SWEN430 - Compilers

Lecture 20 - Register Allocation II

David J. Pearce & Alex Potanin & Roma Klapaukh

School of Engineering and Computer Science
Victoria University of Wellington

Live Variables Analysis

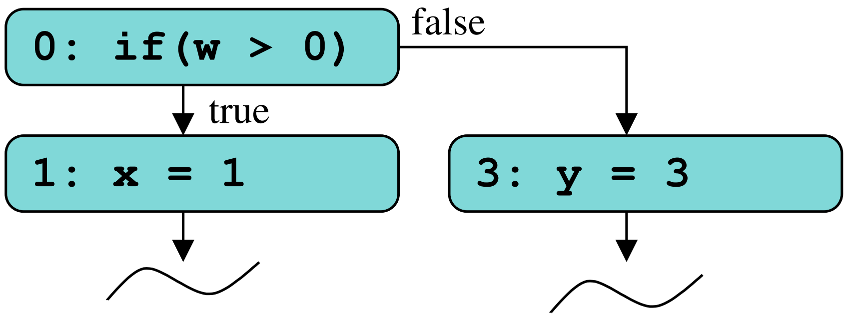
Dataflow Equations

Let $D = (V, E)$ represent the CFG of a method. Then, the dataflow equations for a node $v \in V$ are:

$$LIVE_{IN}(v) = \Big((LIVE_{OUT}(v) - DEF_{AT}(v)) \cup \ USES(v) \Big)$$

$$LIVE_{OUT}(v) = \bigcup_{v \to w \in E} LIVE_{IN}(w)$$

- This is a **backwards** flow-analysis. Compute $LIVE_{OUT}(v)$ by unioning $LIVE_{IN}(w)$ for each successor w of v:



Here, $LIVE_{OUT}(0) = LIVE_{IN}(1) \cup LIVE_{IN}(3)$

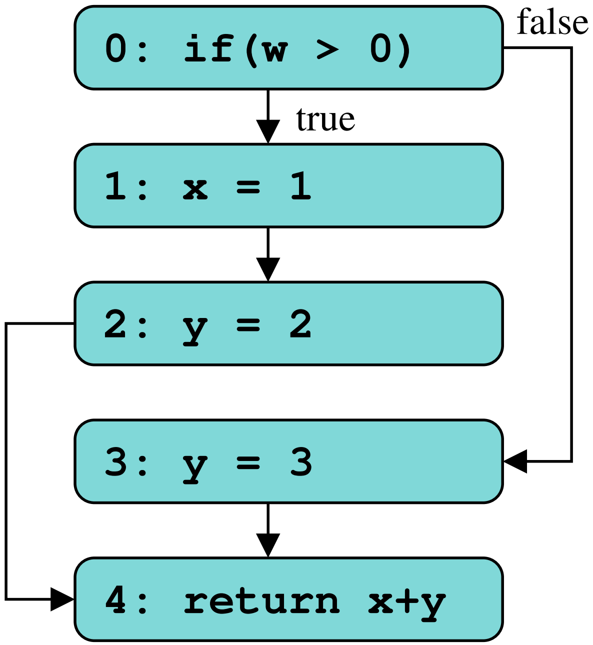
Live Variables: Solving Dataflow Equations

- Basic Algorithm:
  1. Set $LIVE_{OUT}(v) = LIVE_{IN}(v) = \emptyset$ for all nodes v
  2. Select node v
  3. Calculate $LIVE_{OUT}(v)$ based on current solution
  4. Calculate $LIVE_{IN}(v)$ using $LIVE_{OUT}(v)$
  5. Repeat from step 2 — *until solution sets reach fixed point*

  » Guaranteed to *terminate* and yield *smallest* (i.e. most precise) solution for dataflow equations (provided select is fair)
  » Solution sets reach fixed point when they no longer change

- Iteration Strategies:
  » Choice of select function affects **performance**
  » **Chaotic** — select nodes at *random*
  » **Forward Iterative** — select nodes in *increasing order*; when all done start over
  » **Backward Iterative** — select nodes in *decreasing order*; when all done start over

