

Compilers

Design and Implementation

Data-Flow Analysis

Live-Variable Analysis

Copyright 2023, Pedro C. Diniz, all rights reserved.

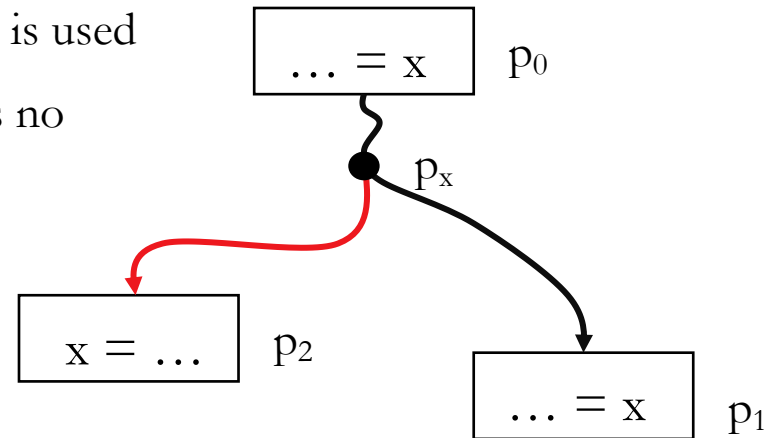
Students enrolled in the Compilers class at Faculdade de Engenharia da Universidade do Porto (FEUP) have explicit permission to make copies of these materials for their personal use.

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

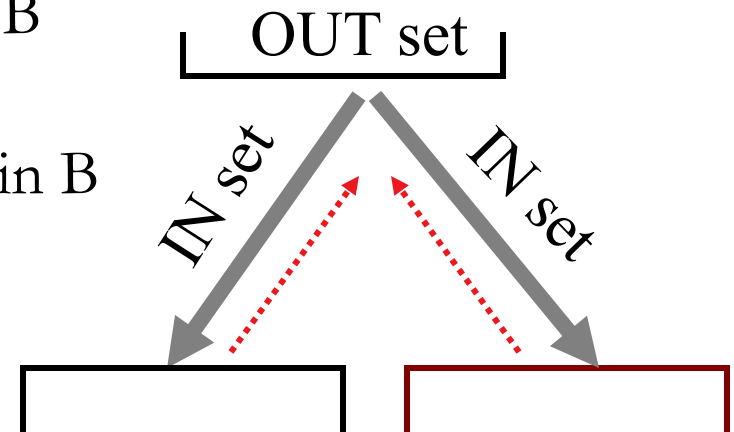


- Need to observe for each variable and for each program point:
 - Where is the last program point beyond which the value is not used
 - Trace back from uses to definitions and observe the first definition (backwards) that reaches that use.
 - That definition kills all uses backwards of it.

