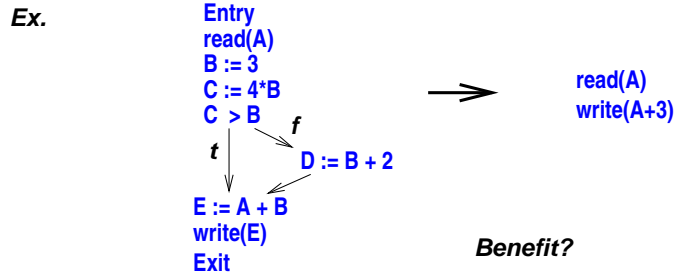


Constant Propagation

Discover values that are constant on all possible executions, and propagate values

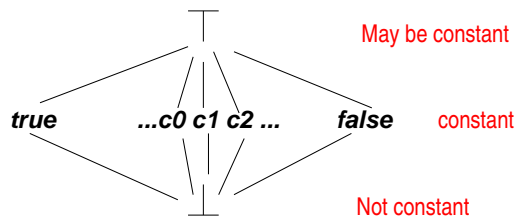
Undecidable \rightarrow Conservative methods

Can create new constants in process by constant folding



1

Lattice for Constant Propagation



$$any \sqcap T = any$$

$$any \sqcap \perp = \perp$$

Initial value:

$$c_i \sqcap c_j = \begin{cases} c_i & \text{if } i=j \\ \perp & \text{if } i \neq j \end{cases}$$

T

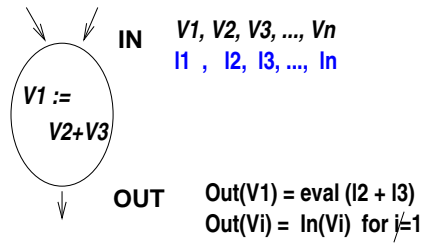
2

Simple Constant Propagation (Kildall)

Propagate constants through CFG, each branch equally likely

Worklist algorithm

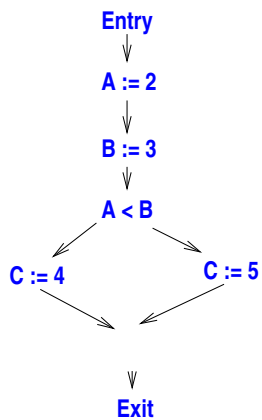
$O(E * N^2)$



$$eval(I2 + I3) = \begin{cases} I2 + I3 & \text{if both are constants} \\ \perp & \text{otherwise} \end{cases}$$

3

Simple CP Example



4

Sparse Simple CP (Reif & Lewis)

Propagate constants through SSA graph (def-use chains from SSA)
for expected efficiency (size of SSA graph)

Worklist algorithm

Assign lattice values to expressions e
 e not evaluated at compile time: \perp
 no variables in e $eval(e)$
 o.w. \perp

Initialize WL to set of SSA edges where def is not \perp

Algorithm terminates when WL is empty

5

Sparse Simple CP (continued)

Take SSA edge off WL

$V1 :=$	$V1, V2, V3, \dots, Vn$
	$I1, I2, I3, \dots, In$
\downarrow	
$e: \dots$	$Use(V1)$
	$V1, V2, V3, \dots, Vn$
	$J1, J2, J3, \dots, Jn$

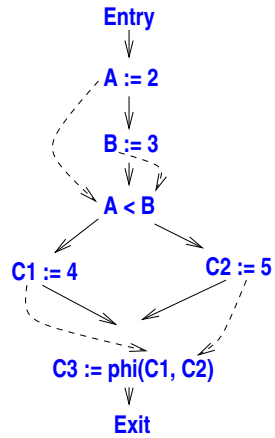
Compute $n1 = i1 \sqcap j1$

If $n1 \neq j1$, set $j1 = n1$

Evaluate e ; if its value has changed, add to WL all SSA edges
from e 's node

6

Sparse Simple CP Example



7

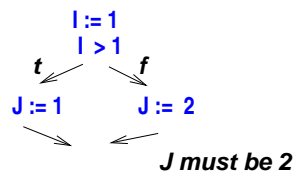
Conditional Constant Prop. (Wegbreit)

Propagate constants through CFG, taking branches into account

Performs CP + Unreachable code elimination

More powerful than SCP or SSCP

Ex.