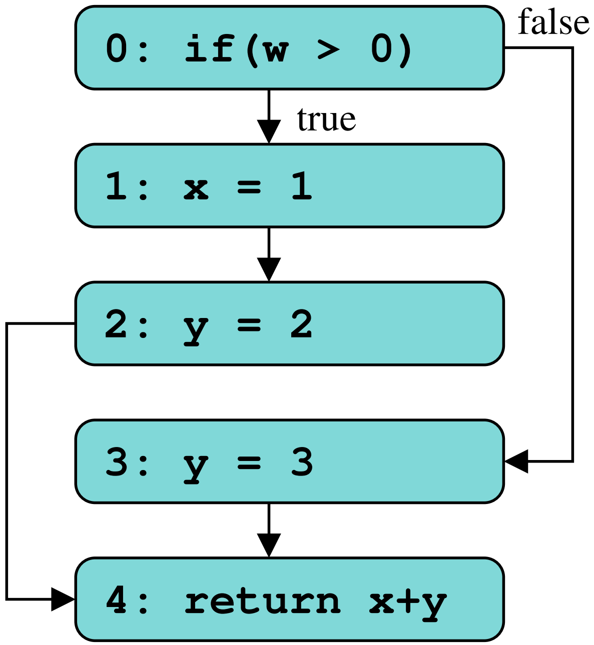
Example: **Forward** Iterative Strategy



Node	0	1	2	3	4
0	$\emptyset$				
1	$\emptyset$				
2	$\emptyset$				
3	$\emptyset$				
4	$\emptyset$				

(Initial state)

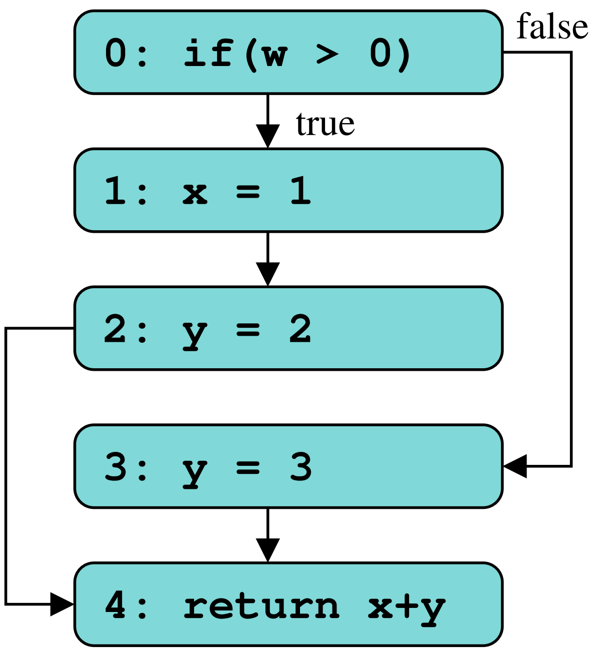
Example: **Forward** Iterative Strategy



Node	0	1	2	3	4
0	$\emptyset$	$\{w\}$			
1	$\emptyset$	$\emptyset$			
2	$\emptyset$	$\emptyset$			
3	$\emptyset$	$\emptyset$			
4	$\emptyset$	$\{x, y\}$			

(State after one iteration)

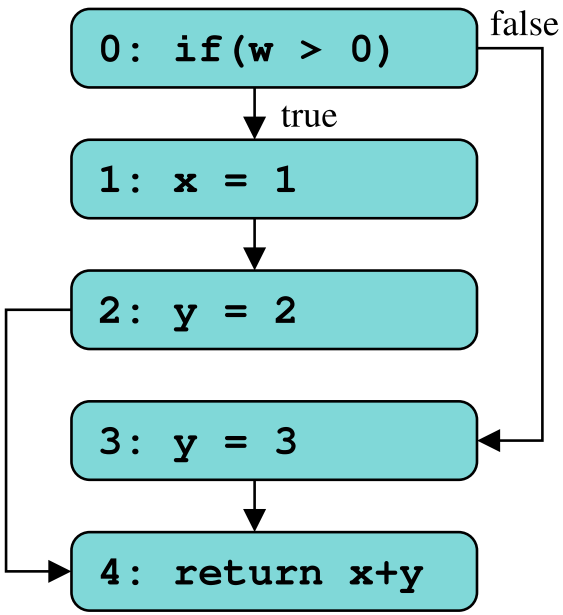
Example: **Forward** Iterative Strategy



Node	0	1	2	3	4
0	$\emptyset$	$\{w\}$	$\{w\}$		
1	$\emptyset$	$\emptyset$	$\emptyset$		
2	$\emptyset$	$\emptyset$	$\{x\}$		
3	$\emptyset$	$\emptyset$	$\{x\}$		
4	$\emptyset$	$\{x, y\}$	$\{x, y\}$		

