Data-Flow Analysis

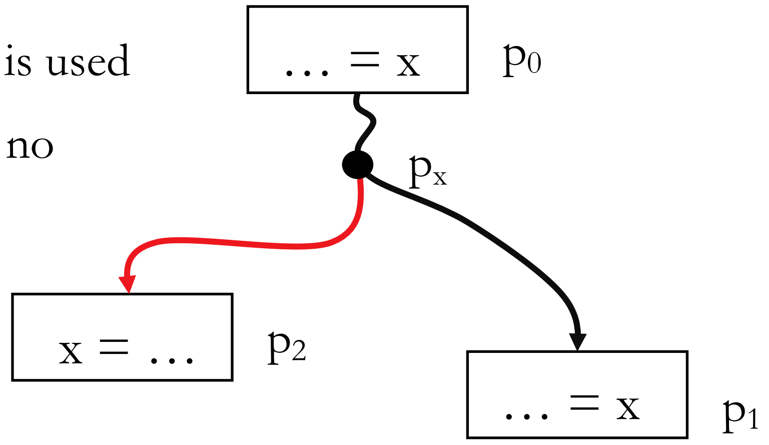
Live-Variable Analysis

Live-Variable Analysis

- What is Live-Variable Analysis?
  - For each Variable x where is the last program point *p* where the a specific value of x is used.
  - In other words, for x and program point *p* determine if the value of x at *p* can still be used along some path starting at *p*.
    - If so, x is live at *p*
    - If not x is dead at *p*
  - Must take Control-Flow into account : a Data-Flow Problem !!!
- Applications:
  - Register Allocation: If a variable is dead at a given point *p*
    - Can reuse its storage, *i.e,* the register it occupies if any;
    - If its value as been modified must save the value to storage unless it is not live on exit of the procedure or loop

Live-Variable Analysis: Illustration

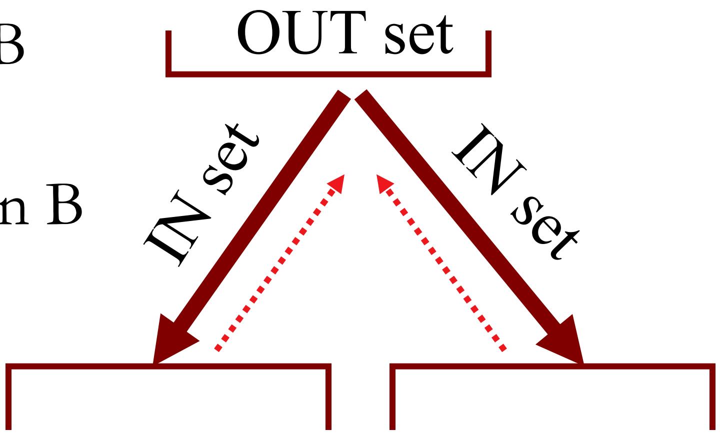
- At point p_0 the x variable is live:
  - There is a path to p_1 where value at p_0 is used
  - Beyond p_x towards p_2 the value of x is no longer needed and is dead



- Need to observe for each variable and for each program point:
  - Where is the last program point beyond which the value is not used
  - Trace back from uses to definitions and observe the first definition (backwards) that reaches that use.
  - That definition kills all uses backwards of it.

Data-Flow Analysis Formulation

- Variable is *live* at a point *p* if its value is used along *at least one Path*
  - A use of x prior to any definition in basic block means x must be alive
  - A definition of x in B prior to any subsequent use means previous uses must be dead
- Gen Set: Set of Variables Used in B
  - Upward Exposed Reads of B
- Kill Set: Set of Variables Defined in B



$$OUT(B) = \bigcup_{S \text{ a successor of } B} IN(s)$$

$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$

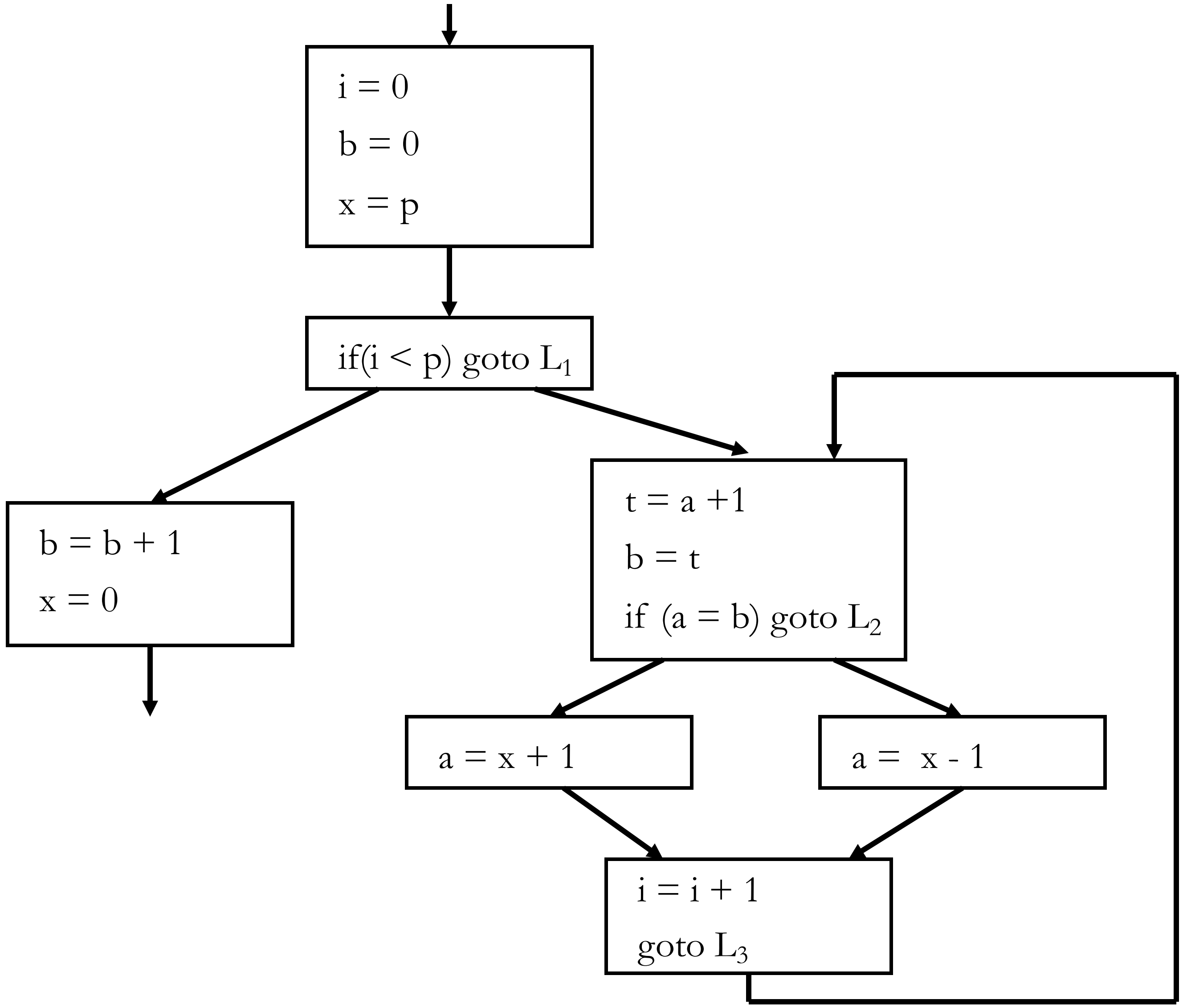
Data-Flow Analysis Formulation

- Initialize IN(B) to Empty Set
- Compute Gen/Use and Kill/Def for each Basic Block
  - Tracing backwards from end of block to beginning of block
  - Initialize Last Instruction's Out(i) to Empty
  - Use $IN(i) = use(i) \cup (OUT(i) - def(i))$

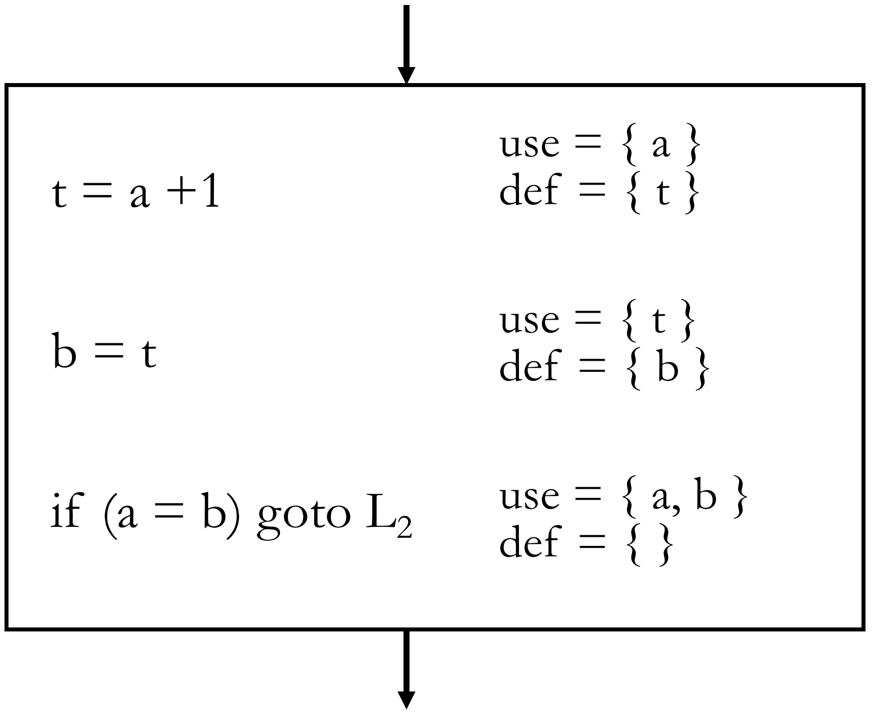
- Iteratively Apply Relations to Basic Block Until Convergence
  - $OUT(B) = \bigcup_{S \text{ a successor of } B} IN(s)$
  - $IN(B) = Use(B) \cup (OUT(B) - Def(B))$

- Given OUT(B) use relations at instruction level to determine the live variables after each instruction

Example


i = 0
b = 0
x = p
if(i < p) goto L_1
b = b + 1
x = 0
t = a +1
b = t
if (a = b) goto L_2
a = x + 1
a = x - 1
i = i + 1
goto L_3

Use & Def Functions for a Basic Block

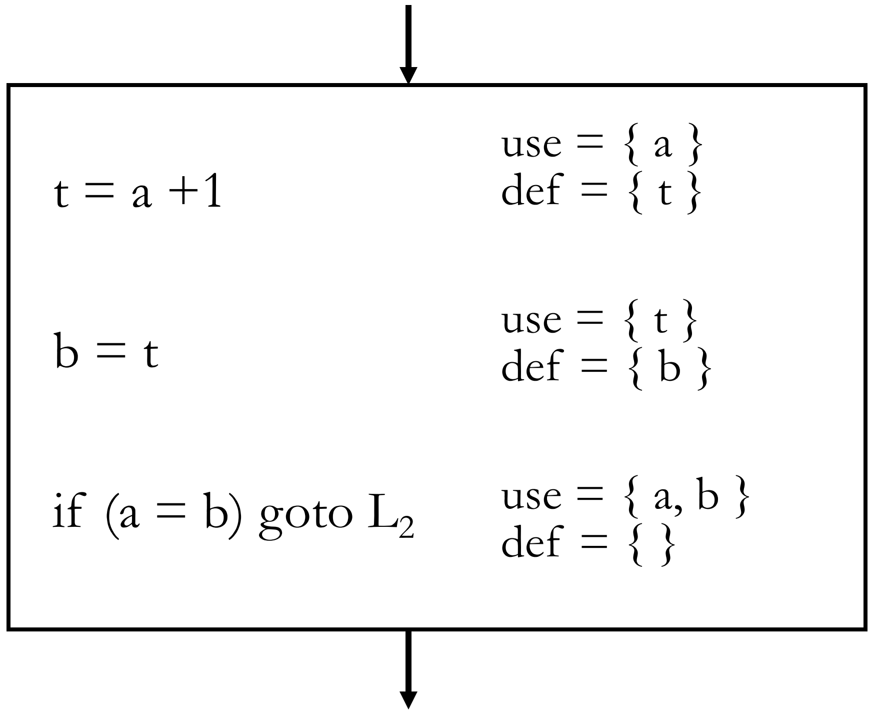


In = Use ∪ (Out - Def)

Out = { }

$$In(i) = Use(i) \cup (Out(i) - Def(i))$$

Use & Def Functions for a Basic Block

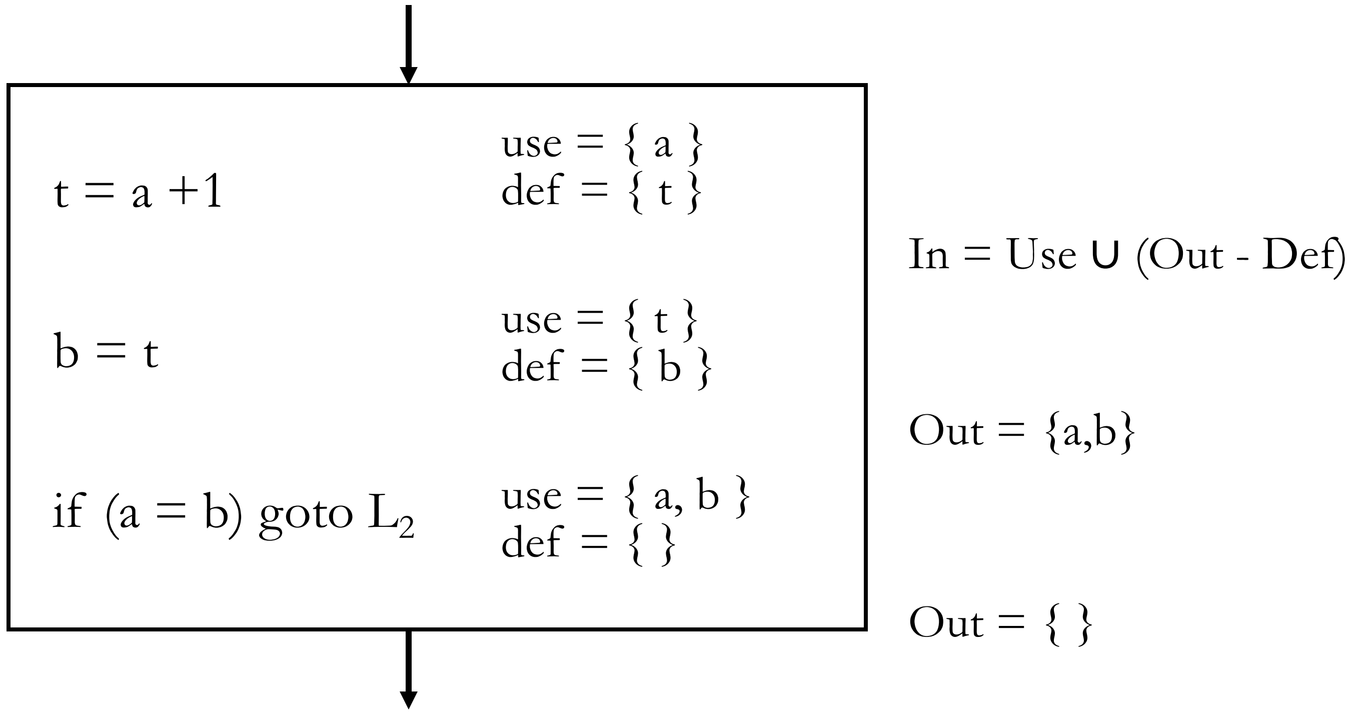


In = {a,b} ∪ ({} - {}) = {a,b}

Out = { }

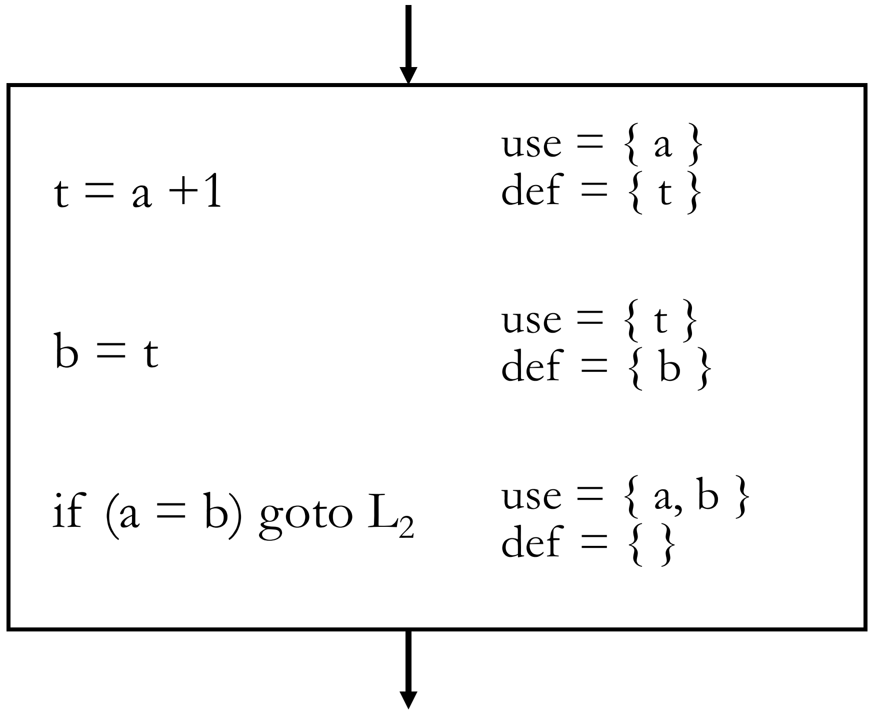
$$In(i) = Use(i) \cup (Out(i) - Def(i))$$