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

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

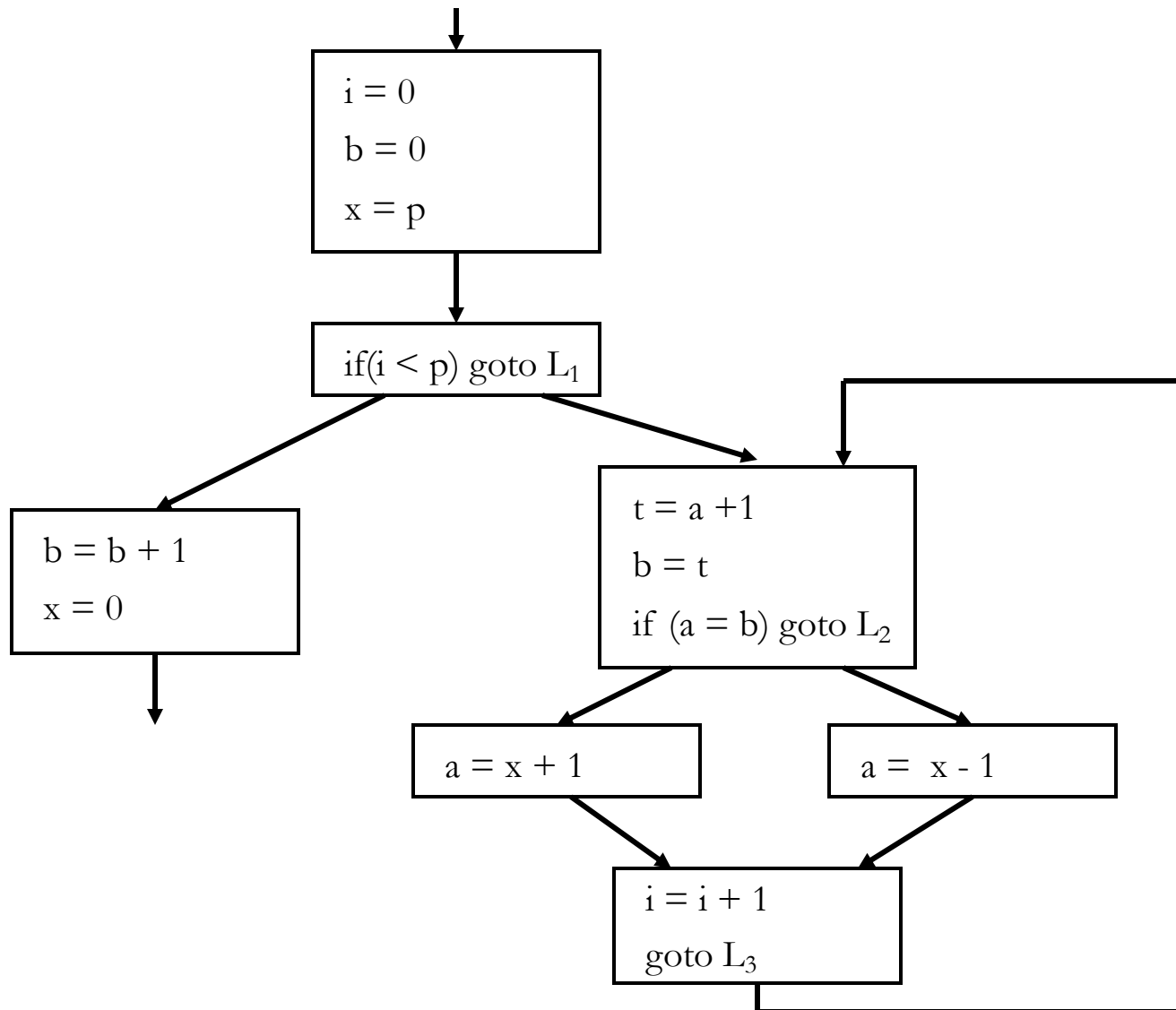


$$\text{IN}(B) = \text{Use}(B) \cup (\text{OUT}(B) - \text{Def}(B))$$

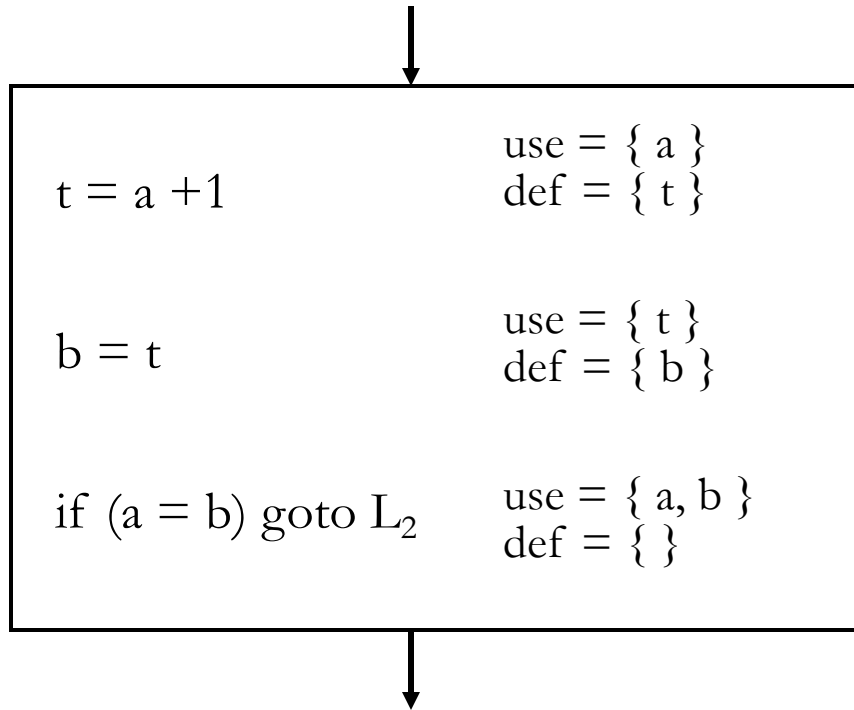
Data-Flow Analysis Formulation

- 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



Use & Def Functions for a Basic Block

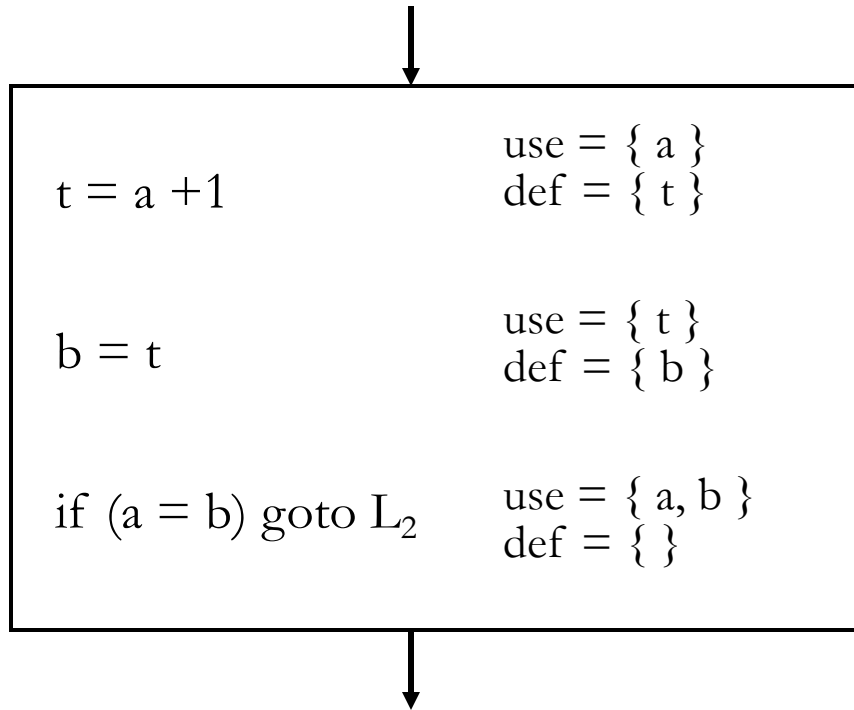


$$\text{In} = \text{Use} \cup (\text{Out} - \text{Def})$$

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

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

Use & Def Functions for a Basic Block

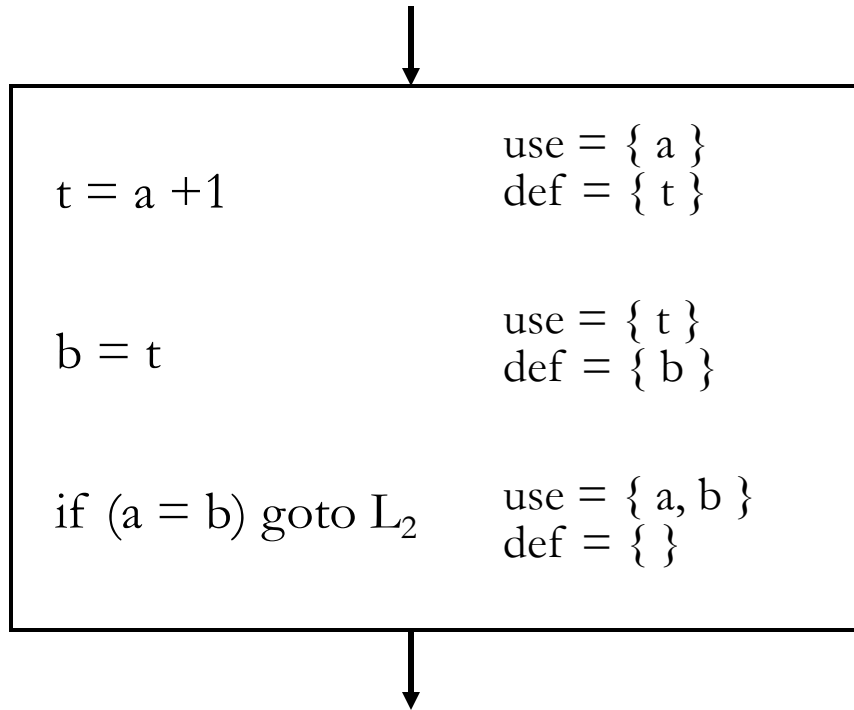


