{aligned}
&Undef \bigwedge c = c \\
&NAC \bigwedge c = NAC \\
&c_1 \bigwedge c_2 = NAC \text{ when } c_1 \neq c_2 \\
&c \bigwedge c = c
\end{aligned}$$

CP Semilattice for an integer variable



- Infinite domain, but finite height

CP Semilattice

- Previous figure was semilattice for one variable of one type

CP Semilattice

- Previous figure was semilattice for one variable of one type
- CP Semilattice = Product of such lattices for all variables (of all types)

CP Semilattice

- Previous figure was semilattice for one variable of one type
- CP Semilattice = Product of such lattices for all variables (of all types)
- Each semilattice has a **finite** height

Computing GEN

- Informal representation

Statement	GEN
$x = c$ // const	
$x = y + z$	
$x =$ *complicated expr*	

Computing GEN

- Informal representation

Statement	GEN
$x = c$ // const	$\{x \to c\}$
$x = y + z$	
$x =$ *complicated expr*	

Computing GEN

- Informal representation

Statement	GEN
$x = c$ // const	$\{x \to c\}$
$x = y + z$	if $\{y \to c_1, z \to c_2\}$ in IN then $\{x \to c_1 + c_2\}$
$x = \mathit{complicated\ expr}$	

Computing GEN

- Informal representation

Statement	GEN
$x = c$ // const	$\{x \to c\}$
$x = y + z$	if $\{y \to c_1, z \to c_2\}$ in IN then $\{x \to c_1 + c_2\}$ else if $\{y \to NAC\}$ in IN then $\{x \to NAC\}$
$x = complicated$ $expr$	

Computing GEN

- Informal representation

Statement	GEN
$x = c$ // const	$\{x \to c\}$
$x = y + z$	if $\{y \to c_1, z \to c_2\}$ in IN then $\{x \to c_1 + c_2\}$ else if $\{y \to NAC\}$ in IN then $\{x \to NAC\}$ else if $\{z \to NAC\}$ in IN then $\{x \to NAC\}$
$x = \mathit{complicated\ expr}$	

Computing GEN

- Informal representation

Statement	GEN
$x = c$ // const	$\{x \rightarrow c\}$
$x = y + z$	if $\{y \rightarrow c_1, z \rightarrow c_2\}$ in IN then $\{x \rightarrow c_1 + c_2\}$ else if $\{y \rightarrow NAC\}$ in IN then $\{x \rightarrow NAC\}$ else if $\{z \rightarrow NAC\}$ in IN then $\{x \rightarrow NAC\}$ else $\{x \rightarrow Undef\}$
$x = complicated$ $expr$	

Computing GEN

- Informal representation

Statement	GEN
$x = c$ // const	$\{x \rightarrow c\}$
$x = y + z$	if $\{y \rightarrow c_1, z \rightarrow c_2\}$ in IN then $\{x \rightarrow c_1 + c_2\}$ else if $\{y \rightarrow NAC\}$ in IN then $\{x \rightarrow NAC\}$ else if $\{z \rightarrow NAC\}$ in IN then $\{x \rightarrow NAC\}$ else $\{x \rightarrow Undef\}$
$x = complicated$ $expr$	$\{x \rightarrow NAC\}$

Monotonicity of CP

- Case analysis on transfer function f

Monotonicity of CP

- Case analysis on transfer function f
- $NAC \leq c \leq Undef$

Monotonicity of CP

- Case analysis on transfer function f
- $NAC \leq c \leq Undef$
- $x = c$ has constant transfer function.

Monotonicity of CP

- Case analysis on transfer function f
- $NAC \leq c \leq Undef$
- $x = c$ has constant transfer function.
- $x = complicated\ expr$ also has constant transfer function

Monotonicity of CP

- Case analysis on transfer function f
- $NAC \leq c \leq Undef$
- $x = c$ has constant transfer function.
- $x =$ *complicated expr* also has constant transfer function
- See the next slide for $x = y + z$ (and similar statements)

Monotonicity of CP: $x = y + z$

- Fix z to be one of *Undef*, c_2, *NAC*

Monotonicity of CP: $x = y + z$

- Fix z to be one of *Undef*, c_2, *NAC*
- Vary y over *Undef*, c_1, *NAC*

Monotonicity of CP: $x = y + z$

- Fix z to be one of *Undef*, c_2, *NAC*
- Vary y over *Undef*, c_1, *NAC*
- Confirm that x does not "*increase*"

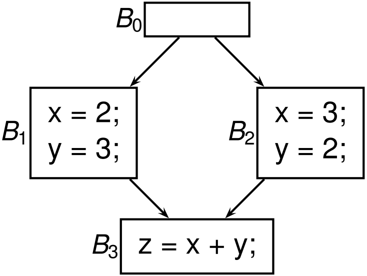
Monotonicity of CP: $x = y + z$

- Fix z to be one of *Undef*, c_2, *NAC*
- Vary y over *Undef*, c_1, *NAC*
- Confirm that x does not "*increase*"
- Do this for all z choices.