Use & Def Functions for a Basic Block



$$In(i) = Use(i) \cup (Out(i) - Def(i))$$

Use & Def Functions for a Basic Block



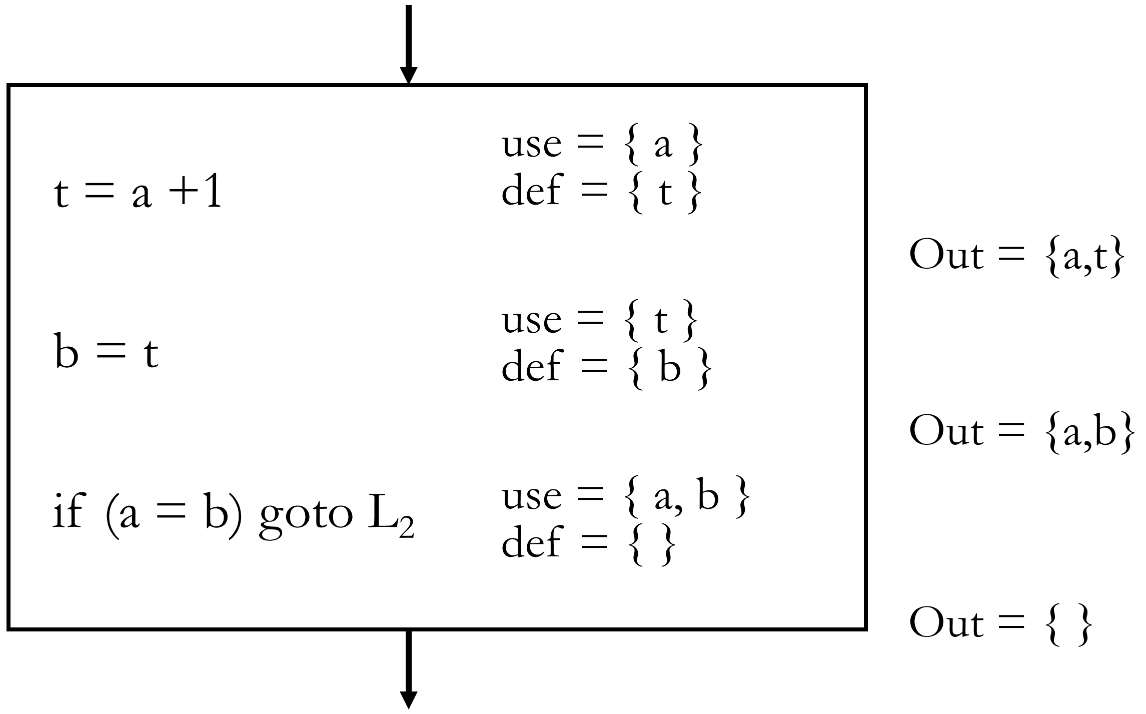
$\text{In} = \{t\} \cup (\{a,b\} - \{b\})$

$\text{Out} = \{a,b\}$

$\text{Out} = \{\ \}$

$$\text{In}(i) = \text{Use}(i) \cup (\text{Out}(i) - \text{Def}(i))$$

Use & Def Functions for a Basic Block



t = a +1 use = { a } def = { t }

Out = {a,t}

b = t use = { t } def = { b }

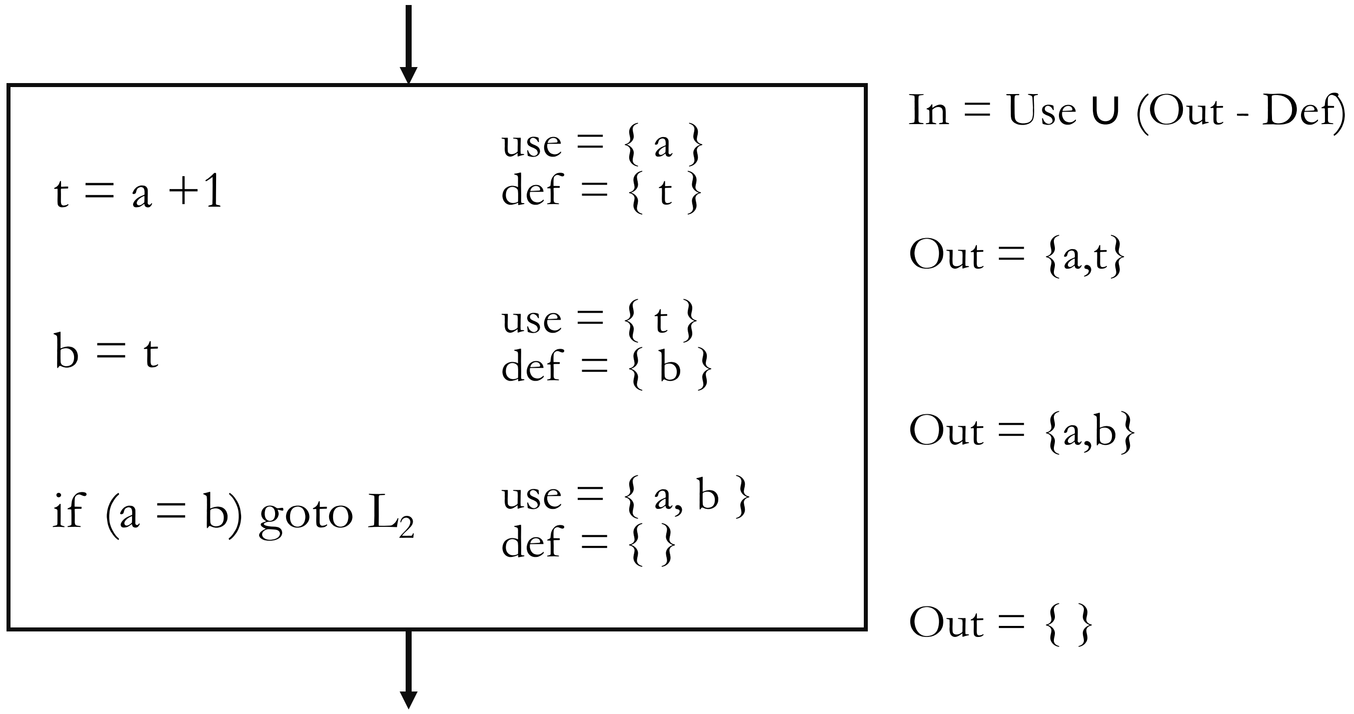
Out = {a,b}

if (a = b) goto L_2 use = { a, b } def = { }

Out = { }

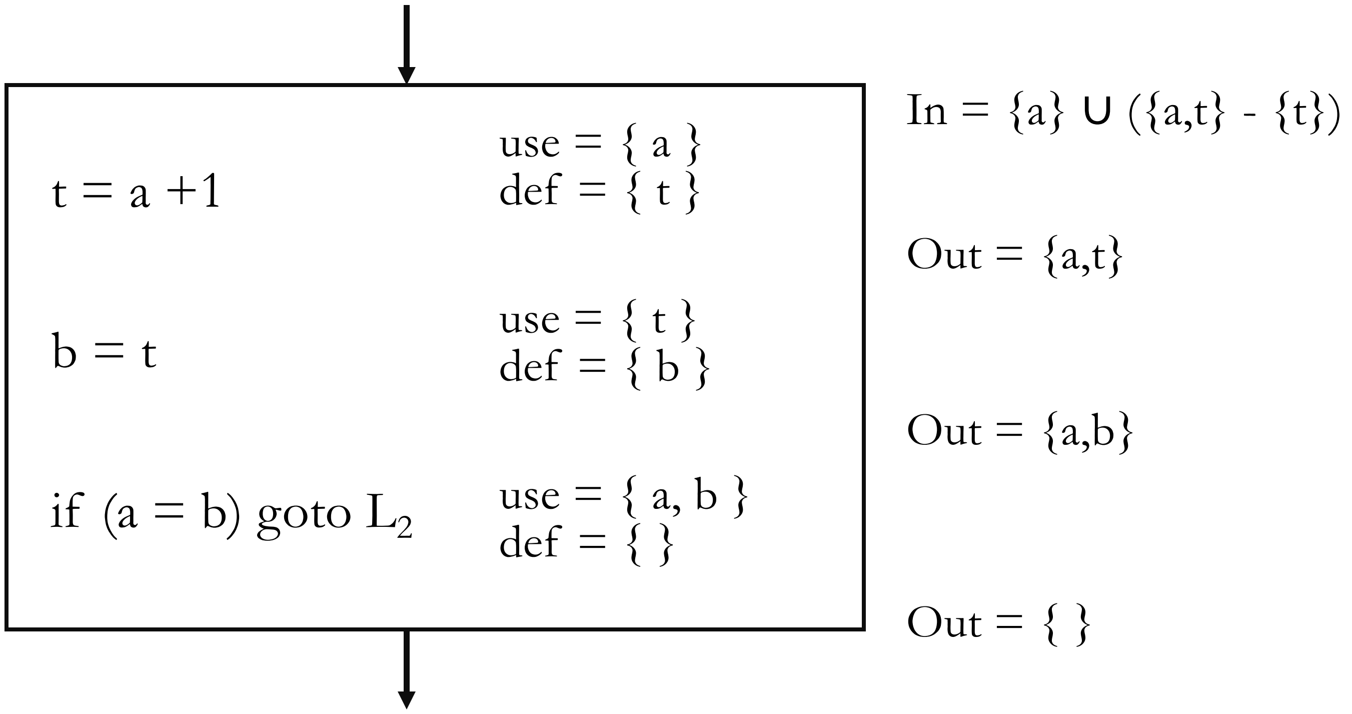
$$In(i) = Use(i) \cup (Out(i) - Def(i))$$

Use & Def Functions for a Basic Block



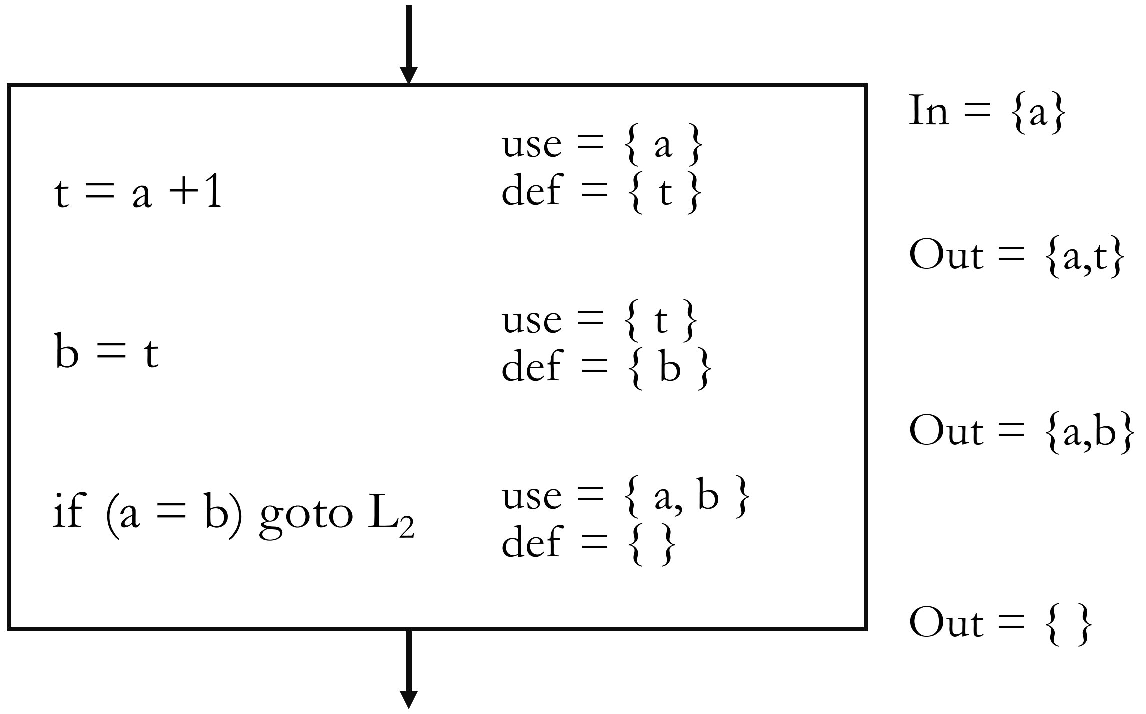
$$In(i) = Use(i) \cup (Out(i) - Def(i))$$