$$In = \{a,b\} \cup (\{\} - \{\}) = \{a,b\}$$

$$Out = \{ \}$$

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

Use & Def Functions for a Basic Block



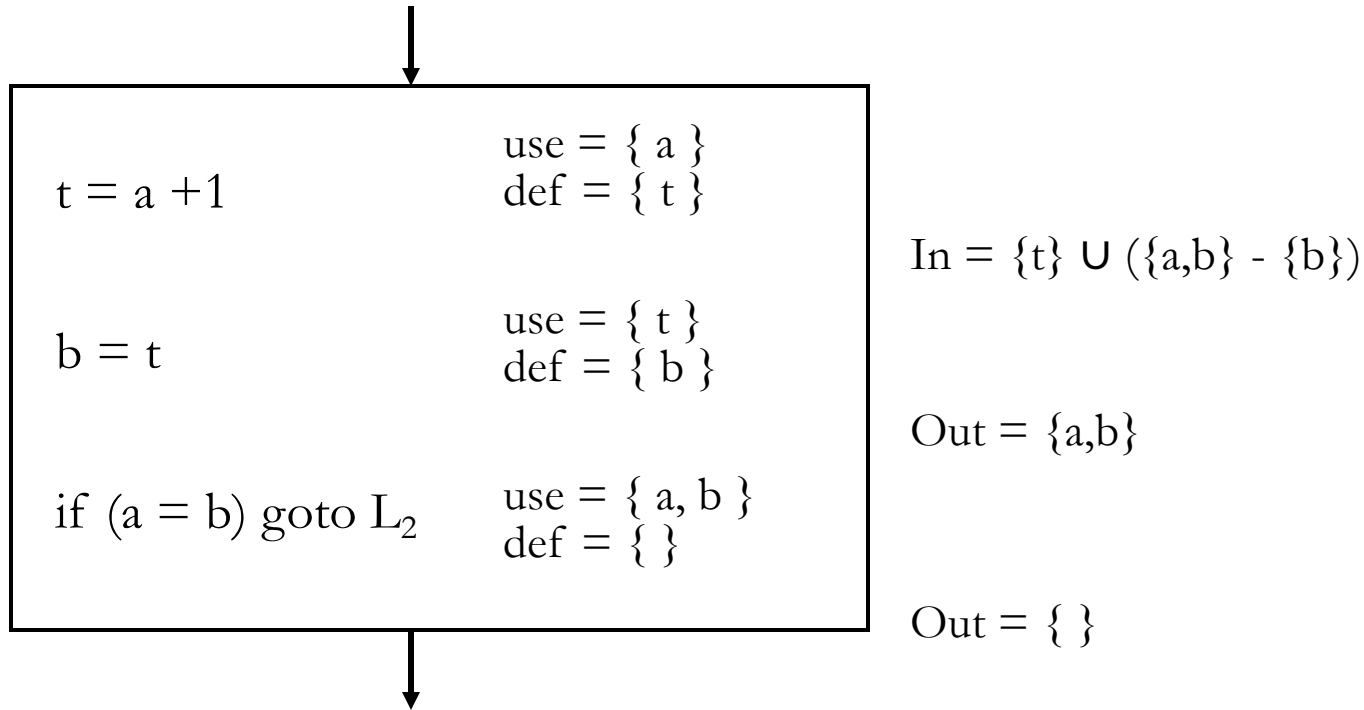
$$In = Use \cup (Out - Def)$$

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

$$Out = \{ \}$$

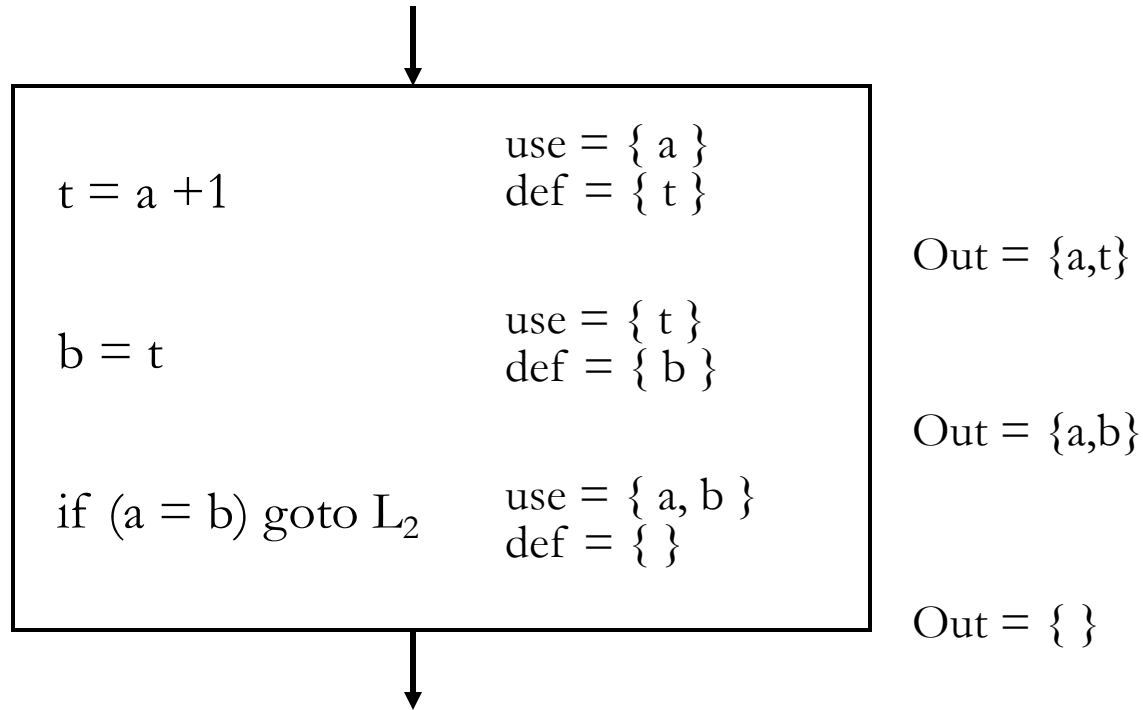
$$In(i) = Use(i) \cup (Out(i) - Def(i))$$

Use & Def Functions for a Basic Block



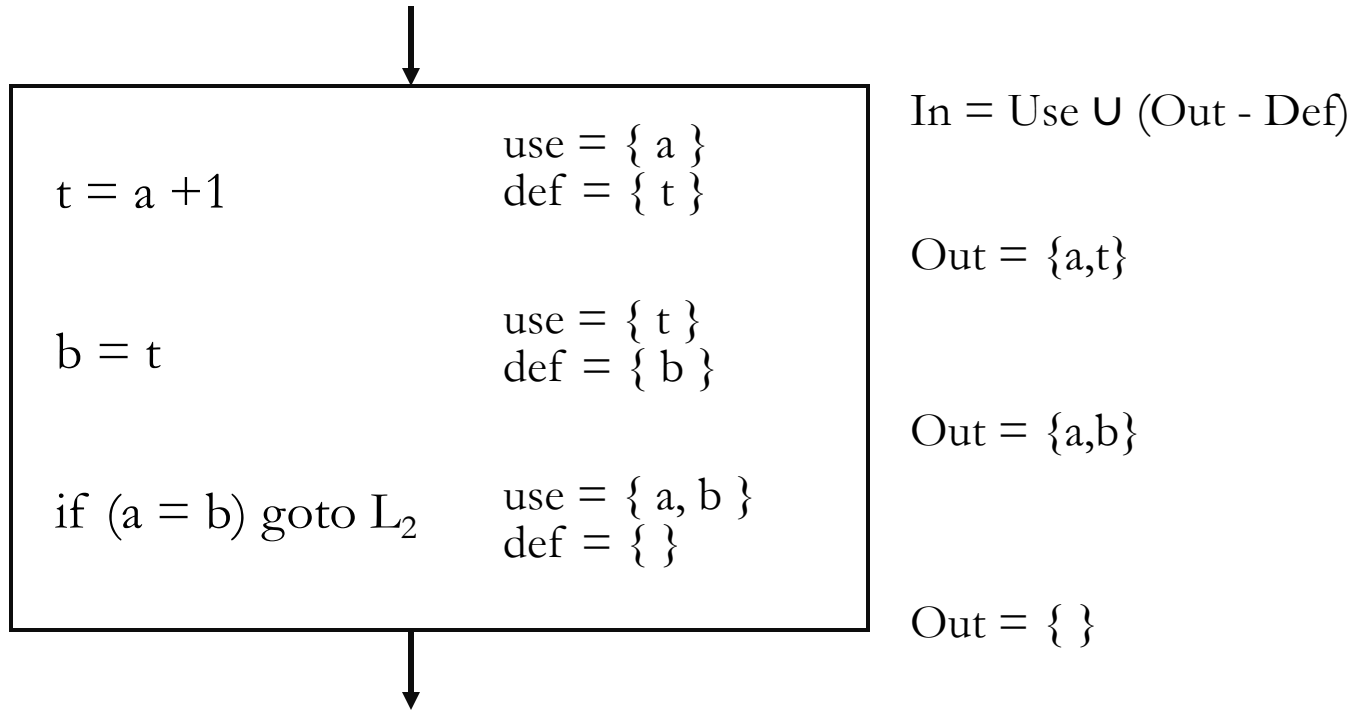
$$\text{In}(i) = \text{Use}(i) \cup (\text{Out}(i) - \text{Def}(i))$$

Use & Def Functions for a Basic Block



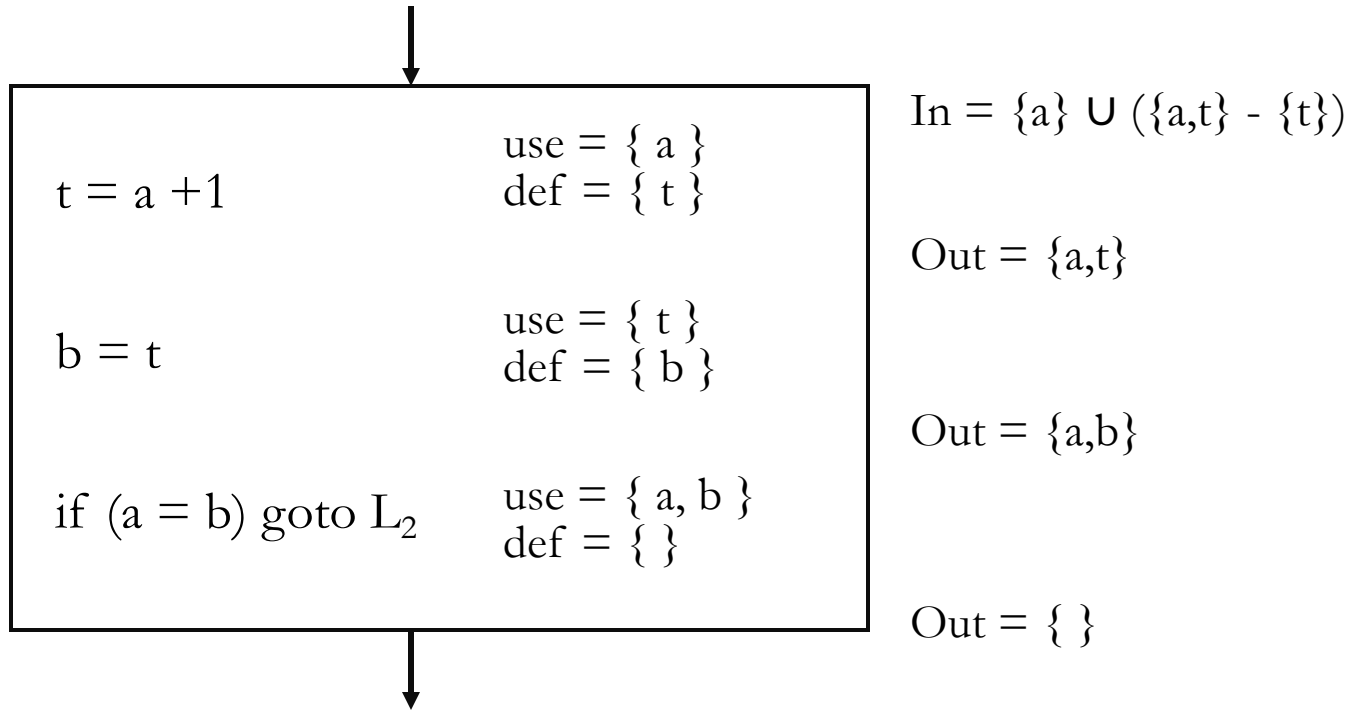
$$In(i) = Use(i) \cup (Out(i) - Def(i))$$