(State after two iterations)

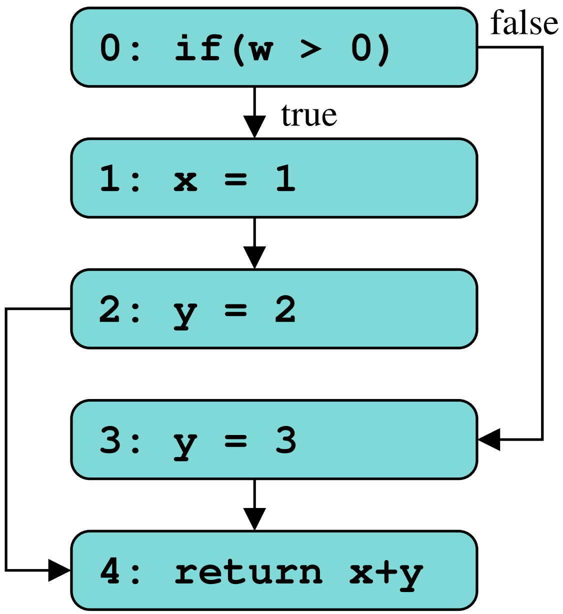
Example: **Forward** Iterative Strategy



Node	0	1	2	3	4
0	$\emptyset$	$\{w\}$	$\{w\}$	$\{w, x\}$	
1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	
2	$\emptyset$	$\emptyset$	$\{x\}$	$\{x\}$	
3	$\emptyset$	$\emptyset$	$\{x\}$	$\{x\}$	
4	$\emptyset$	$\{x, y\}$	$\{x, y\}$	$\{x, y\}$	

(State after three iterations)

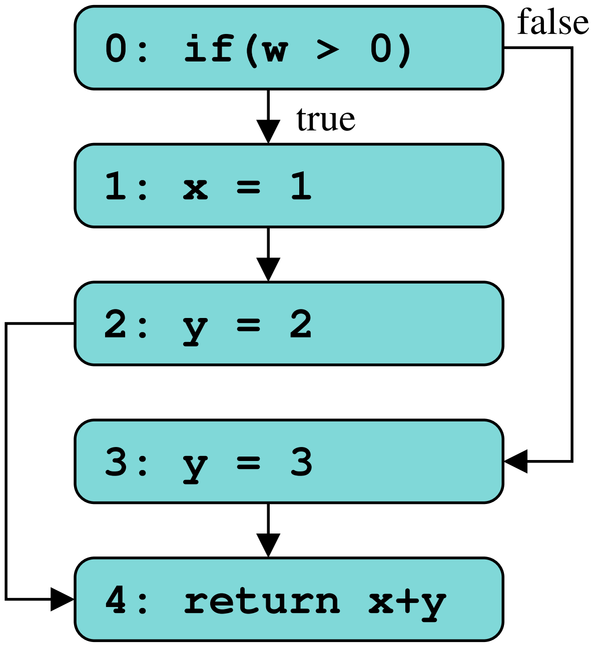
Example: **Forward** Iterative Strategy



Node	0	1	2	3	4
0	$\emptyset$	$\{w\}$	$\{w\}$	$\{w, x\}$	$\{w, x\}$
1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
2	$\emptyset$	$\emptyset$	$\{x\}$	$\{x\}$	$\{x\}$
3	$\emptyset$	$\emptyset$	$\{x\}$	$\{x\}$	$\{x\}$
4	$\emptyset$	$\{x, y\}$	$\{x, y\}$	$\{x, y\}$	$\{x, y\}$

(State after four iterations — fixed-point reached)