Use & Def Functions for a Basic Block



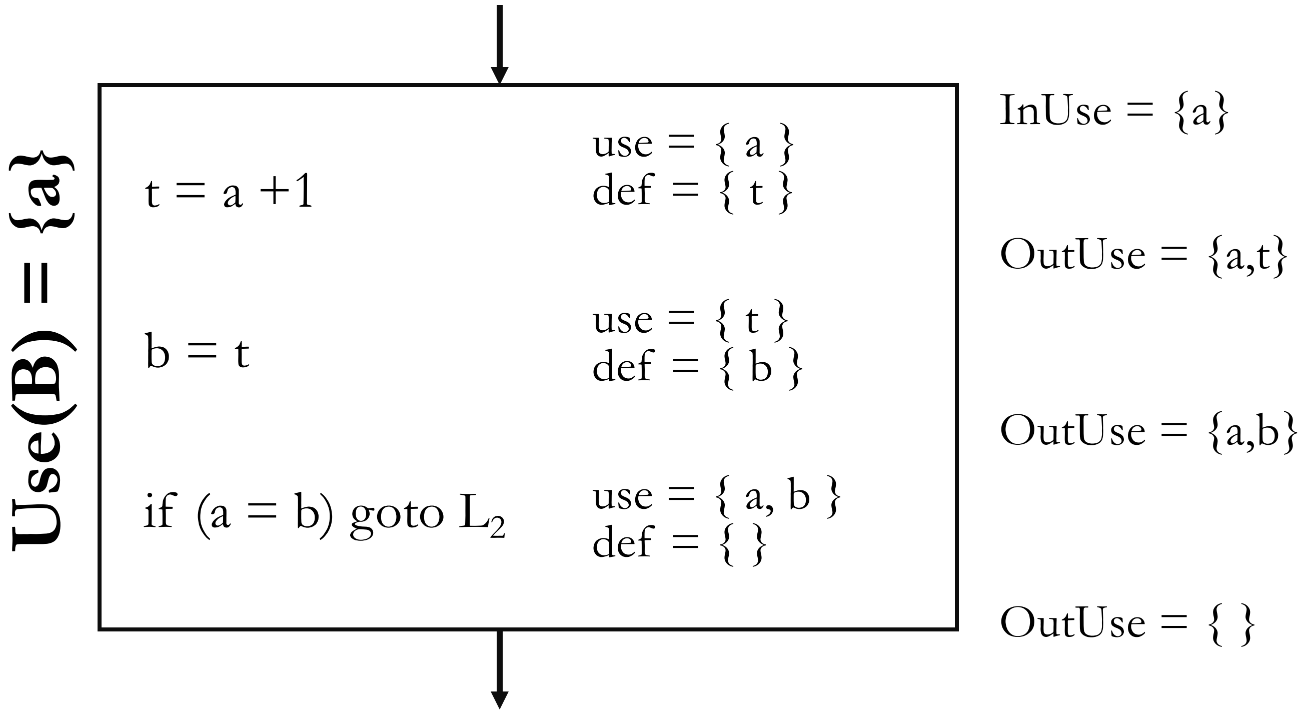
$$In(i) = Use(i) \cup (Out(i) - Def(i))$$

Use & Def Functions for a Basic Block



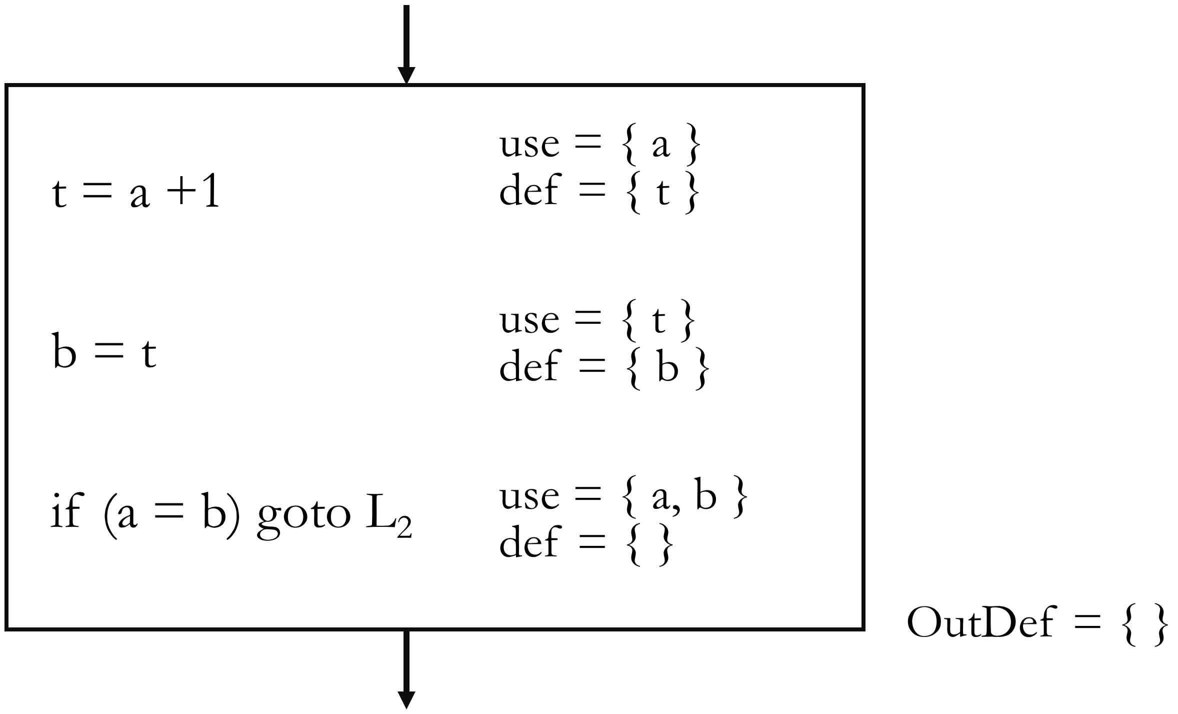
$$In(i) = Use(i) \cup (Out(i) - Def(i))$$

Use & Def Functions for a Basic Block



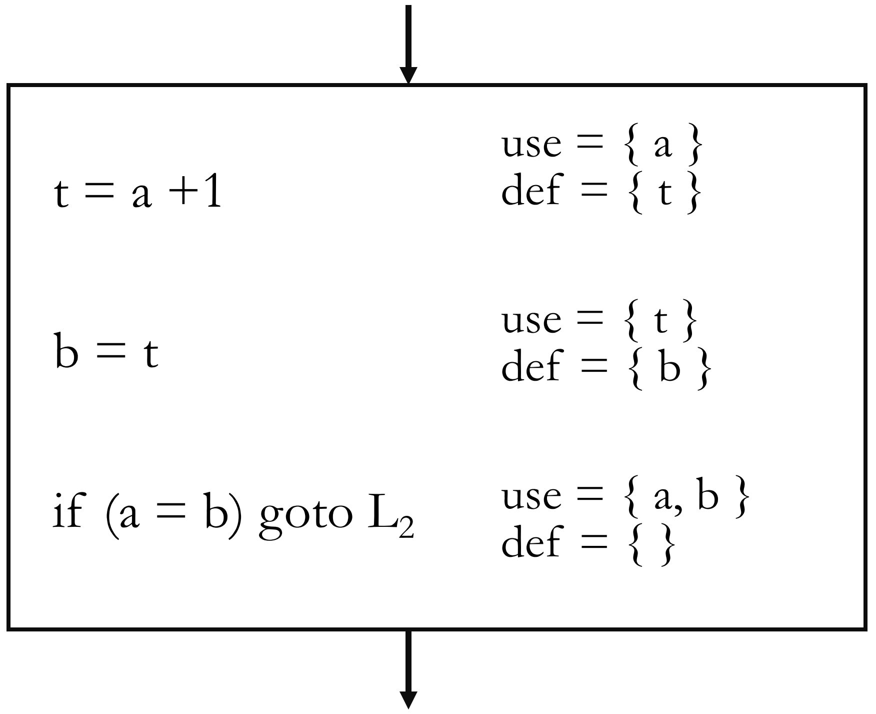
$$InUse(i) = Use(i) \cup (OutUse(i) - Def(i))$$

Use & Def Functions for a Basic Block



$$InDef(i) = Def(i) \cup OutDef(i)$$

Use & Def Functions for a Basic Block

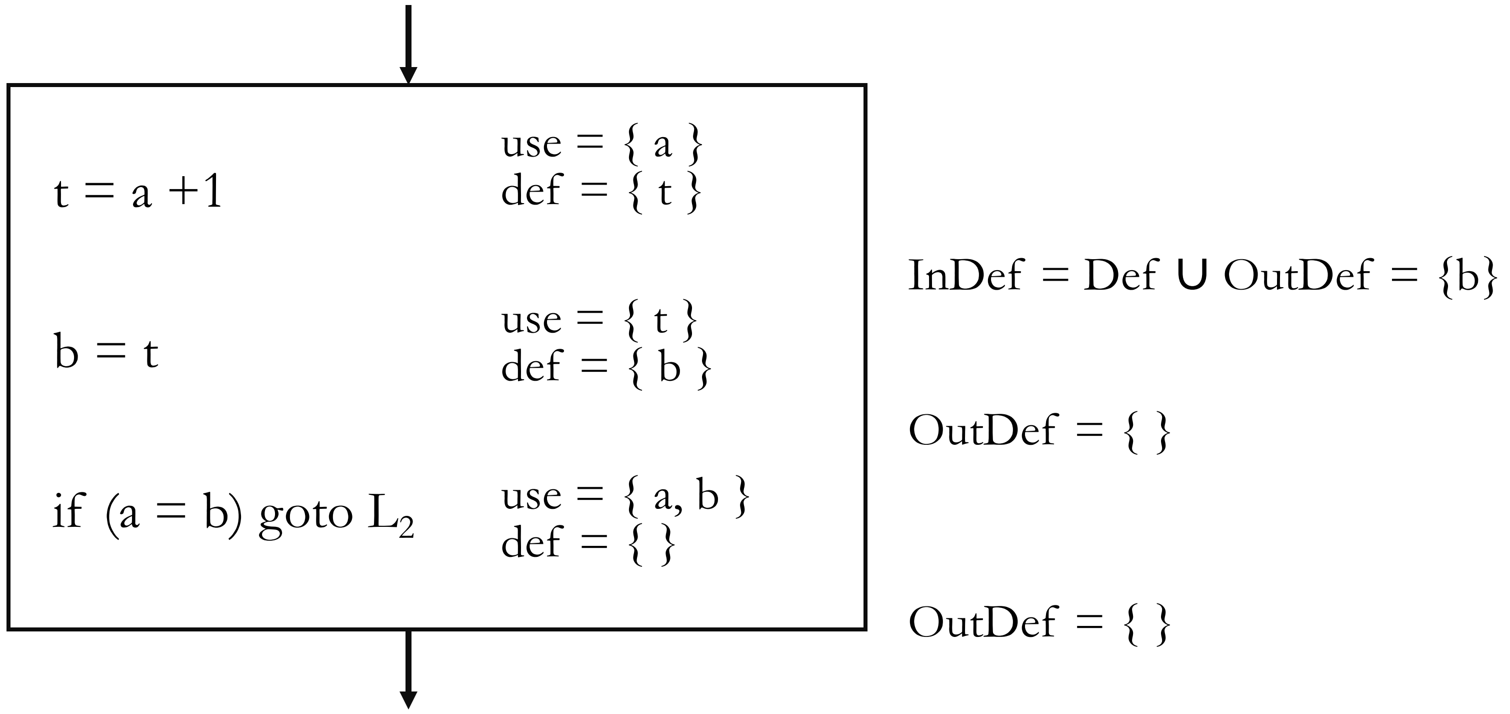


InDef = Def ∪ OutDef = { }

OutDef = { }

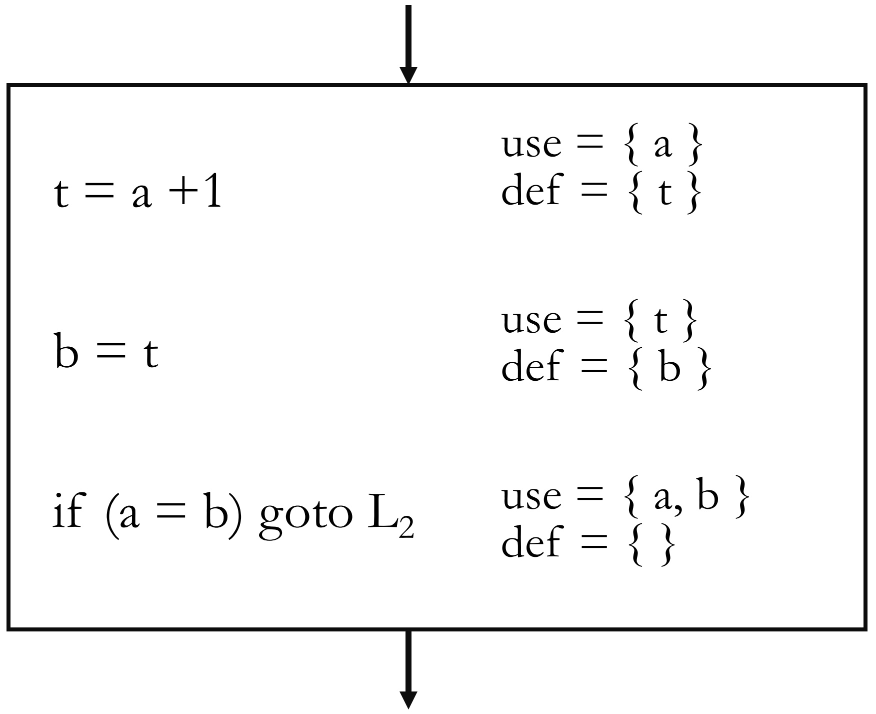
InDef(i) = Def(i) ∪ OutDef(i)