Use & Def Functions for a Basic Block



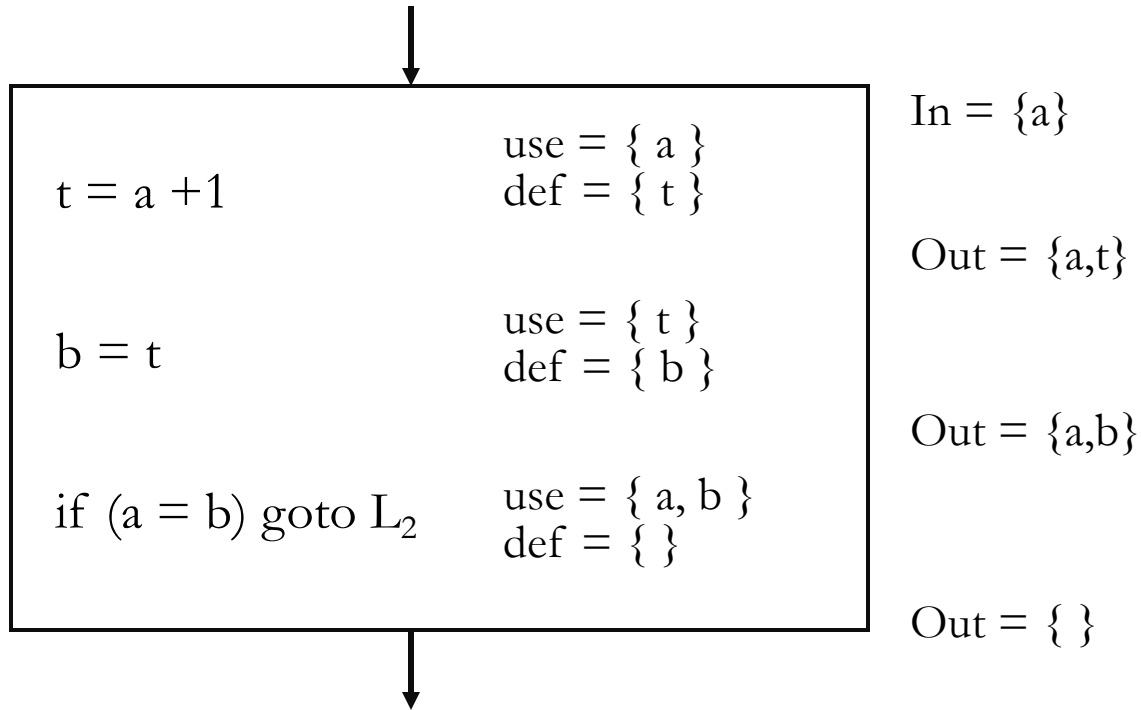
$$\text{In}(i) = \text{Use}(i) \cup (\text{Out}(i) - \text{Def}(i))$$

Use & Def Functions for a Basic Block



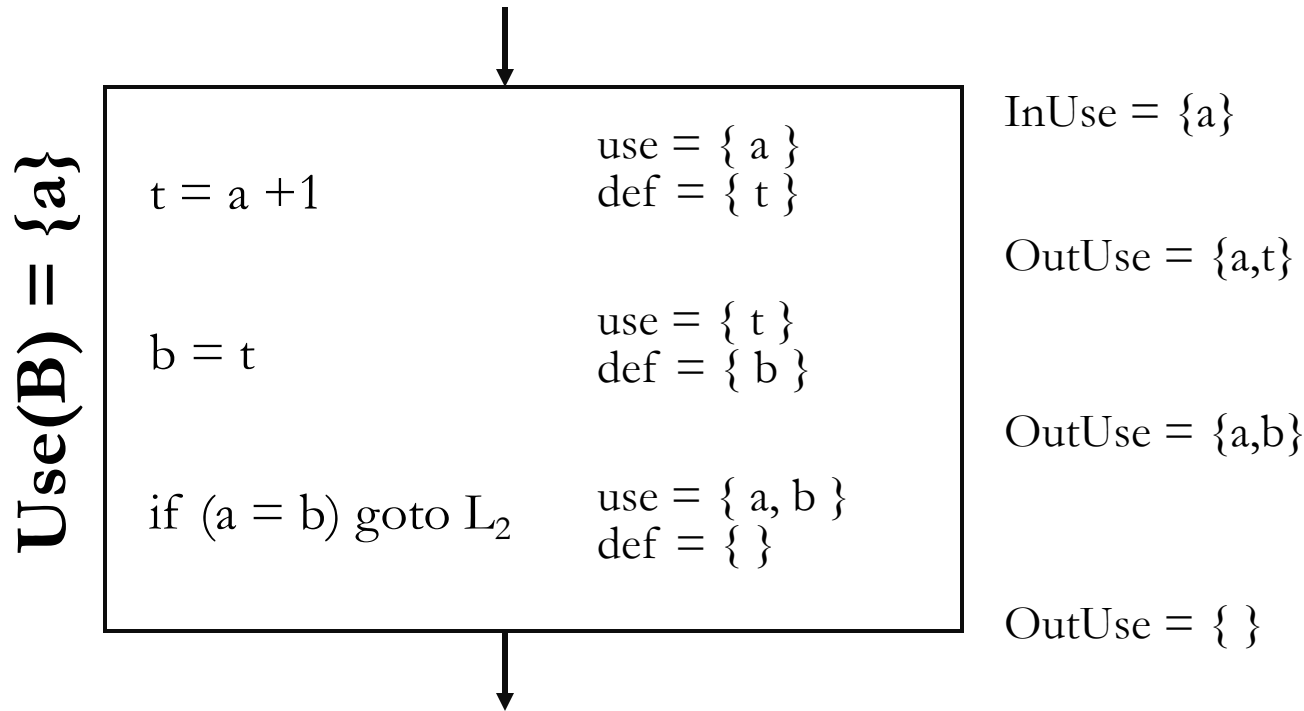
$$In(i) = Use(i) \cup (Out(i) - Def(i))$$

Use & Def Functions for a Basic Block



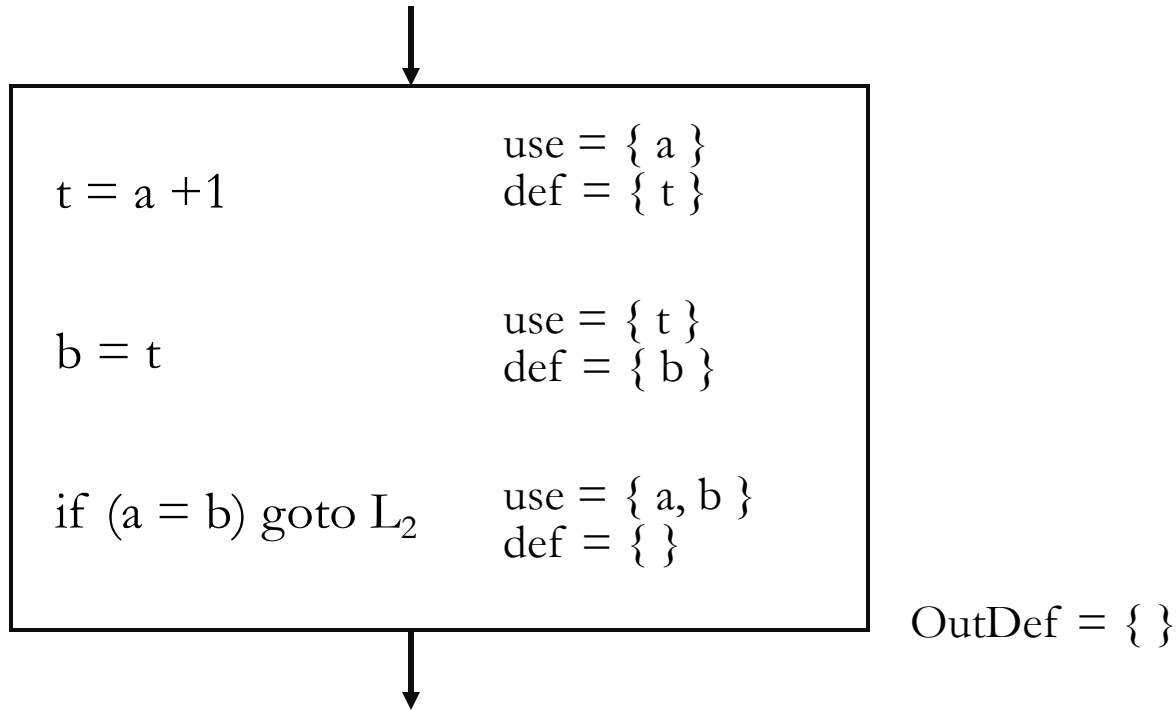
$$\text{In}(i) = \text{Use}(i) \cup (\text{Out}(i) - \text{Def}(i))$$