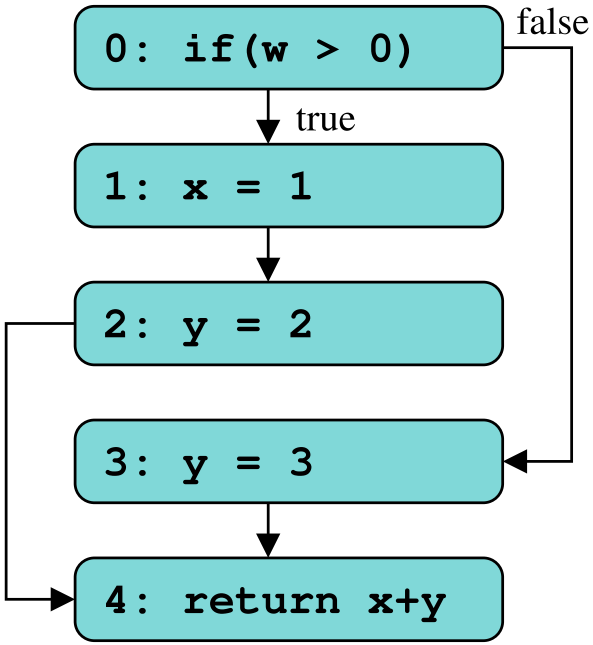
Example: **Backward** Iterative Strategy



Node	0	1	2	3	4
0	$\emptyset$				
1	$\emptyset$				
2	$\emptyset$				
3	$\emptyset$				
4	$\emptyset$				

(Initial State)

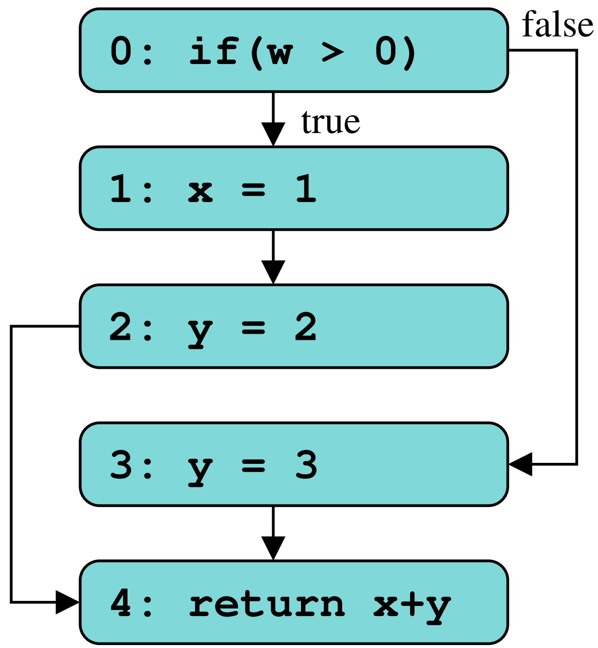
Example: **Backward** Iterative Strategy



Node	0	1	2	3	4
0	$\emptyset$	$\{w, x\}$			
1	$\emptyset$	$\emptyset$			
2	$\emptyset$	$\{x\}$			
3	$\emptyset$	$\{x\}$			
4	$\emptyset$	$\{x, y\}$			

(State after one iteration)

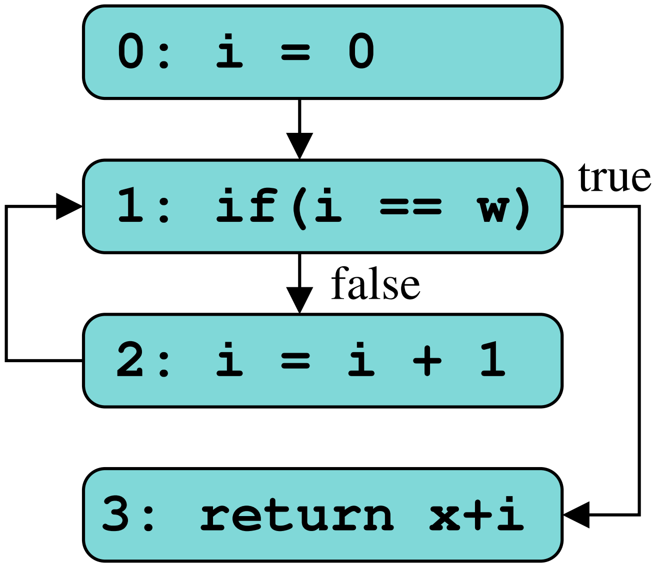
Example: **Backward** Iterative Strategy



Node	0	1	2	3	4
0	$\emptyset$	$\{w, x\}$	$\{w, x\}$		
1	$\emptyset$	$\emptyset$	$\emptyset$		
2	$\emptyset$	$\{x\}$	$\{x\}$		
3	$\emptyset$	$\{x\}$	$\{x\}$		
4	$\emptyset$	$\{x, y\}$	$\{x, y\}$		