Use & Def Functions for a Basic Block



$$InDef(i) = Def(i) \cup OutDef(i)$$

Use & Def Functions for a Basic Block



InDef = Def ∪ OutDef = {t} ∪ {b}

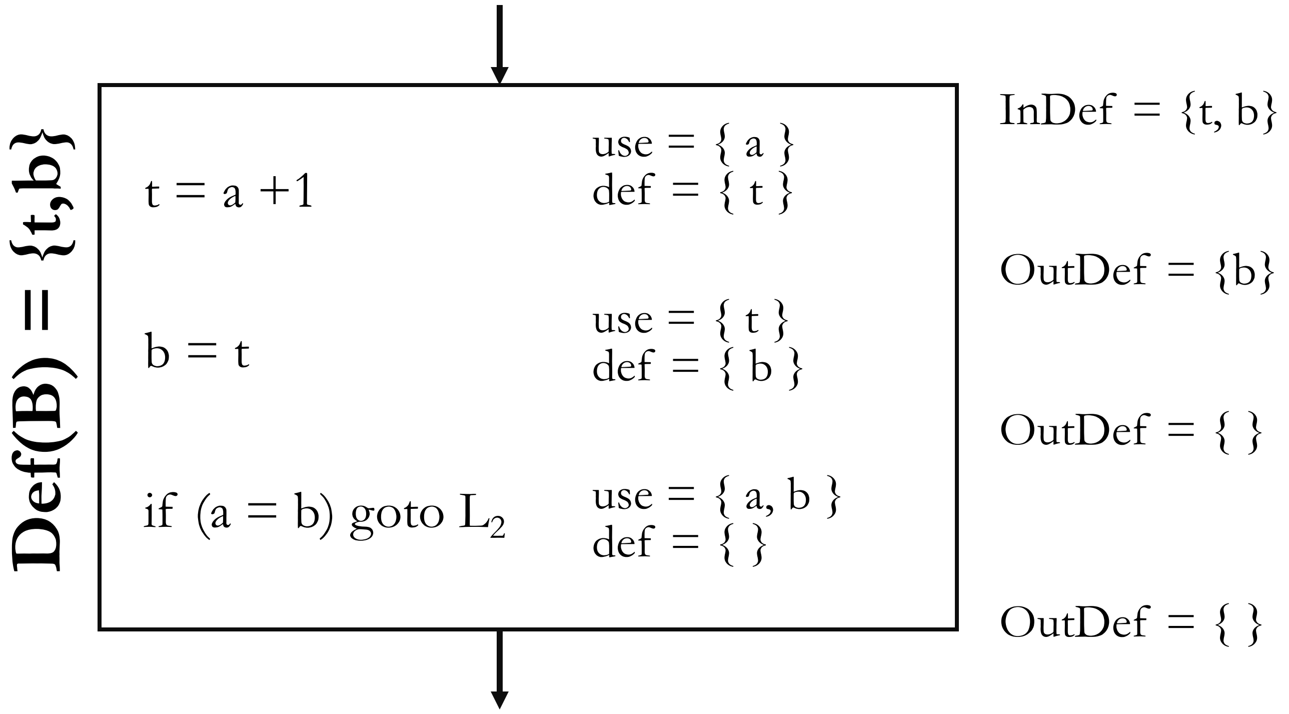
OutDef = {b}

OutDef = { }

OutDef = { }

$$InDef(i) = Def(i) \cup OutDef(i)$$

Use & Def Functions for a Basic Block



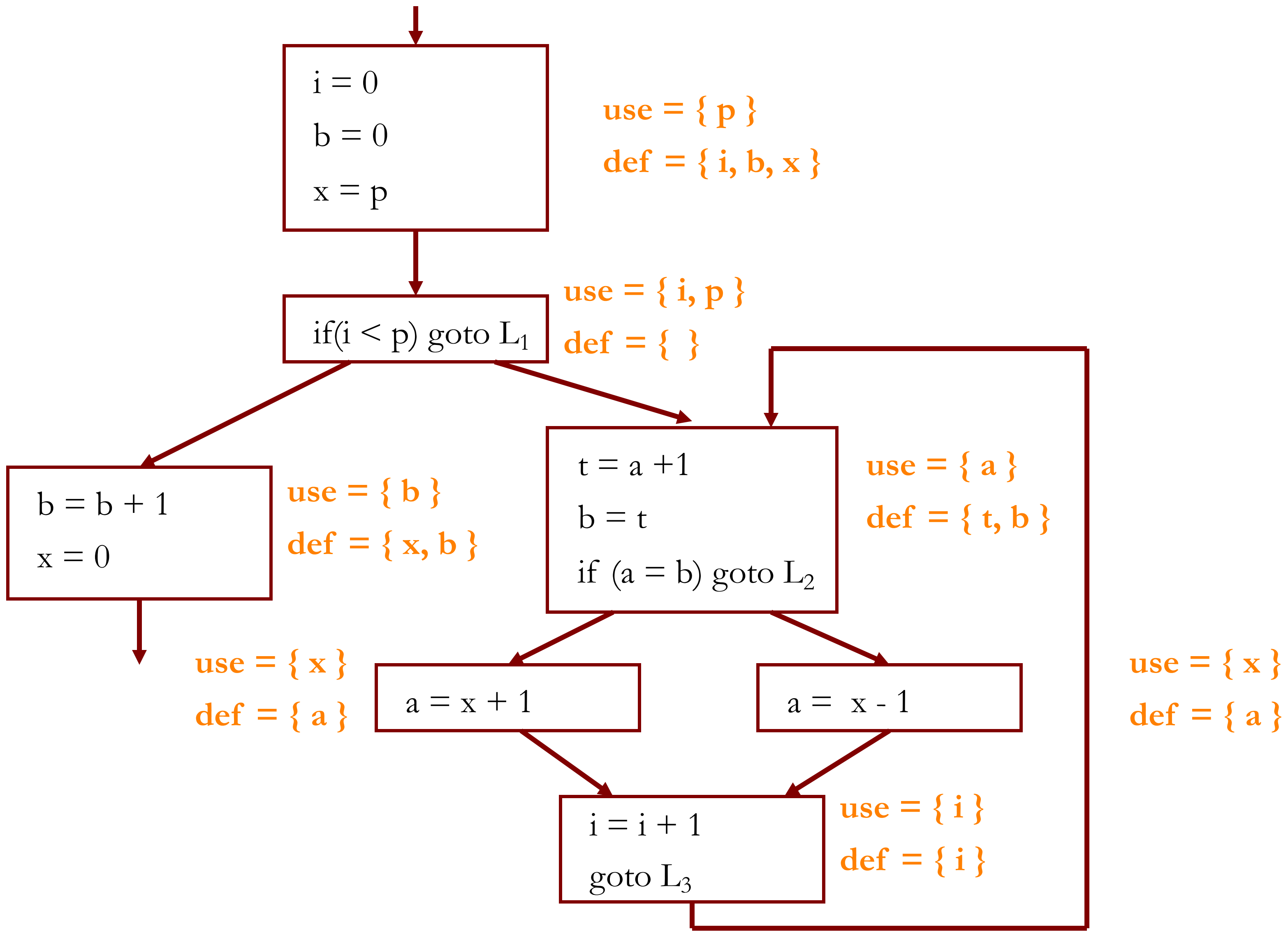
InDef(i) = Def(i) ∪ OutDef(i)

Use & Def Functions for a Basic Block

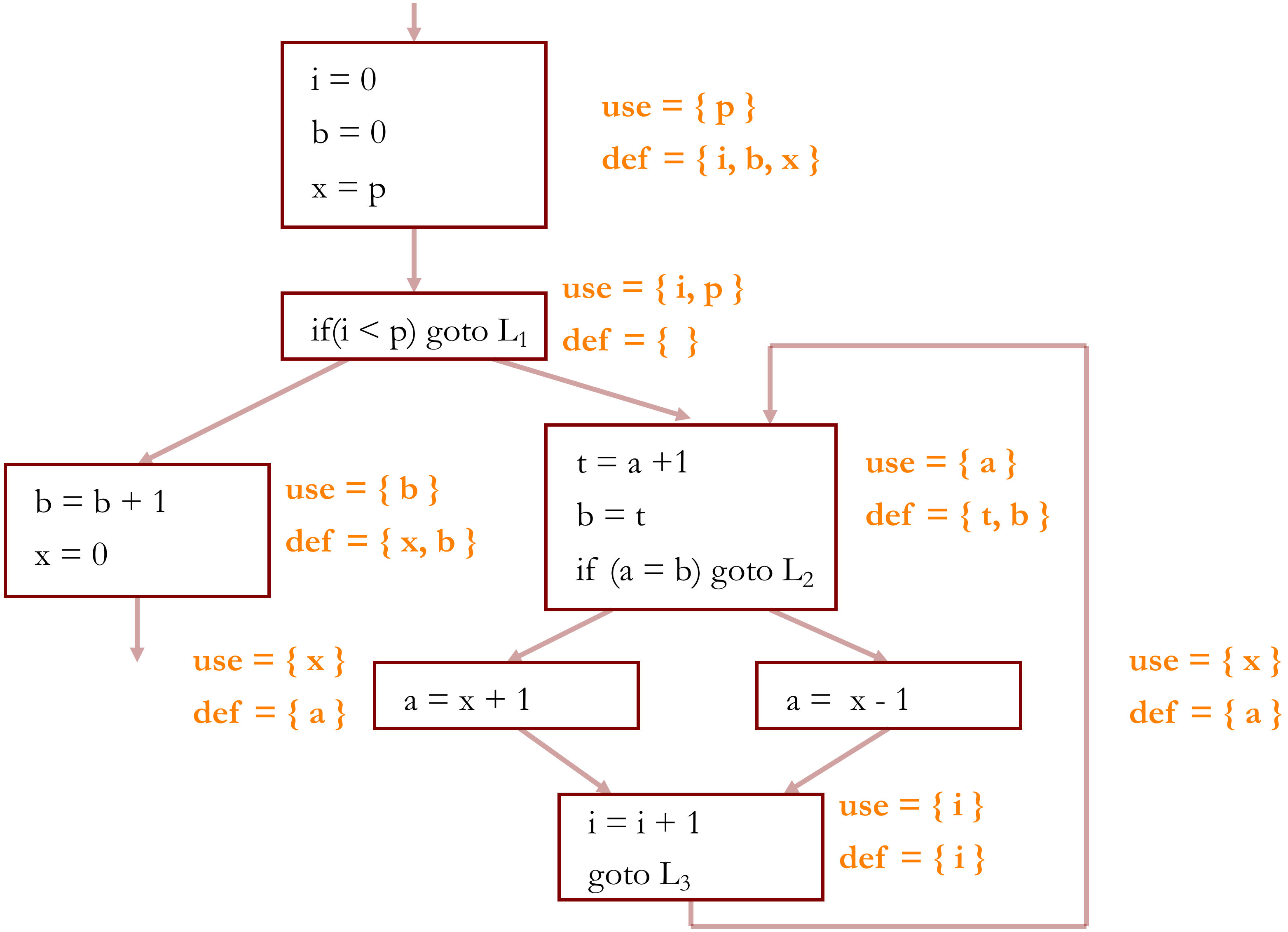
- Can be Accomplished by a Forward Scanning of the Block
  - Keep Track of Which Variables are Read before they are written thus computing the Upwards Exposed Reads (UpExp) or Use Function
  - Track Variables that are Written or Killed (VarKill) or Def Function

```
// Assume instruction in format “x ← y op z”
for i ← 1 to Num Instructions in B do
    if (instr(i) is leader of B) then
        b ← Number(B);
        UpExp(b) ← ∅;
        VarKill(b) ← ∅;
    if y ∉ VarKill(b) then
        UpExp(b) ← UpExp(b) ∪{y}
    if z ∉ VarKill(b) then
        UpExp(b) ← UpExp(b) ∪{z}
    VarKill(b) ← VarKill(b) ∪{x}
```

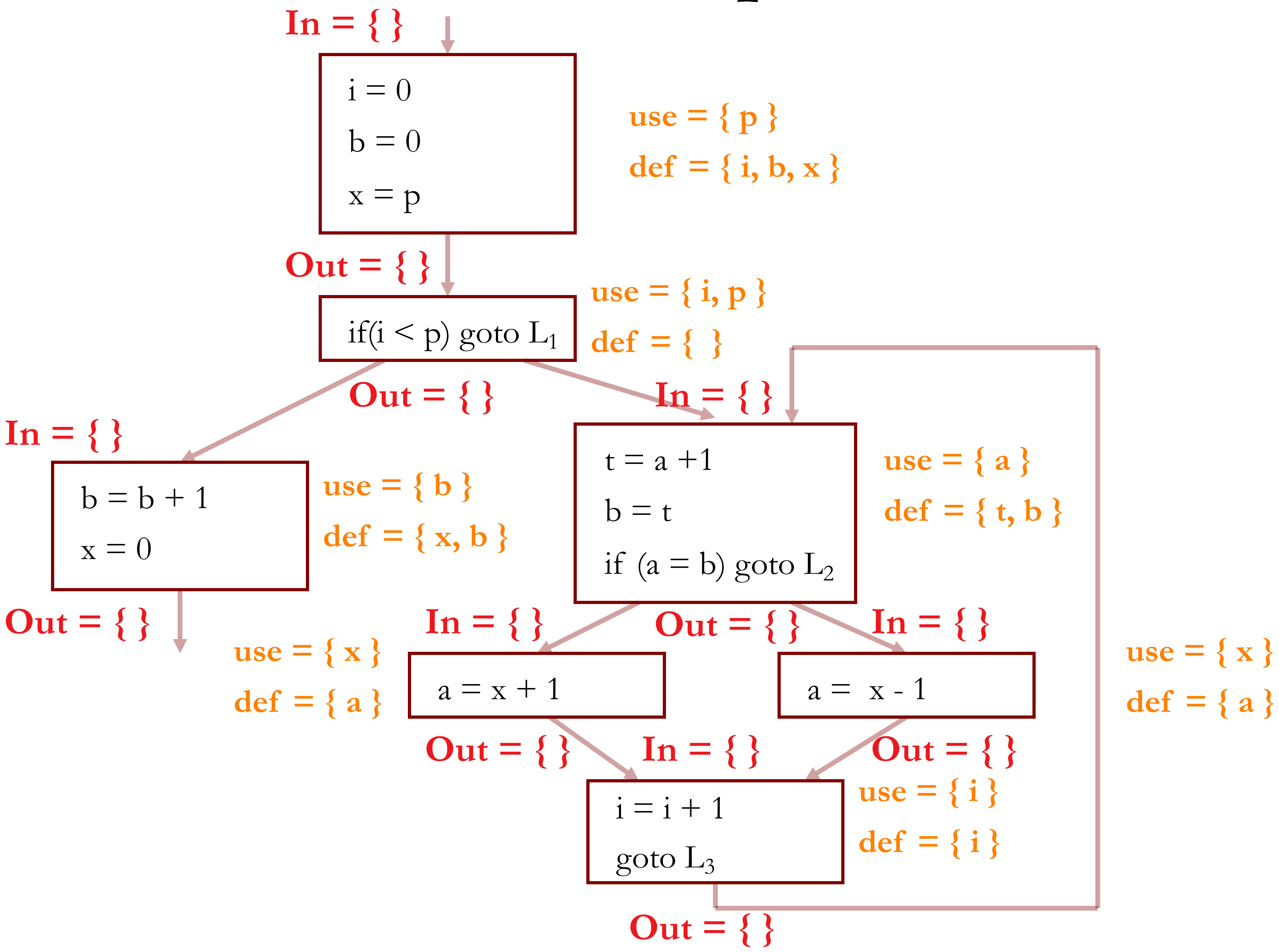
Example


i = 0
b = 0
x = p
use = { p }
def = { i, b, x }
if(i < p) goto L1
use = { i, p }
def = { }
b = b + 1
x = 0
use = { b }
def = { x, b }
t = a +1
b = t
if (a = b) goto L2
use = { a }
def = { t, b }
use = { x }
def = { a }
a = x + 1
a = x - 1
use = { x }
def = { a }
i = i + 1
goto L3
use = { i }
def = { i }

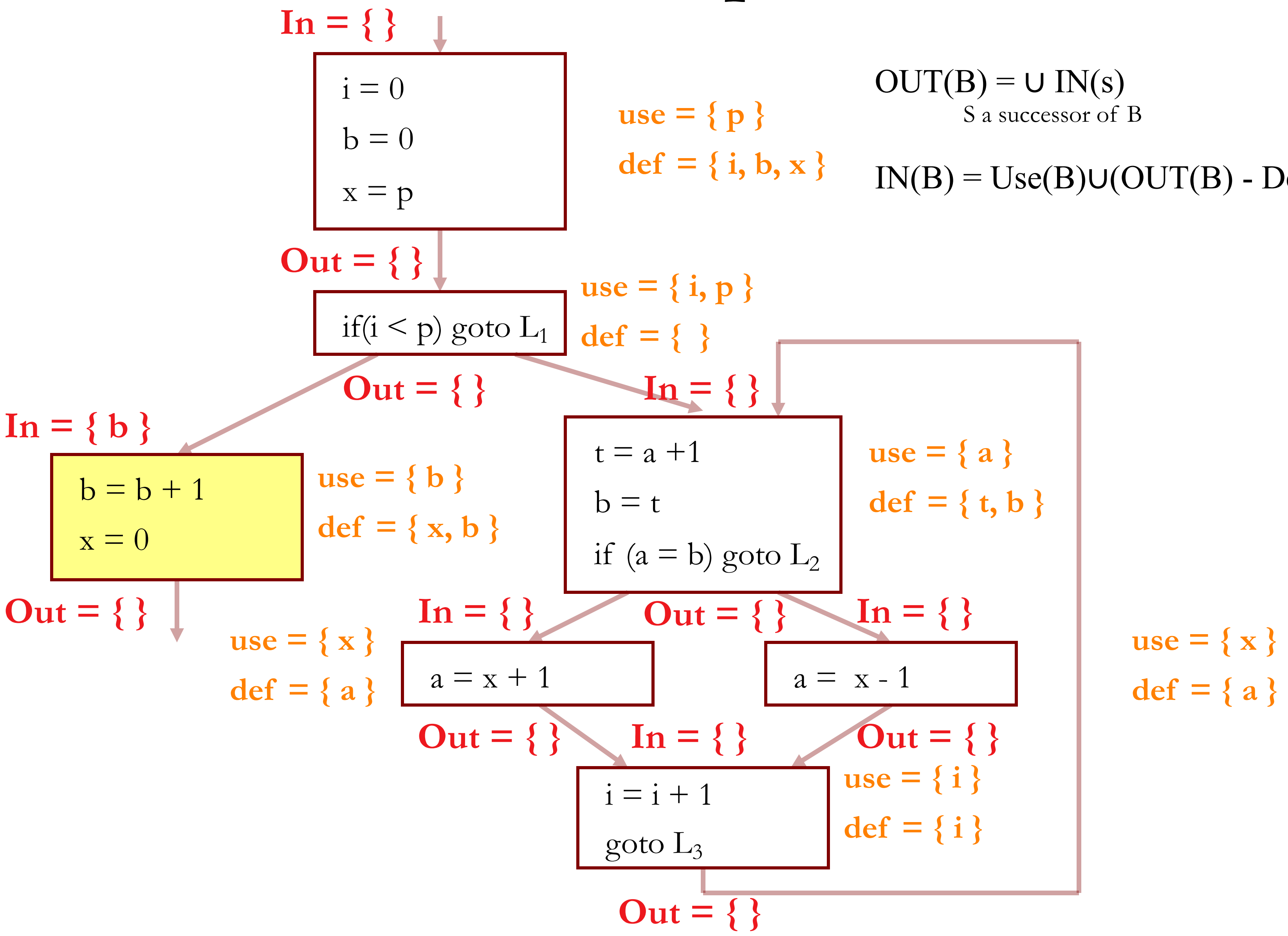
Example


i = 0
b = 0
x = p
use = { p }
def = { i, b, x }
if(i < p) goto L_1
use = { i, p }
def = { }
b = b + 1
x = 0
use = { b }
def = { x, b }
t = a +1
b = t
if (a = b) goto L_2
use = { a }
def = { t, b }
use = { x }
def = { a }
a = x + 1
a = x - 1
use = { x }
def = { a }
i = i + 1
goto L_3
use = { i }
def = { i }