Use & Def Functions for a Basic Block



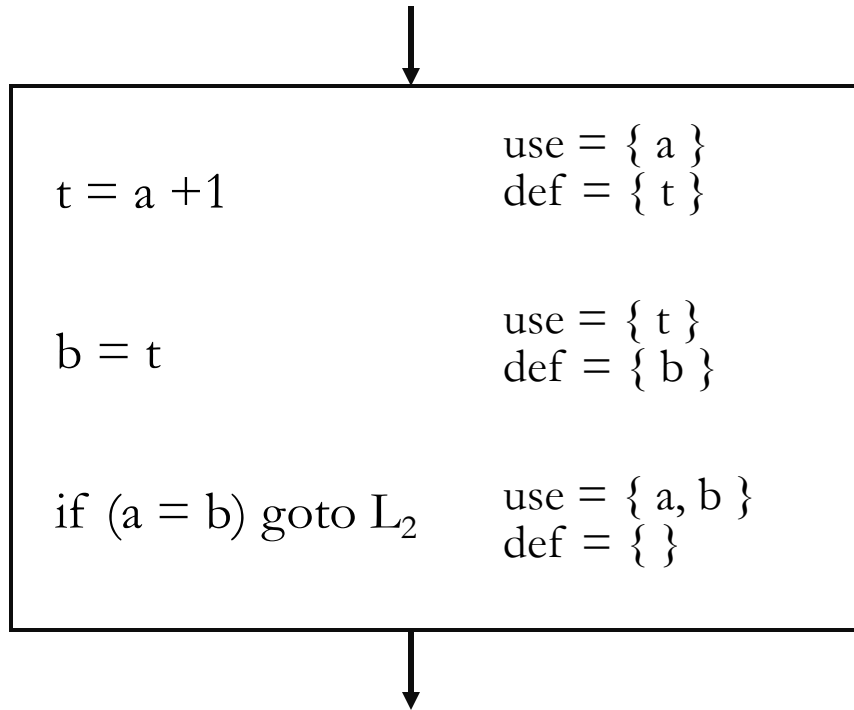
$$InUse(i) = Use(i) \cup (OutUse(i) - Def(i))$$

Use & Def Functions for a Basic Block



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

Use & Def Functions for a Basic Block

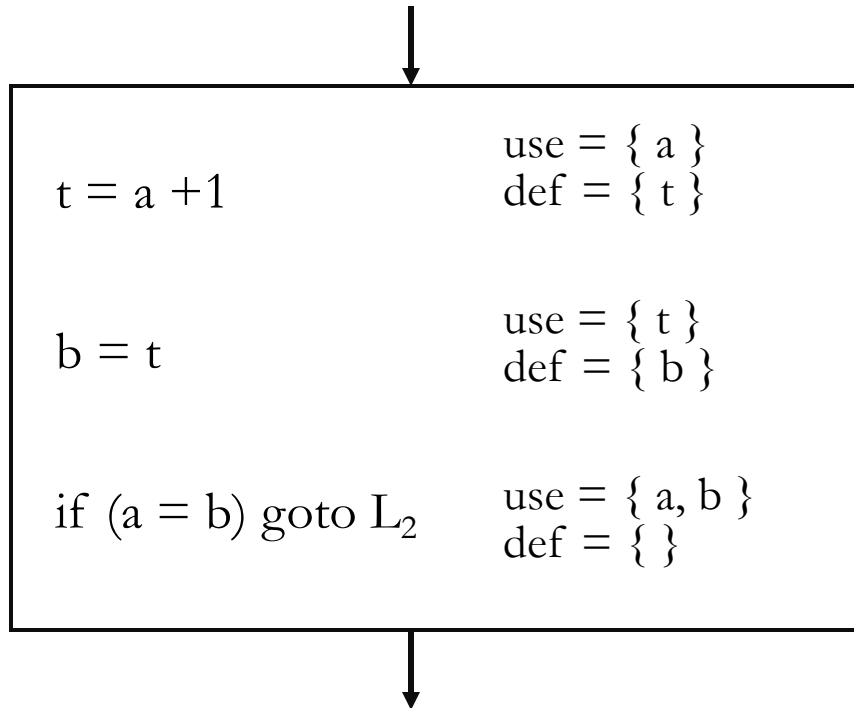


$$\text{InDef} = \text{Def} \cup \text{OutDef} = \{ \}$$

$$\text{OutDef} = \{ \}$$

$$\text{InDef}(i) = \text{Def}(i) \cup \text{OutDef}(i)$$

Use & Def Functions for a Basic Block



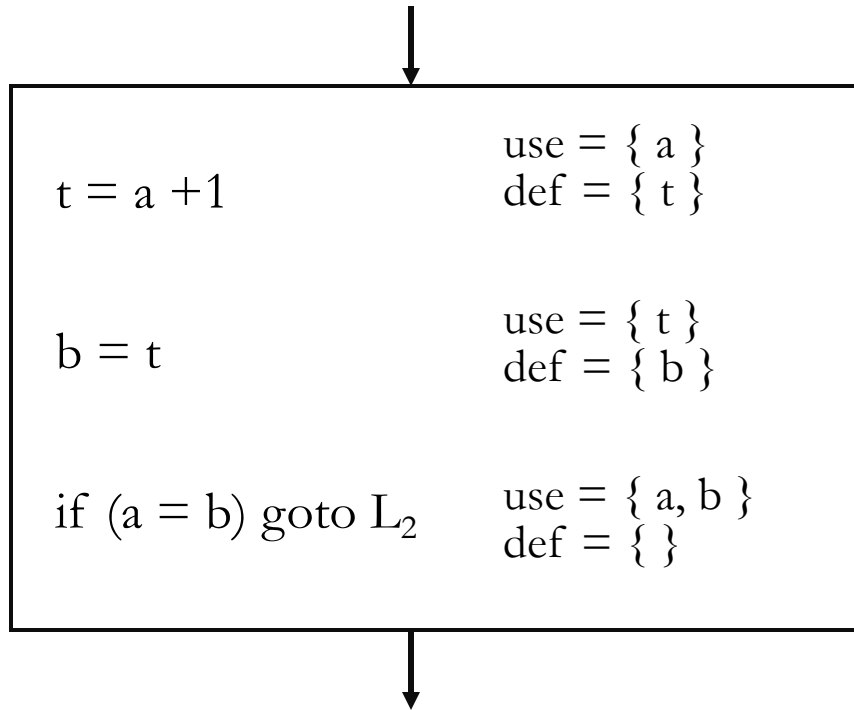
$$\text{InDef} = \text{Def} \cup \text{OutDef} = \{b\}$$