(State after two iterations — fixed-point reached)

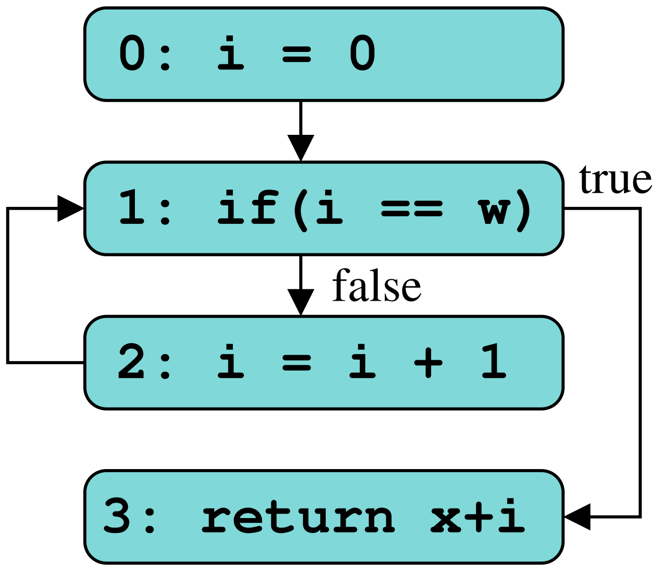
Another Example: **Backward** Iterative Strategy



Node	0	1	2	3	4
0	$\emptyset$				
1	$\emptyset$				
2	$\emptyset$				
3	$\emptyset$				

(Initial State)

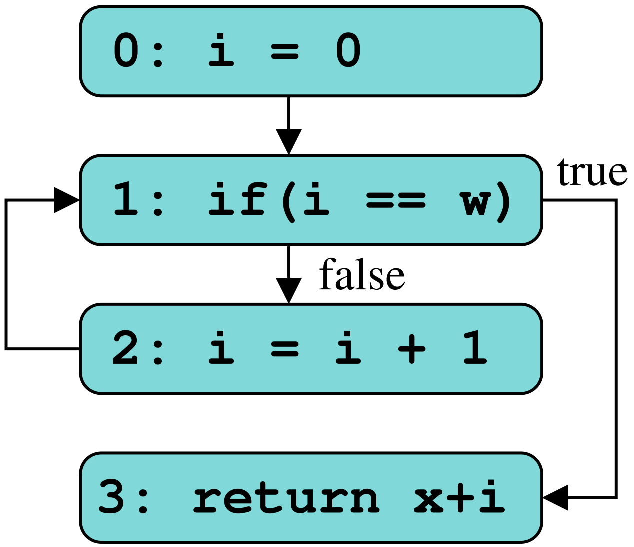
Another Example: **Backward** Iterative Strategy



Node	0	1	2	3	4
0	$\emptyset$	$\{x, w\}$			
1	$\emptyset$	$\{x, w, i\}$			
2	$\emptyset$	$\{i\}$			
3	$\emptyset$	$\{x, i\}$			

(State after one iteration)

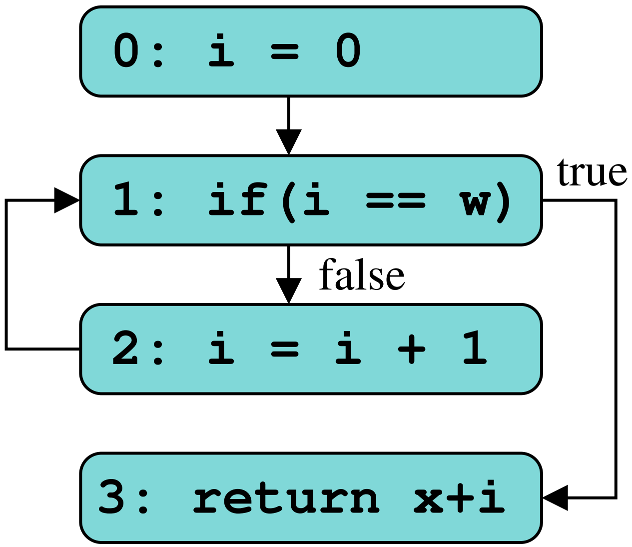
Another Example: **Backward** Iterative Strategy