Example


In = { }
i = 0
b = 0
x = p
use = { p }
def = { i, b, x }
Out = { }
use = { i, p }
def = { }
if(i < p) goto L₁
Out = { }
In = { }
In = { }
t = a +1
b = t
if (a = b) goto L₂
use = { a }
def = { t, b }
b = b + 1
x = 0
use = { b }
def = { x, b }
Out = { }
use = { x }
def = { a }
In = { }
Out = { }
In = { }
a = x + 1
a = x - 1
use = { x }
def = { a }
Out = { }
In = { }
Out = { }
i = i + 1
goto L₃
use = { i }
def = { i }
Out = { }

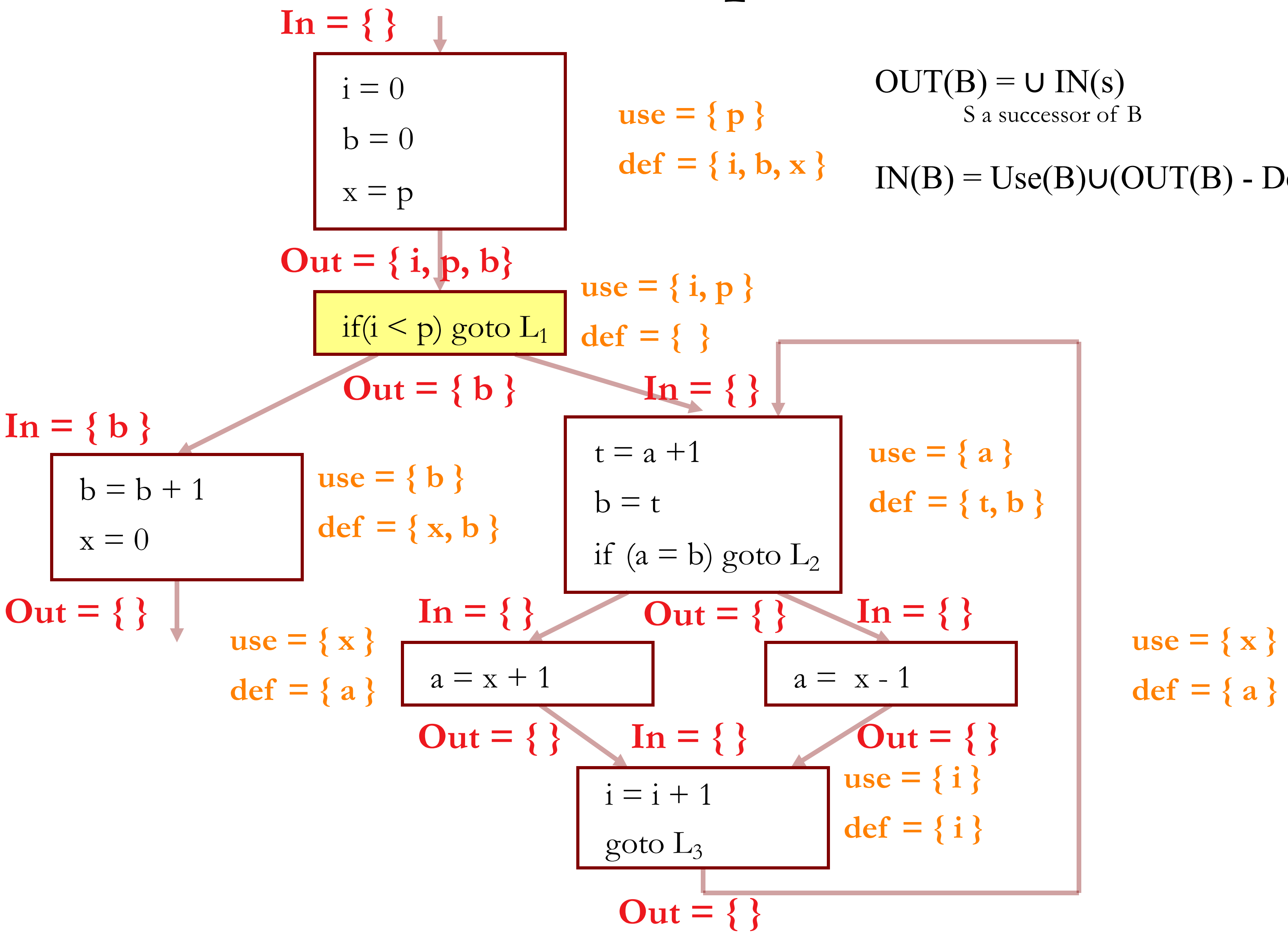
Example



$$OUT(B) = \bigcup_{S \text{ a successor of } B} IN(s)$$

$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$

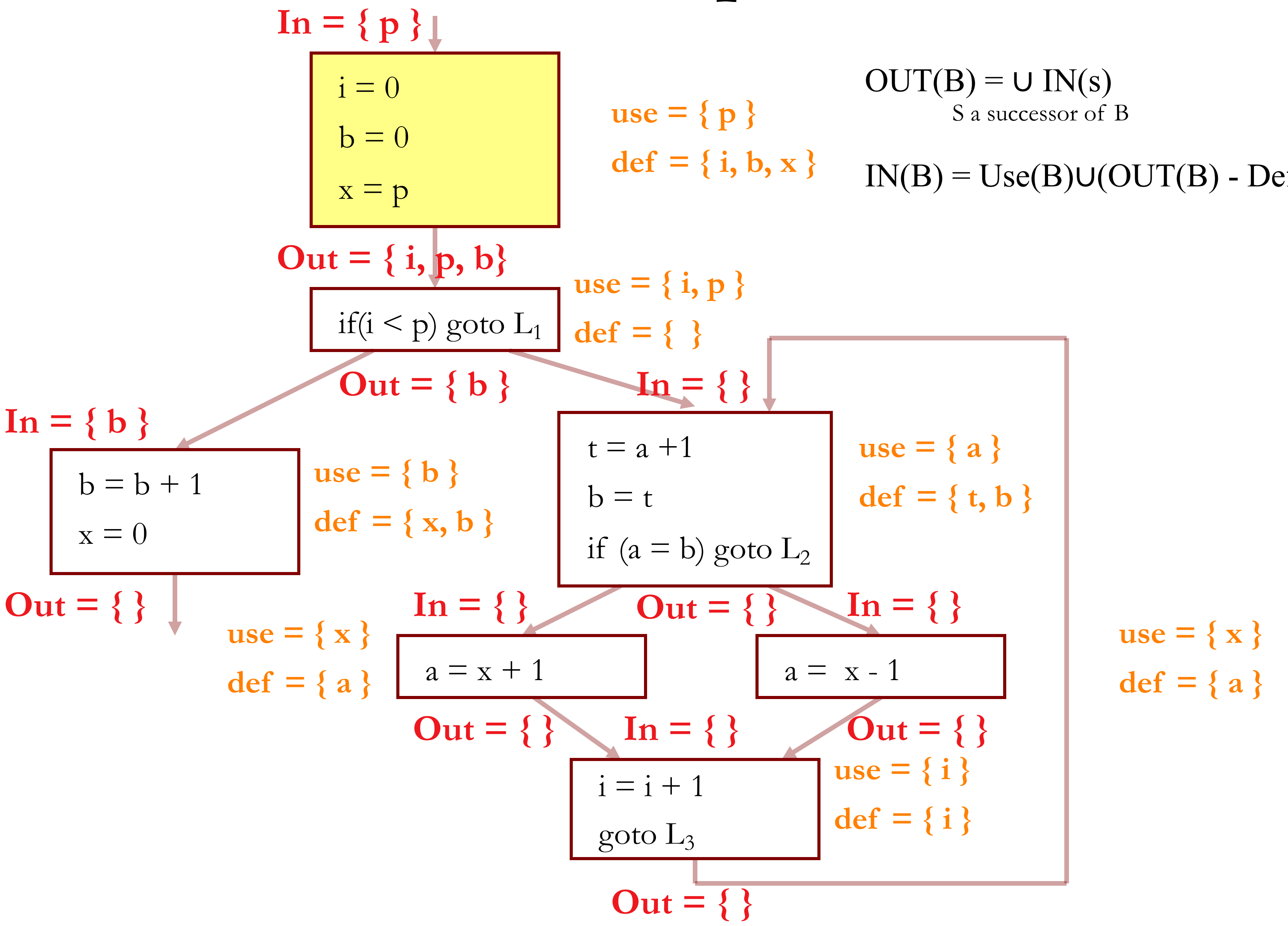
Example



$$OUT(B) = \bigcup_{S \text{ a successor of } B} IN(s)$$