$$\text{OutDef} = \{ \}$$

$$\text{OutDef} = \{ \}$$

$$\text{InDef}(i) = \text{Def}(i) \cup \text{OutDef}(i)$$

Use & Def Functions for a Basic Block



$$\text{InDef} = \text{Def} \cup \text{OutDef} = \{t\} \cup \{b\}$$

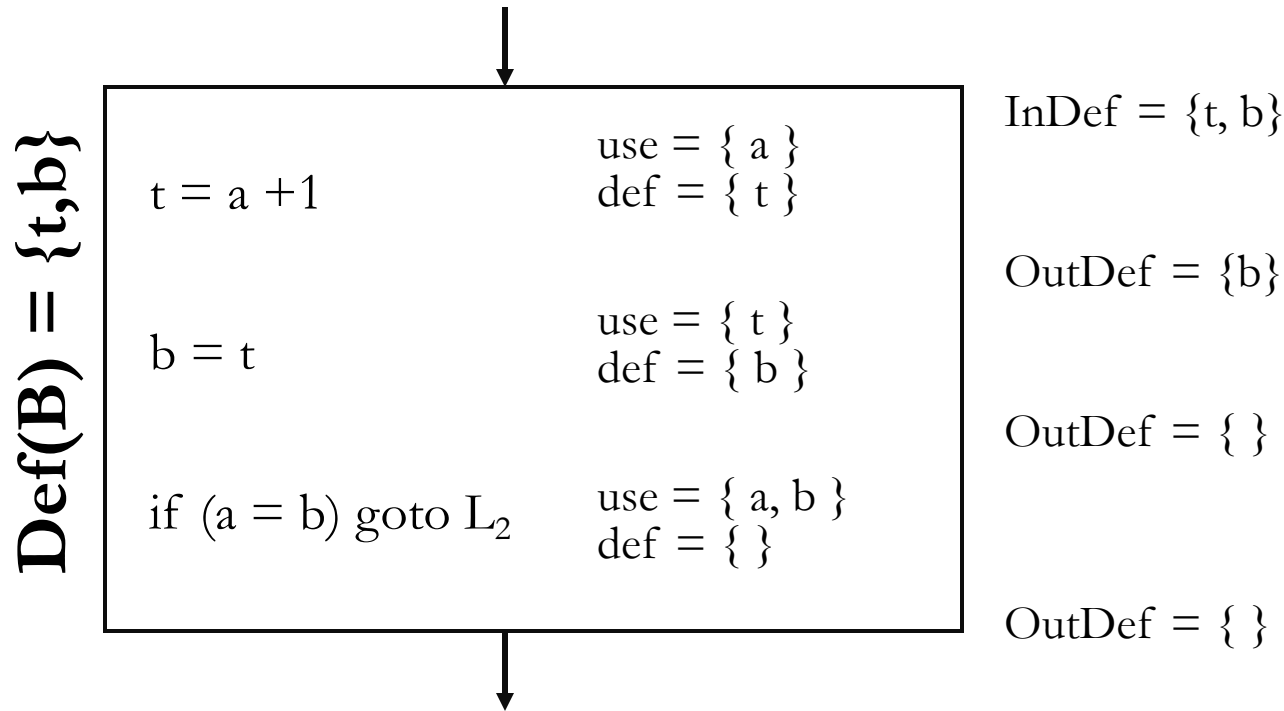
$$\text{OutDef} = \{b\}$$

$$\text{OutDef} = \{ \}$$

$$\text{OutDef} = \{ \}$$

$$\text{InDef}(i) = \text{Def}(i) \cup \text{OutDef}(i)$$

Use & Def Functions for a Basic Block



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

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 “ $x \leftarrow y \text{ op } z$ ”

for $i \leftarrow 1$ to Num Instructions in B do

 if (instr(i) is leader of B) then

$b \leftarrow \text{Number}(B)$;

$\text{UpExp}(b) \leftarrow \emptyset$;

$\text{VarKill}(b) \leftarrow \emptyset$;

 if $y \notin \text{VarKill}(b)$ then

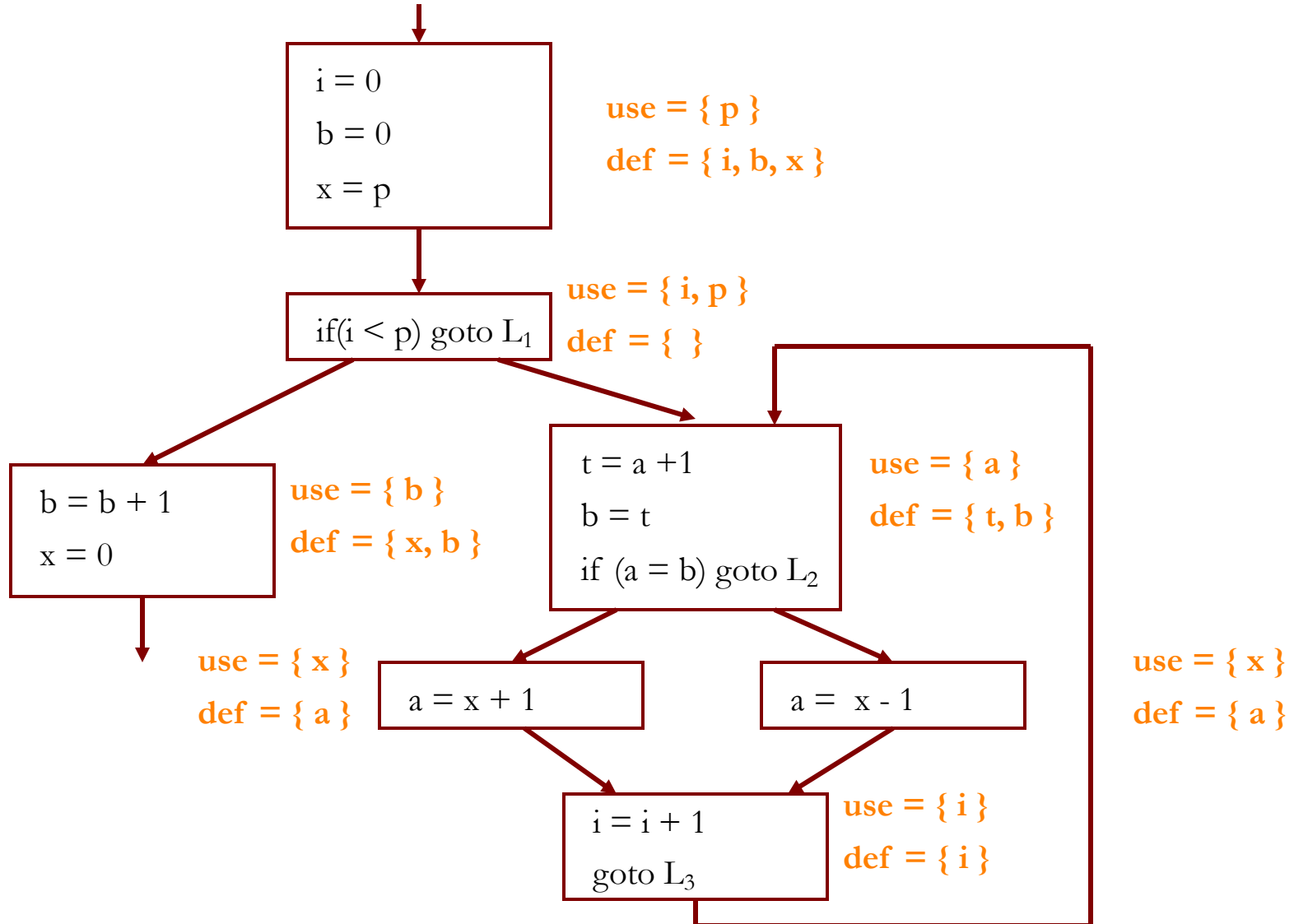
$\text{UpExp}(b) \leftarrow \text{UpExp}(b) \cup \{y\}$

 if $z \notin \text{VarKill}(b)$ then

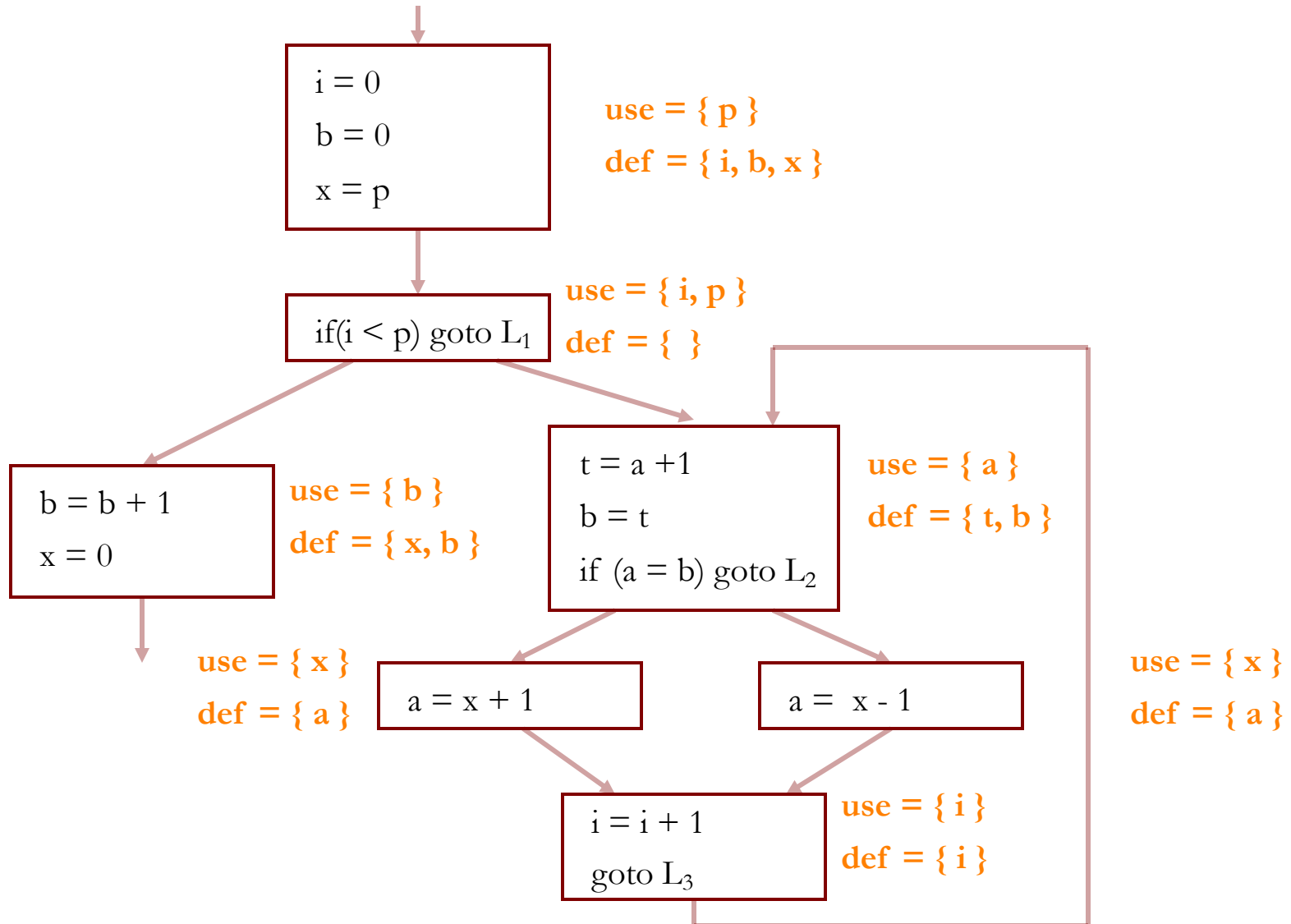
$\text{UpExp}(b) \leftarrow \text{UpExp}(b) \cup \{z\}$