Executable Edges:

Determined by symbolic execution, starting from Entry

If node with single successor executed, that successor edge is marked executable

If predicate is executed, evaluate predicate and determine branch taken, if possible

8

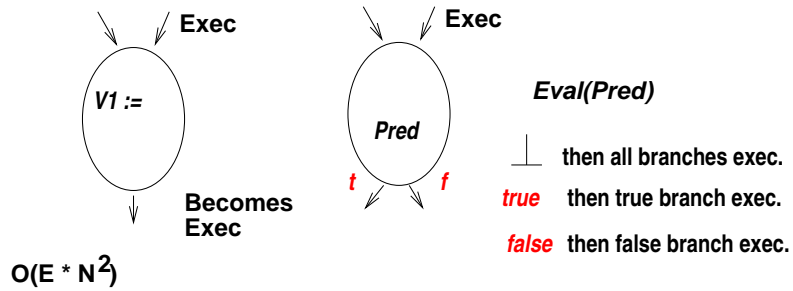
Conditional CP (continued)

Worklist algorithm:

Each edge is initialized to be non-executable

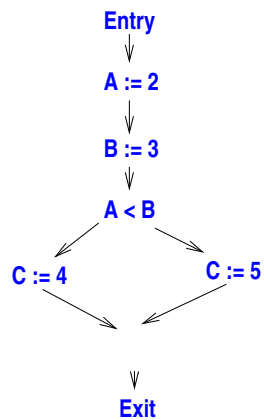
The edge from Entry is marked executable, and put on WL

Proceed by symbolic execution:



9

Conditional CP Example



10

Sparse Conditional CP (Wegman,Zadeck)

Like Conditional CP but uses SSA graph for efficiency
Like Sparse Simple CP but meet applied only to arguments which
correspond to executable incoming edges to Phi function

$O(\# \text{ Flow edges} + \# \text{ SSA edges})$

Assumes only 1 statement or predicate per node

Worklist Algorithm

FlowWL flow graph edge work list, initialized to edges out of Entry

SSAWL SSA edge work list, initialized to empty set

ExecFlag(e) records whether flow graph edge e is executable

LatCell(V,B) lattice value for variable V at node B

11

Sparse Conditional CP Algorithm

Do while (FlowWL nonempty or SSA WL nonempty)

Take edge e from WL.

If e is a flow edge, then if it is executable, do nothing

else Set ExecFlag to true

Call VisitPhi for all Phi functions at e 's target

If this is the first time the target node is visited,

Call VisitExp

if the target node has 1 out flow edge, add to WL

If e is a SSA edge, and its target has a Phi function, call VisitPhi

if its target is an expression, and an incoming flow
edge to it is executable, then Call VisitExp

ow do nothing

12

Sparse Conditional CP Algorithm, cont.

VisitPhi: Called when value of Lattice Cell of operand is lowered, or when edge becomes executable

$ex \searrow \text{nex} \downarrow \text{ex} \swarrow$
 $V4 := \text{Phi}(V1, V2, V3)$
 $I1, \perp, I3$

Compute $I4$ by taking meet of these values

VisitExp: Evaluate expression, taking values from lattice cells where defined

If value of e changes,

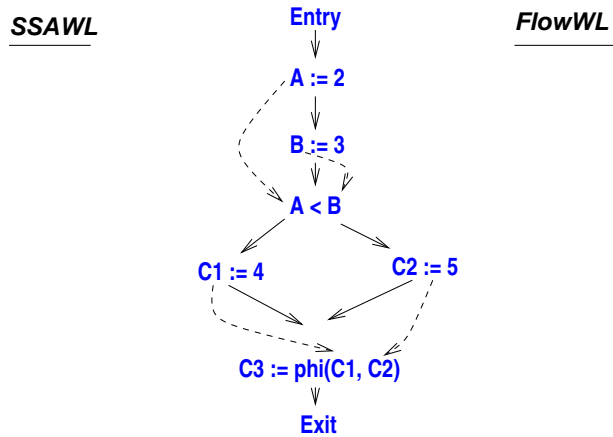
if its an asst., add all SSA edges starting at asst. to SSAWL

if its a pred., and value is \perp , then add all exit edges to FlowWL

value is a constant, add appropriate edge to Flow WL

13

Sparse Simple CP Example



14