$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$

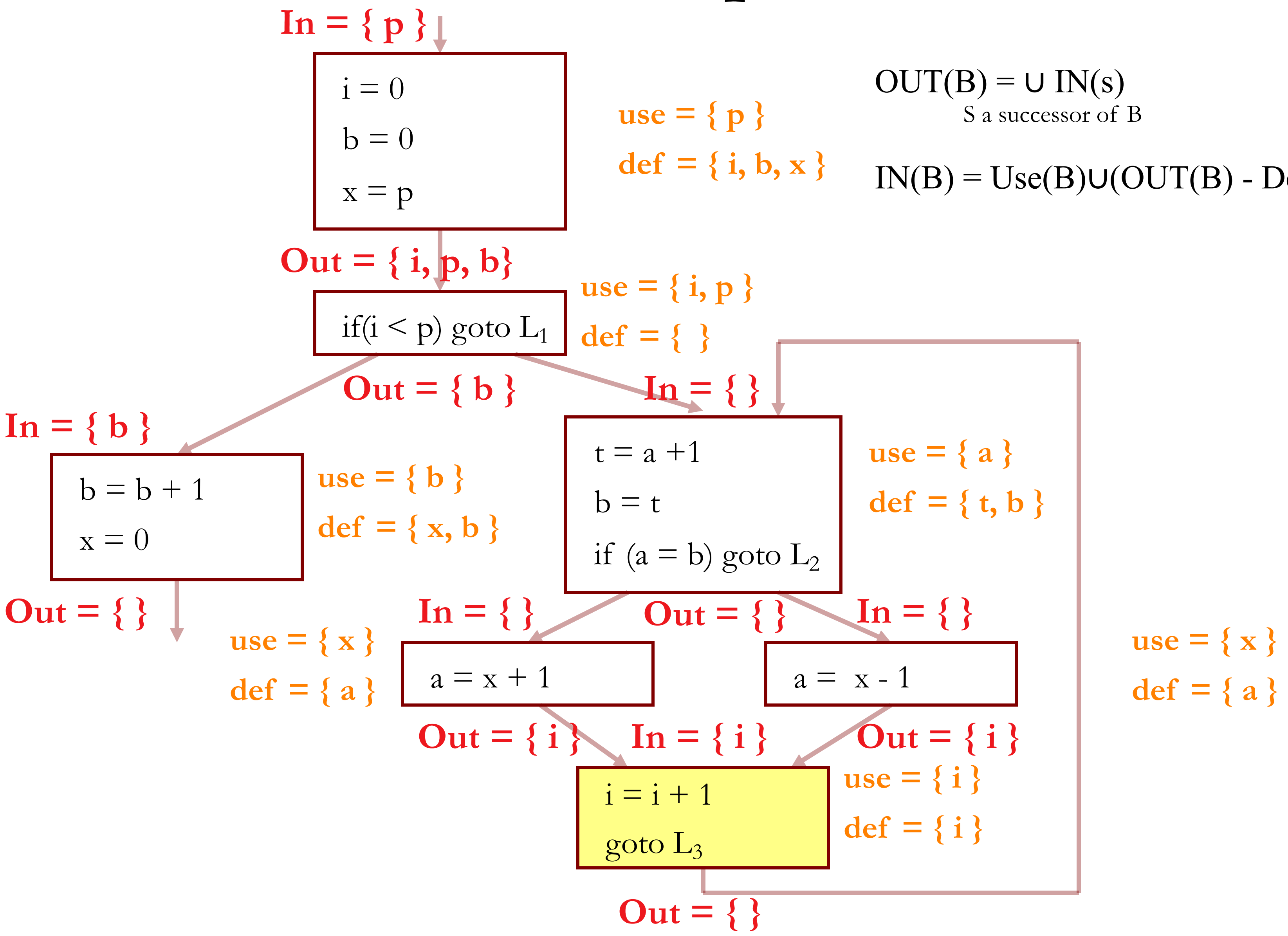
Example



$$OUT(B) = \bigcup_{S \text{ a successor of } B} IN(s)$$

$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$

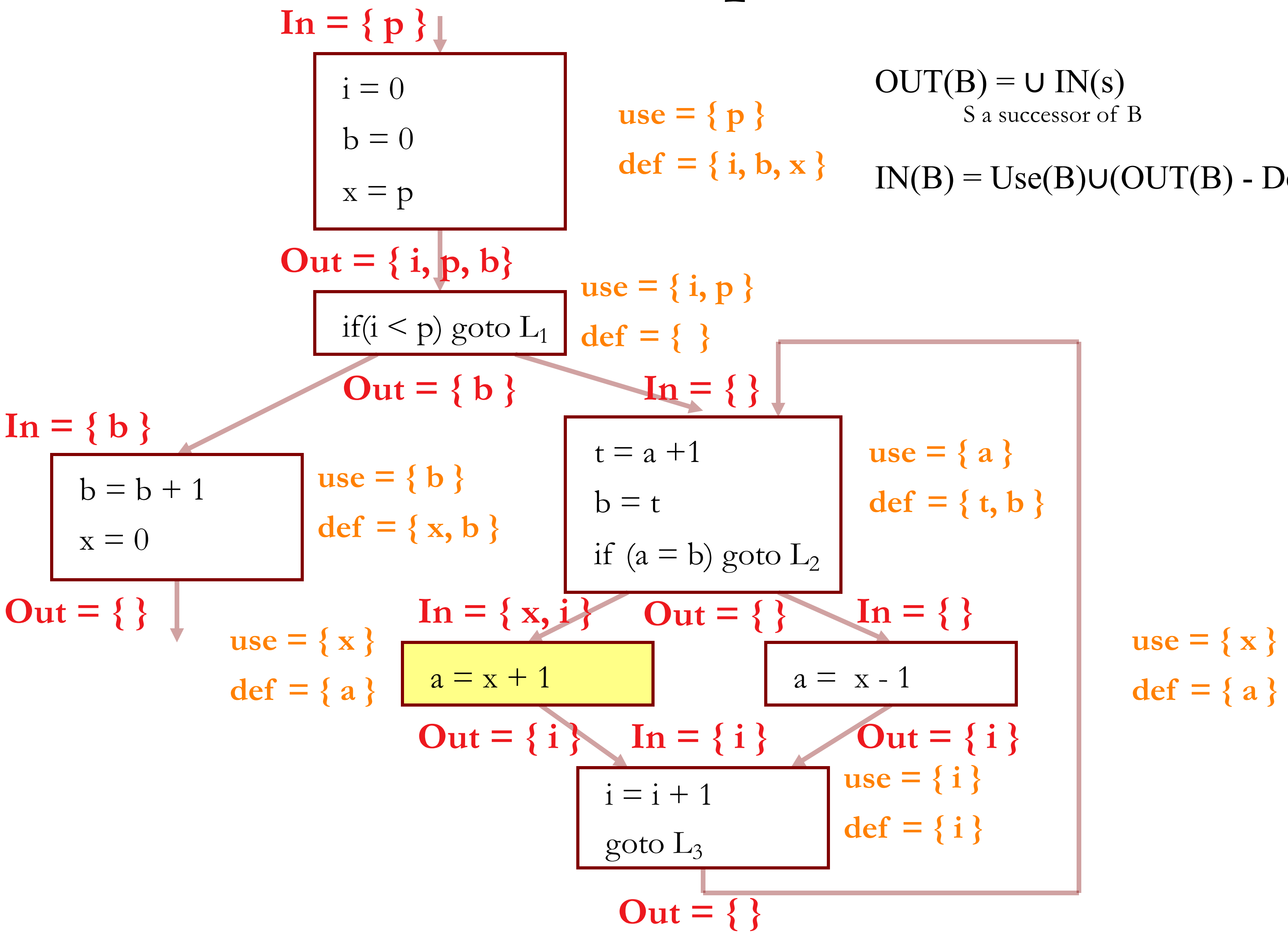
Example



$$OUT(B) = \bigcup_{S \text{ a successor of } B} IN(s)$$

$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$

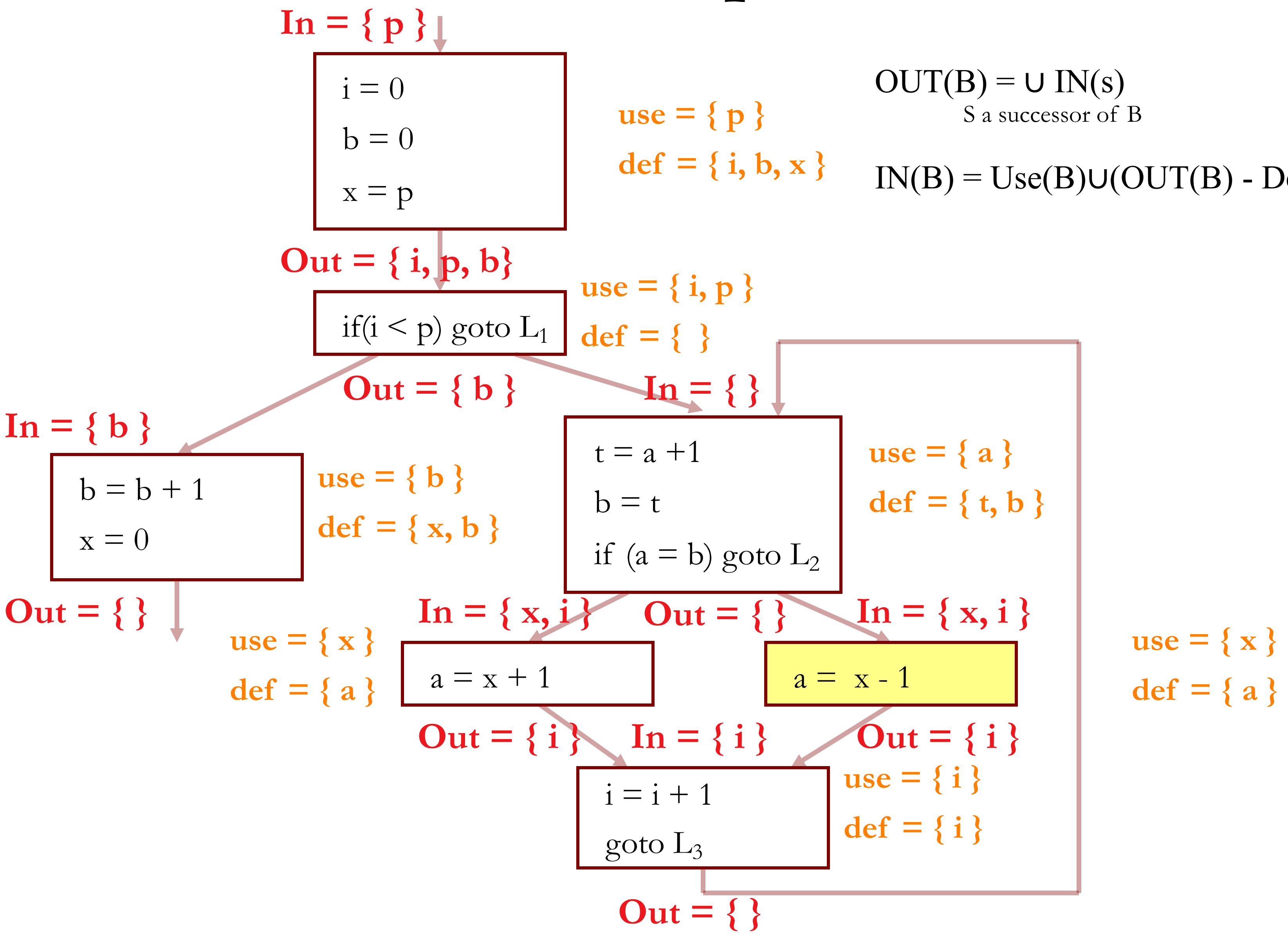
Example



$$OUT(B) = \bigcup_{S \text{ a successor of } B} IN(s)$$

$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$

Example



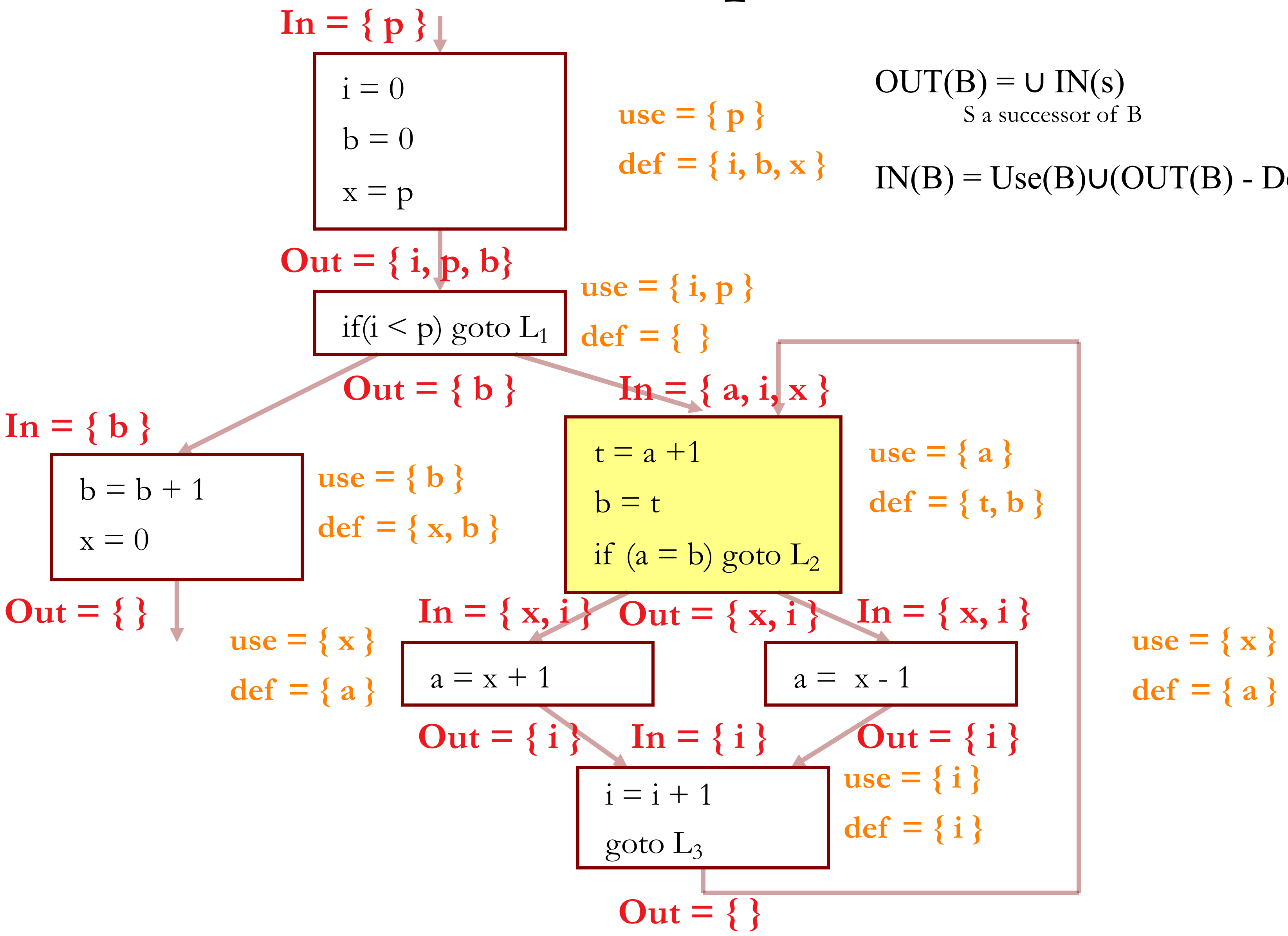
$$OUT(B) = \underset{\text{S a successor of B}}{\cup} IN(s)$$