$\text{VarKill}(b) \leftarrow \text{VarKill}(b) \cup \{x\}$

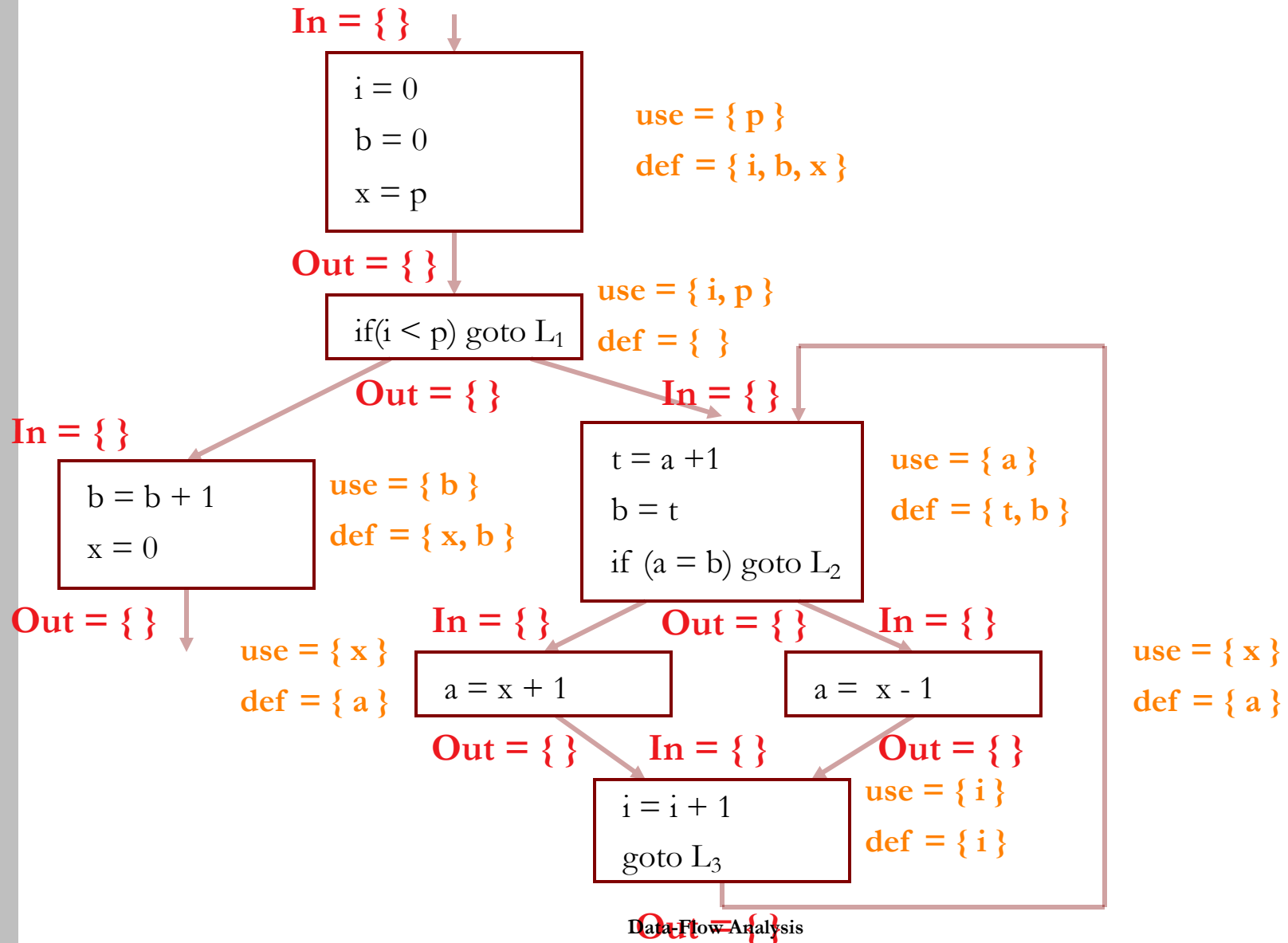
Example



Example



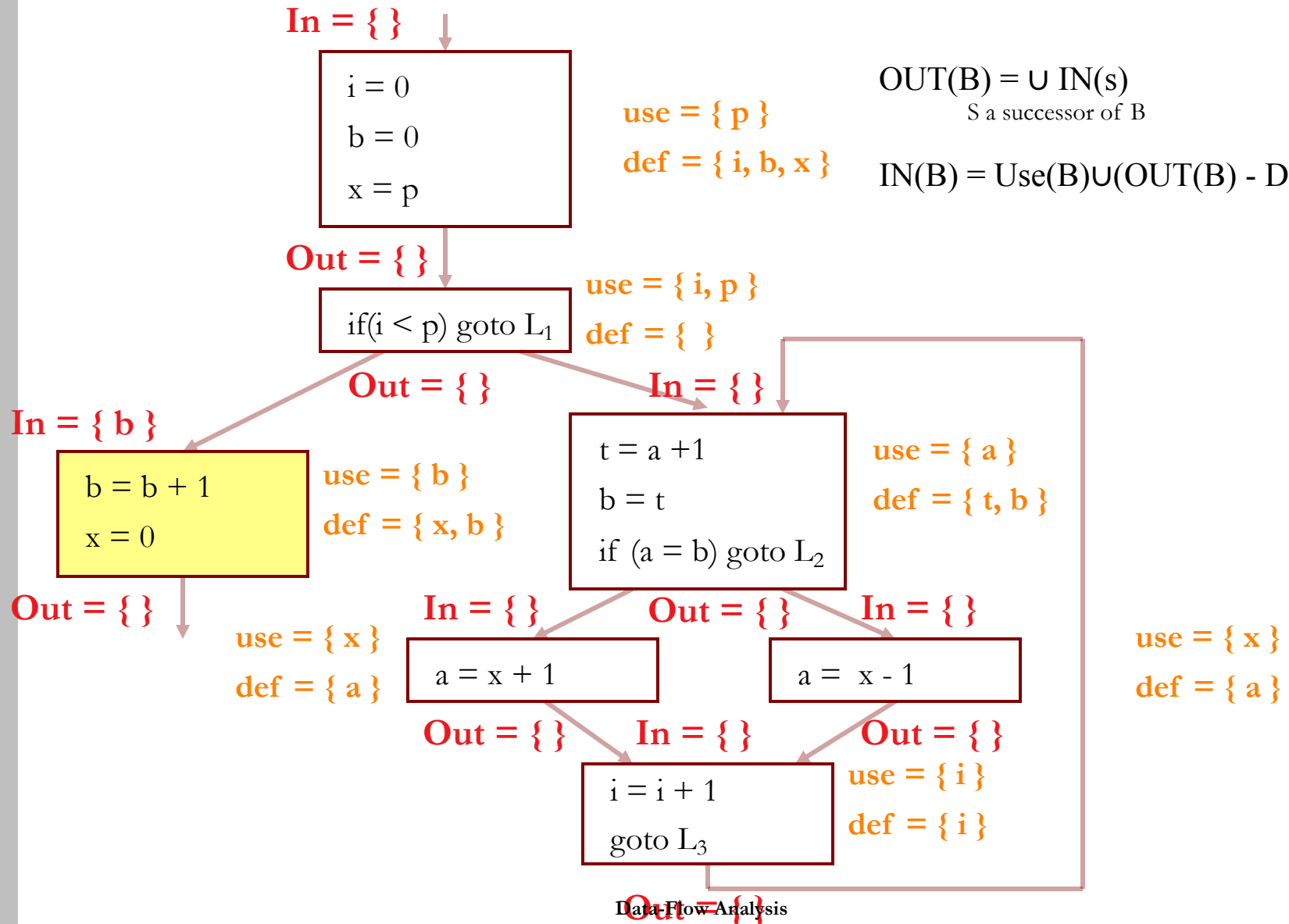
Example



Example

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

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