Node	0	1	2	3	4
0	$\emptyset$	$\{x, w\}$	$\{x, w\}$		
1	$\emptyset$	$\{x, w, i\}$	$\{x, w, i\}$		
2	$\emptyset$	$\{i\}$	$\{x, w, i\}$		
3	$\emptyset$	$\{x, i\}$	$\{x, i\}$		

(State after two iterations)

Another Example: **Backward** Iterative Strategy

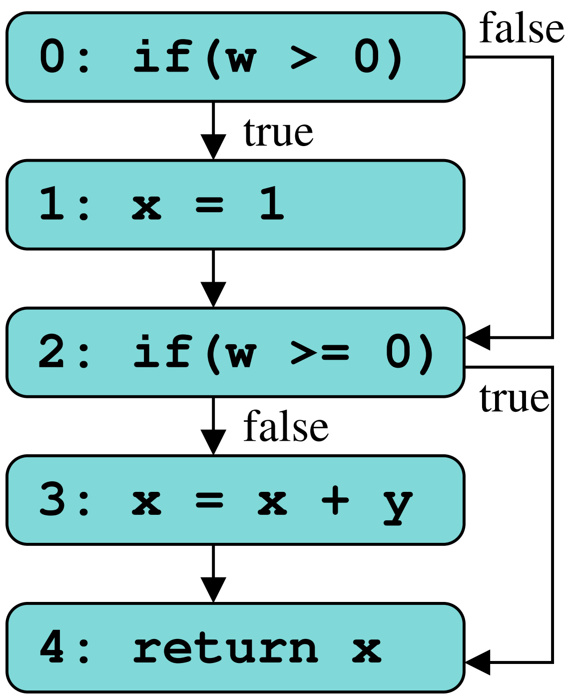


Node	0	1	2	3
0	$\emptyset$	$\{x, w\}$	$\{x, w\}$	$\{x, w\}$
1	$\emptyset$	$\{x, w, i\}$	$\{x, w, i\}$	$\{x, w, i\}$
2	$\emptyset$	$\{i\}$	$\{x, w, i\}$	$\{x, w, i\}$
3	$\emptyset$	$\{x, i\}$	$\{x, i\}$	$\{x, i\}$

(State after three iterations — fixed-point reached)

Limitations of Live Variable Analysis

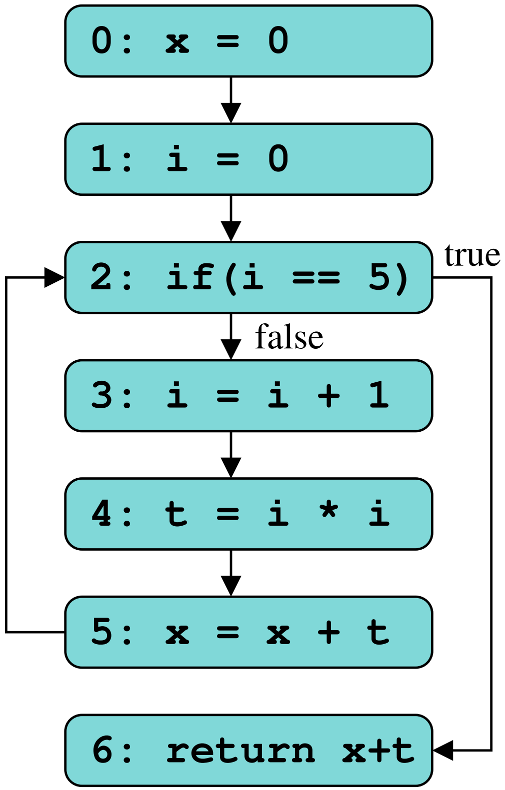
- Consider the solution for this method:



Does our dataflow algorithm produce most accurate solution possible?

Limitations of Live Variable Analysis

- Consider the solution for this method:



Does our dataflow algorithm produce most accurate solution possible?

Generating Interference Graph

Interference Graph

An *interference graph* is an undirected graph containing exactly one node per live range. An edge between two nodes exists iff their live ranges interfere.

- Simple algorithm:
  1. Compute Live Variables Analysis
  2. Create graph G with one node per variable
  3. Consider $\{a, b\} \subseteq LIVE_{OUT}(v)$ for any node v, then $(a, b) \in G$
  4. Consider $\{a, b\} \subseteq LIVE_{IN}(v)$ for any node v, then $(a, b) \in G$
- Do we need last step?
- More precise approach is to split out *live ranges*
- How do we determine *live range* information from solution to live variables analysis?