$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$

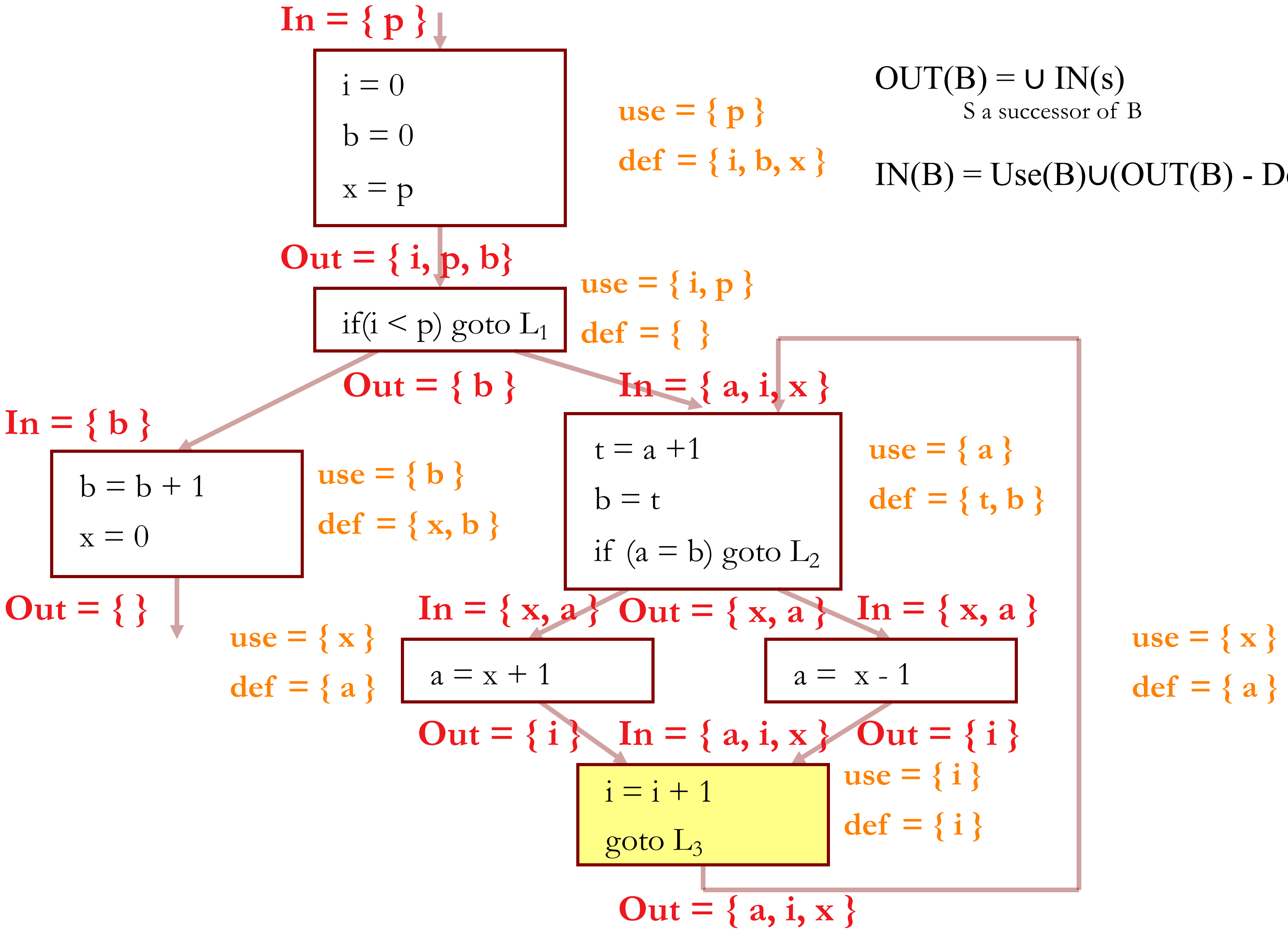
Example

$$OUT(B) = \bigcup_{S \text{ a successor of } B} IN(s)$$

$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$



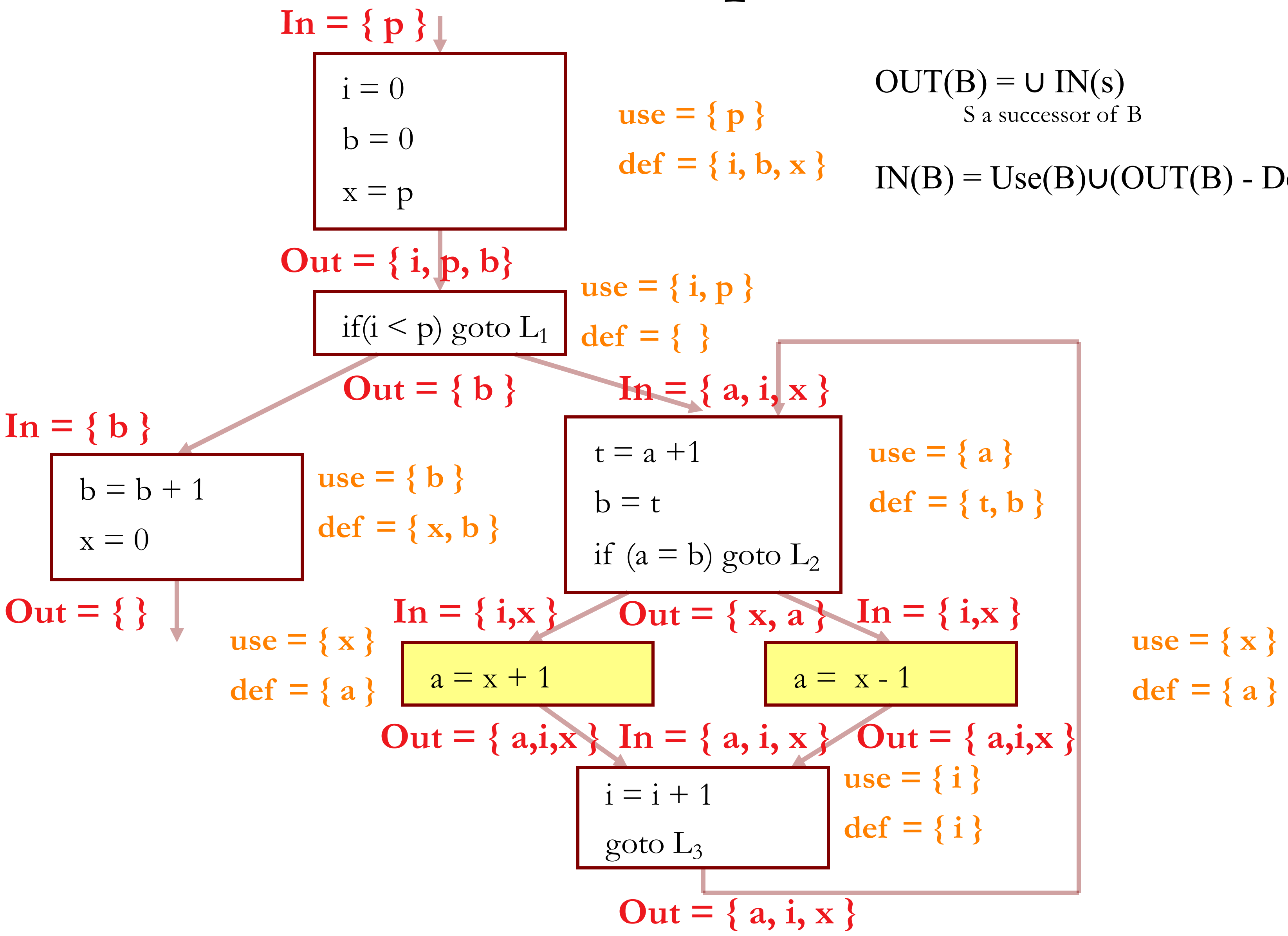
Example



$$OUT(B) = \bigcup_{S \text{ a successor of } B} IN(s)$$

$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$

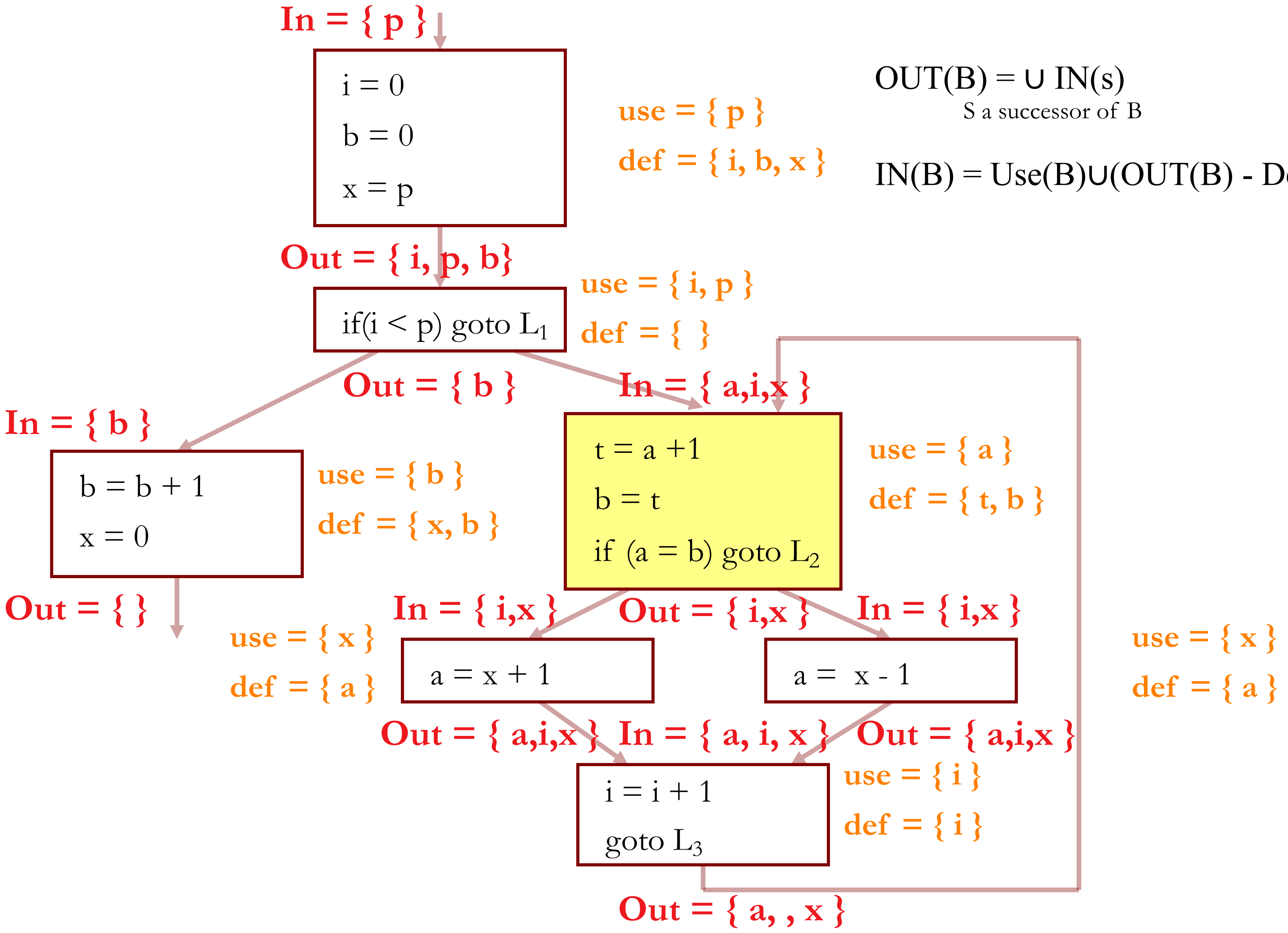
Example



$$OUT(B) = \bigcup_{S \text{ a successor of } B} IN(s)$$

$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$

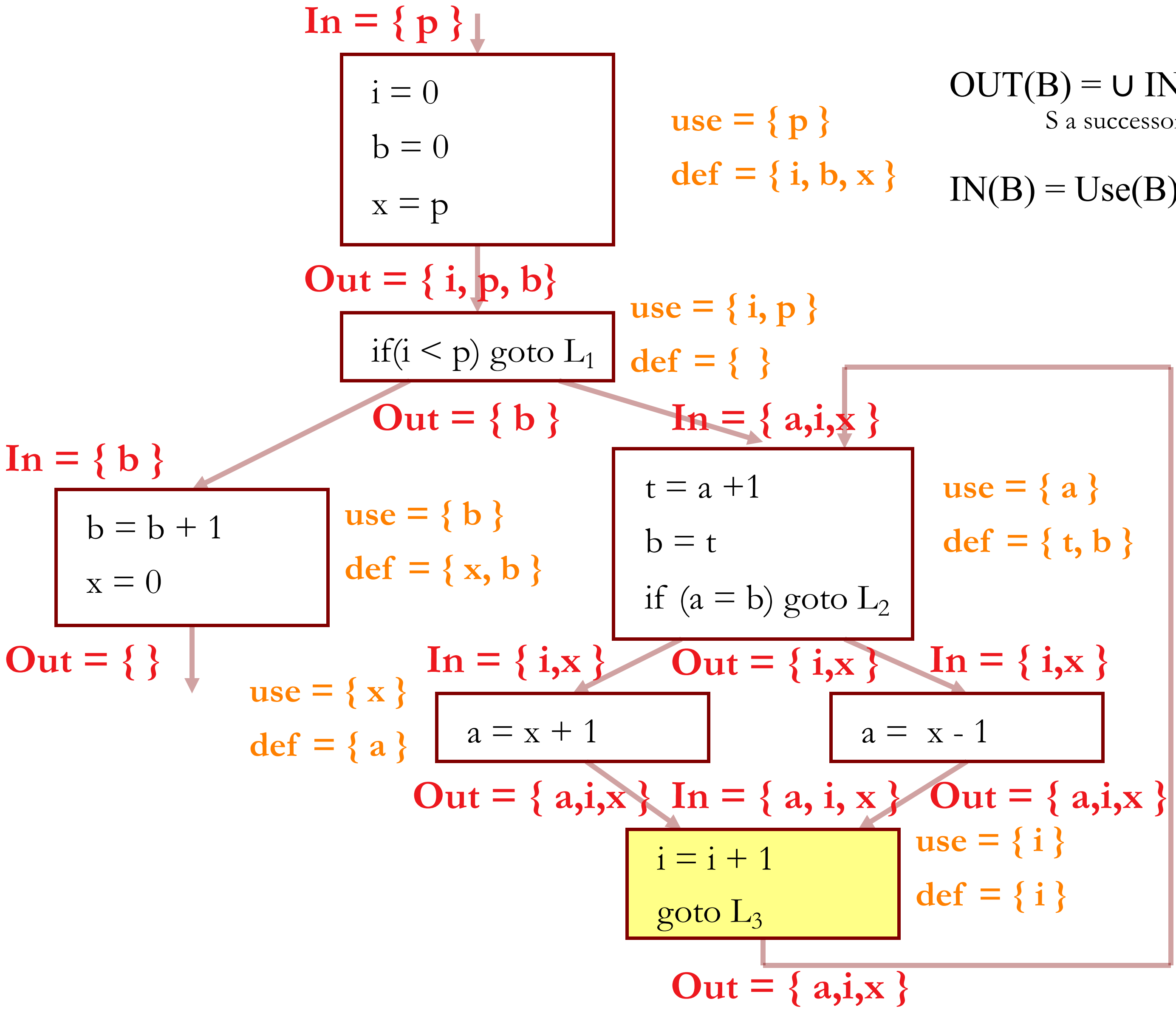
Example



$$OUT(B) = \bigcup_{S \text{ a successor of } B} IN(s)$$

$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$

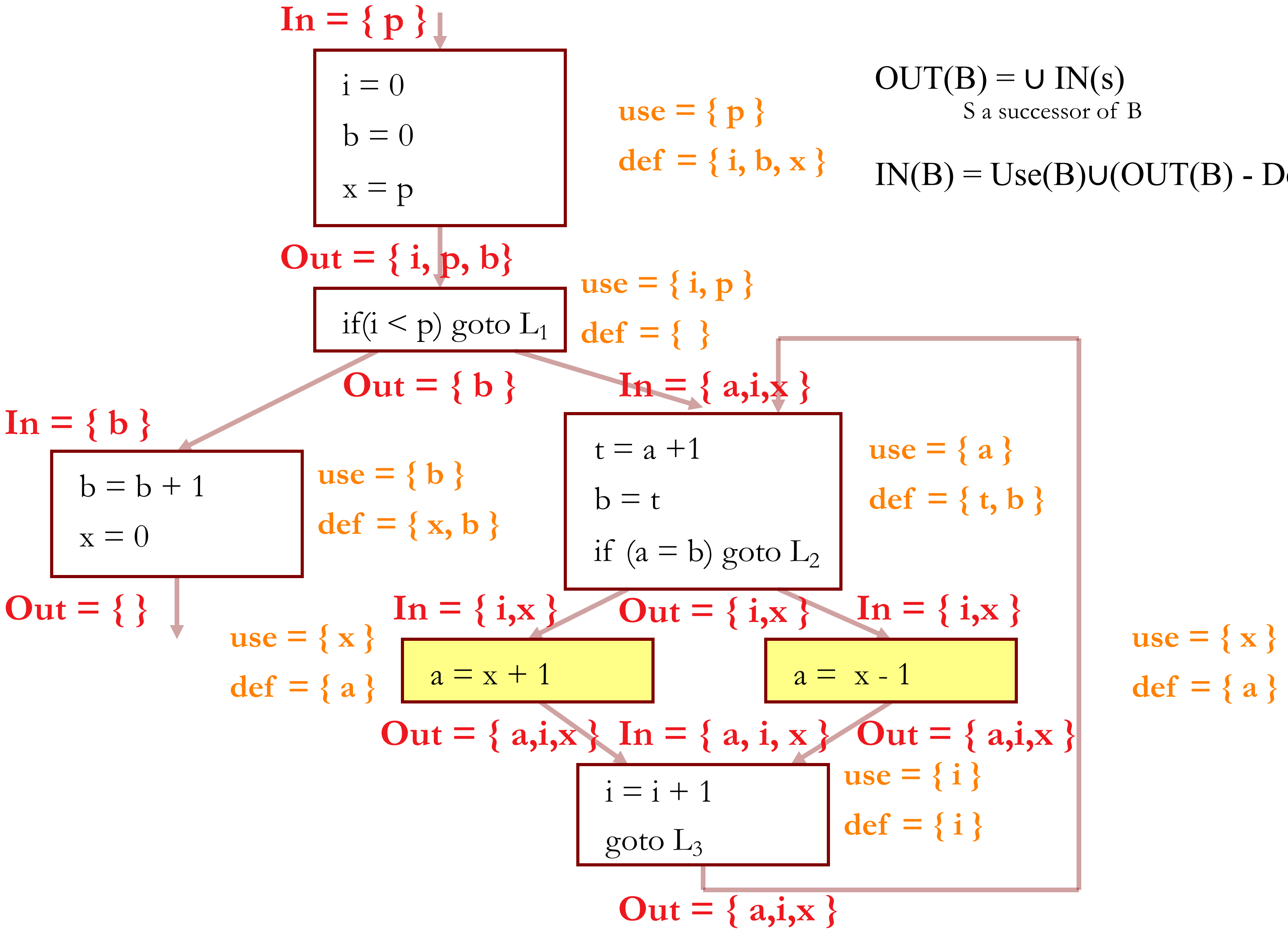
Example



$$OUT(B) = \bigcup_{S \text{ a successor of } B} IN(s)$$

$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$

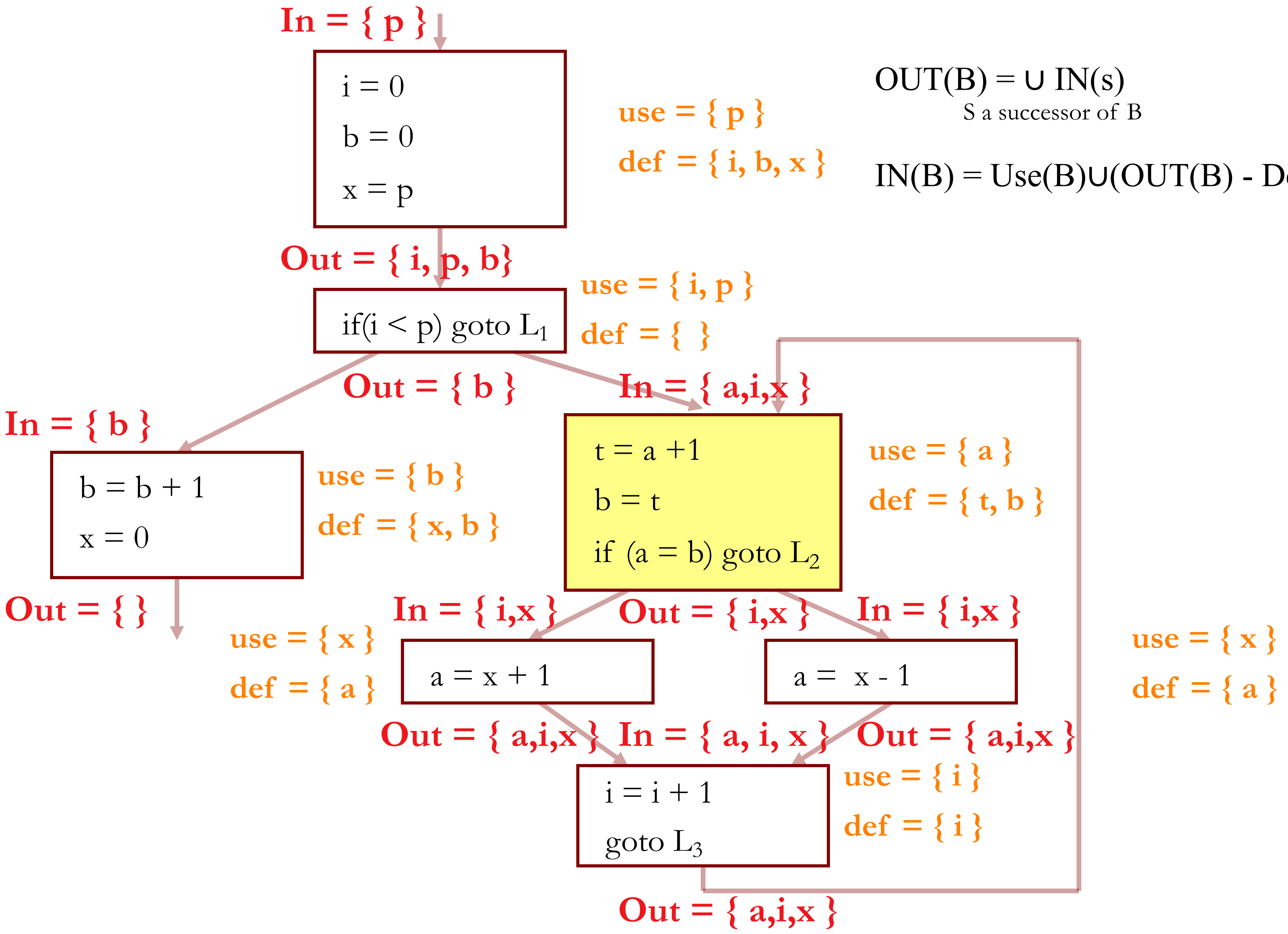
Example



$$OUT(B) = \bigcup_{S \text{ a successor of } B} IN(s)$$