Monotonicity of CP: $x = y + z$

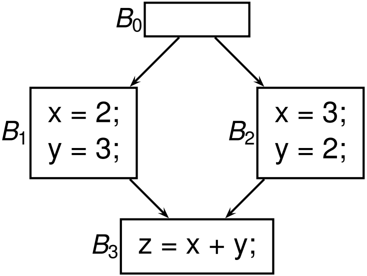
- Fix z to be one of *Undef*, c_2, *NAC*
- Vary y over *Undef*, c_1, *NAC*
- Confirm that x does not "*increase*"
- Do this for all z choices.
- Similarly, fix y and vary z.

Nondistributivity of CP



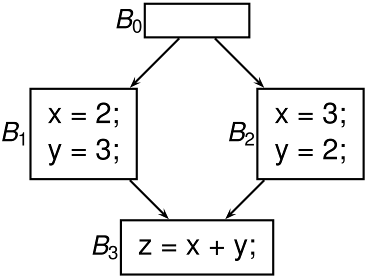
- All paths:

Nondistributivity of CP



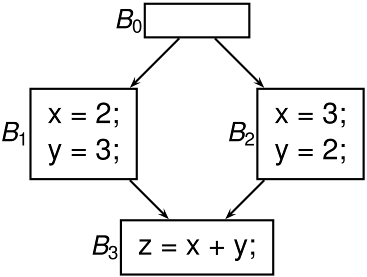
- All paths:
  - $B_0 \rightarrow B_1 \rightarrow B_3$

Nondistributivity of CP



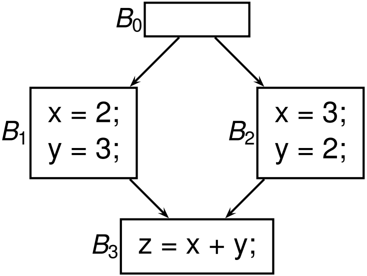
- All paths:
  - $B_0 \to B_1 \to B_3$
  - $B_0 \to B_2 \to B_3$

Nondistributivity of CP



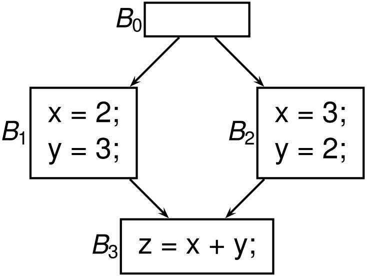
- All paths:
  - $B_0 \rightarrow B_1 \rightarrow B_3$
  - $B_0 \rightarrow B_2 \rightarrow B_3$
- Value of z is 5 along both the paths.

Nondistributivity of CP



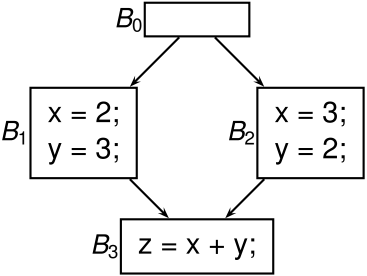
- All paths:
  - $B_0 \rightarrow B_1 \rightarrow B_3$
  - $B_0 \rightarrow B_2 \rightarrow B_3$
- Value of z is 5 along both the paths.
- MOP value for z is 5.

Nondistributivity of CP



- All paths:
  - $B_0 \rightarrow B_1 \rightarrow B_3$
  - $B_0 \rightarrow B_2 \rightarrow B_3$
- Value of z is 5 along both the paths.
- MOP value for z is 5.
- MFP value for z is *NAC*. (Exercise)

Nondistributivity of CP



- All paths:
  - $B_0 \rightarrow B_1 \rightarrow B_3$
  - $B_0 \rightarrow B_2 \rightarrow B_3$
- Value of z is 5 along both the paths.
- MOP value for z is 5.
- MFP value for z is NAC. (Exercise)
- MFP value $\neq$ MOP value (MFP $<$ MOP)