$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$

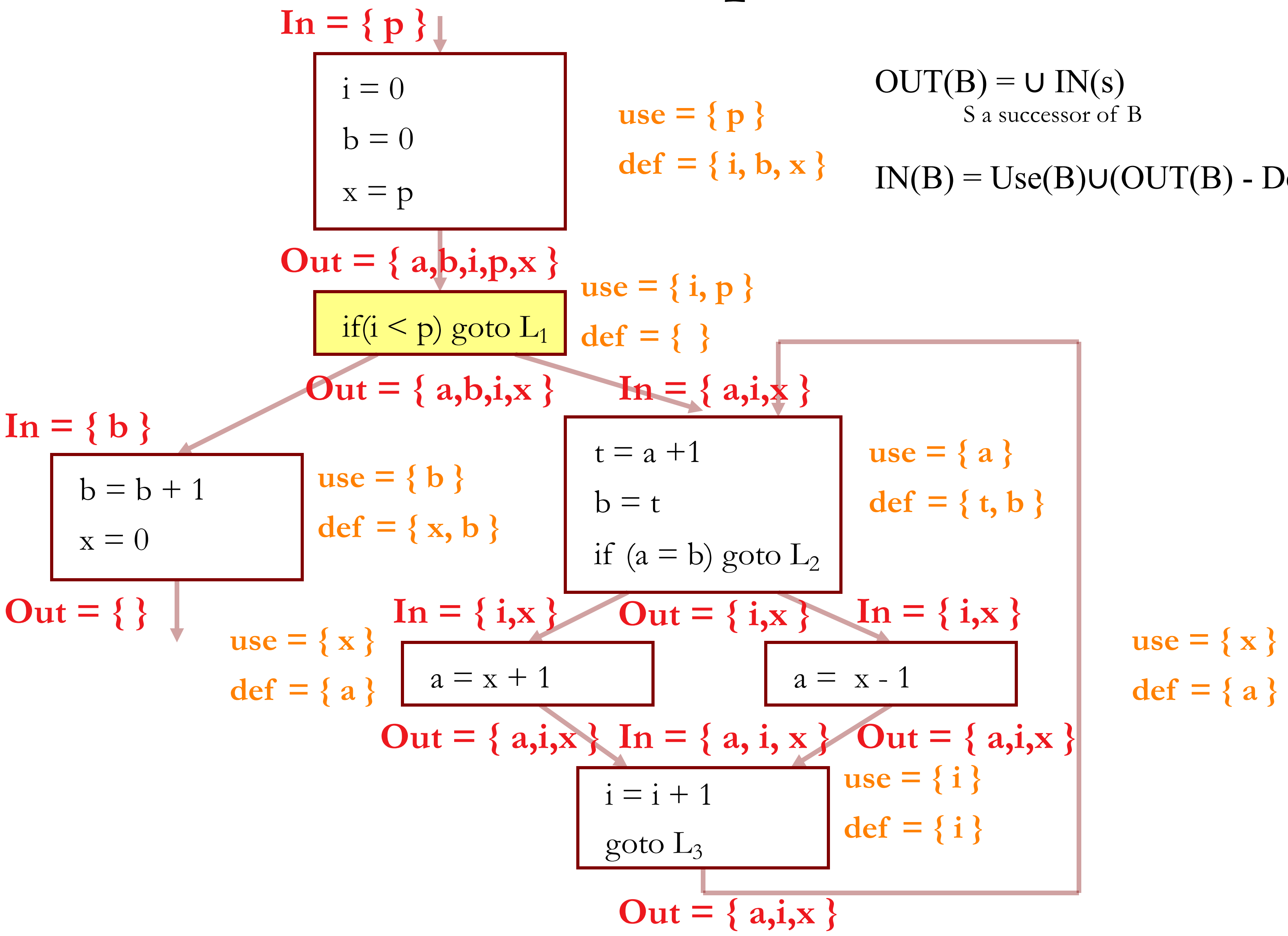
Example



$$OUT(B) = \bigcup_{S \text{ a successor of } B} IN(s)$$

$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$

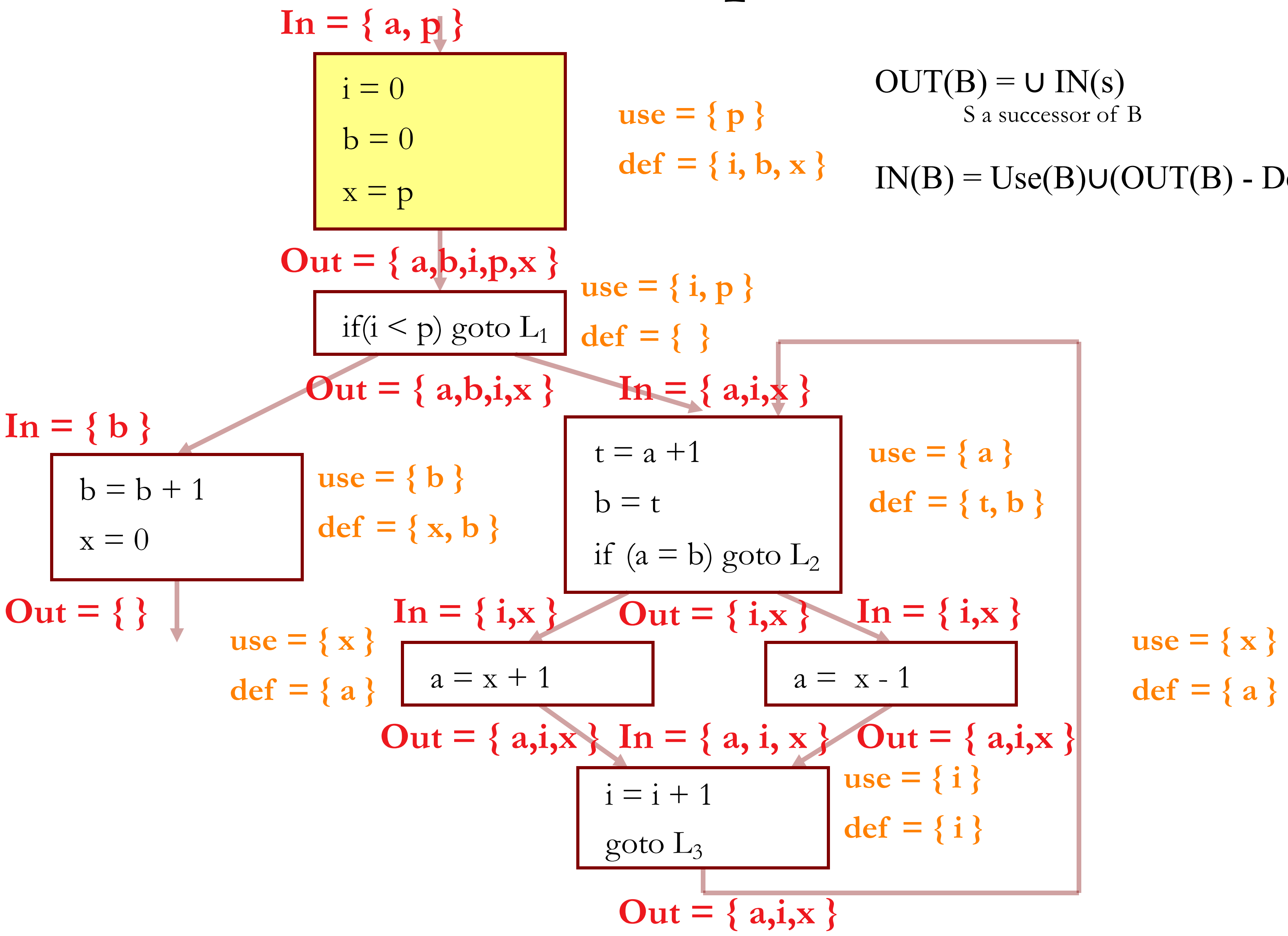
Example



$$OUT(B) = \bigcup_{S \text{ a successor of } B} IN(s)$$

$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$

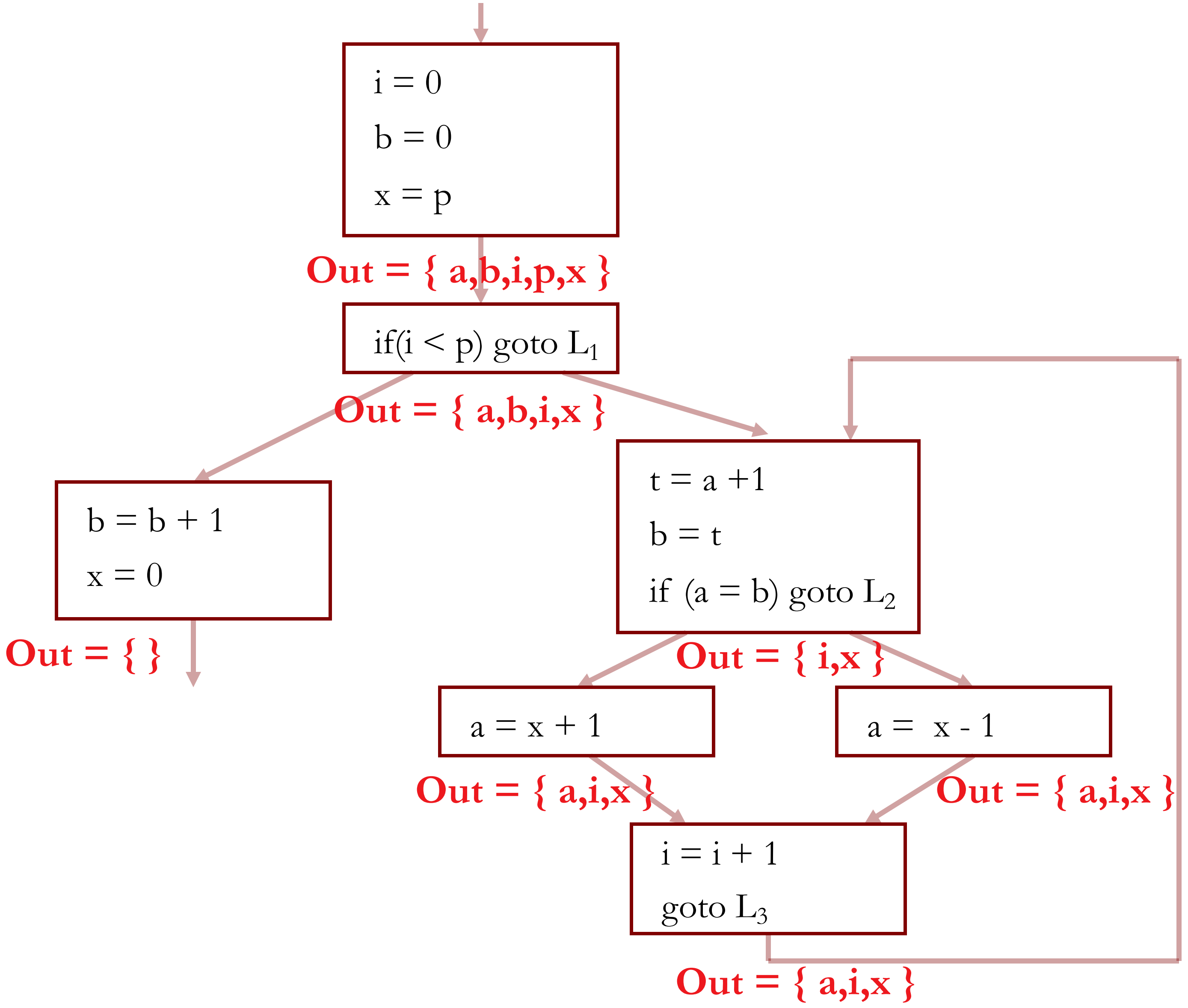
Example



$$OUT(B) = \bigcup_{S \text{ a successor of } B} IN(s)$$

$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$

Example



i = 0
b = 0
x = p
Out = { a,b,i,p,x }
if(i < p) goto L_1
Out = { a,b,i,x }
b = b + 1
x = 0
Out = { }
t = a +1
b = t
if (a = b) goto L_2
Out = { i,x }
a = x + 1
a = x - 1
Out = { a,i,x }
Out = { a,i,x }
i = i + 1
goto L_3
Out = { a,i,x }

Summary

- What is Live-Variable Analysis?
  - Backward Data-Flow Analysis Problem
  - Upwards-Exposed (Gen): Forward Pass computation
- Most Significant Application
  - Register Allocation