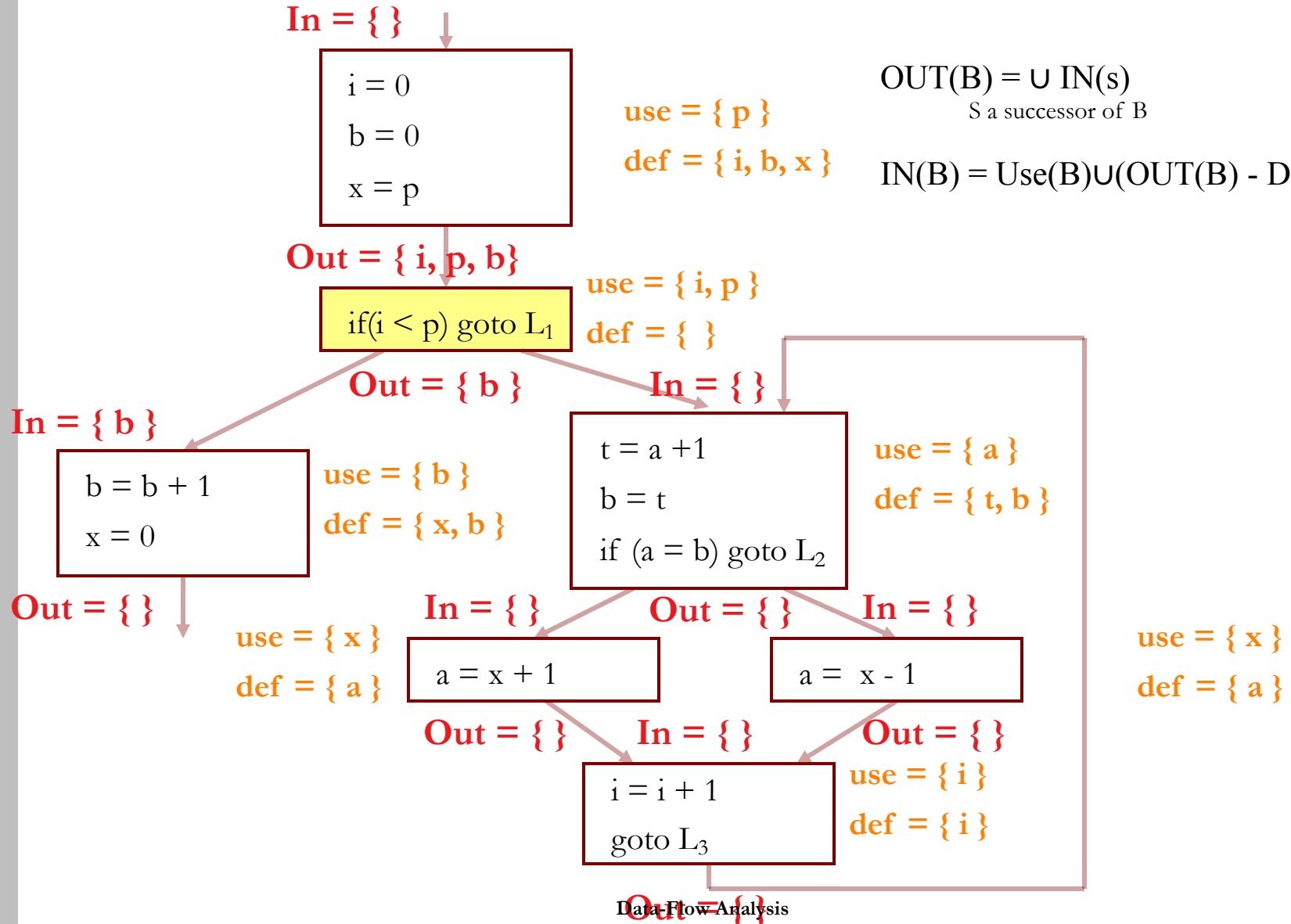


Example

$$OUT(B) = \bigcup IN(s)$$

S a successor of B

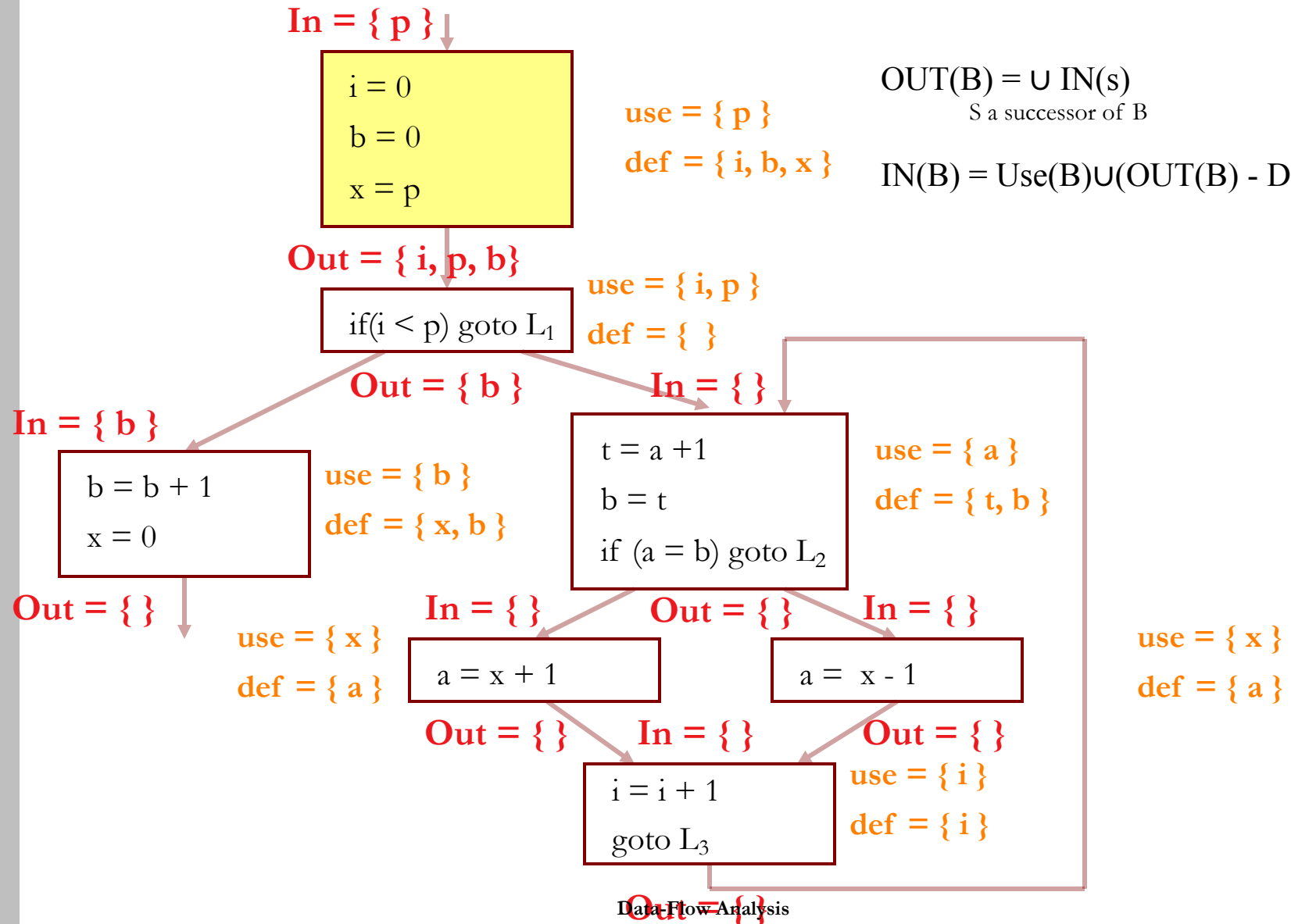
$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$



Example

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

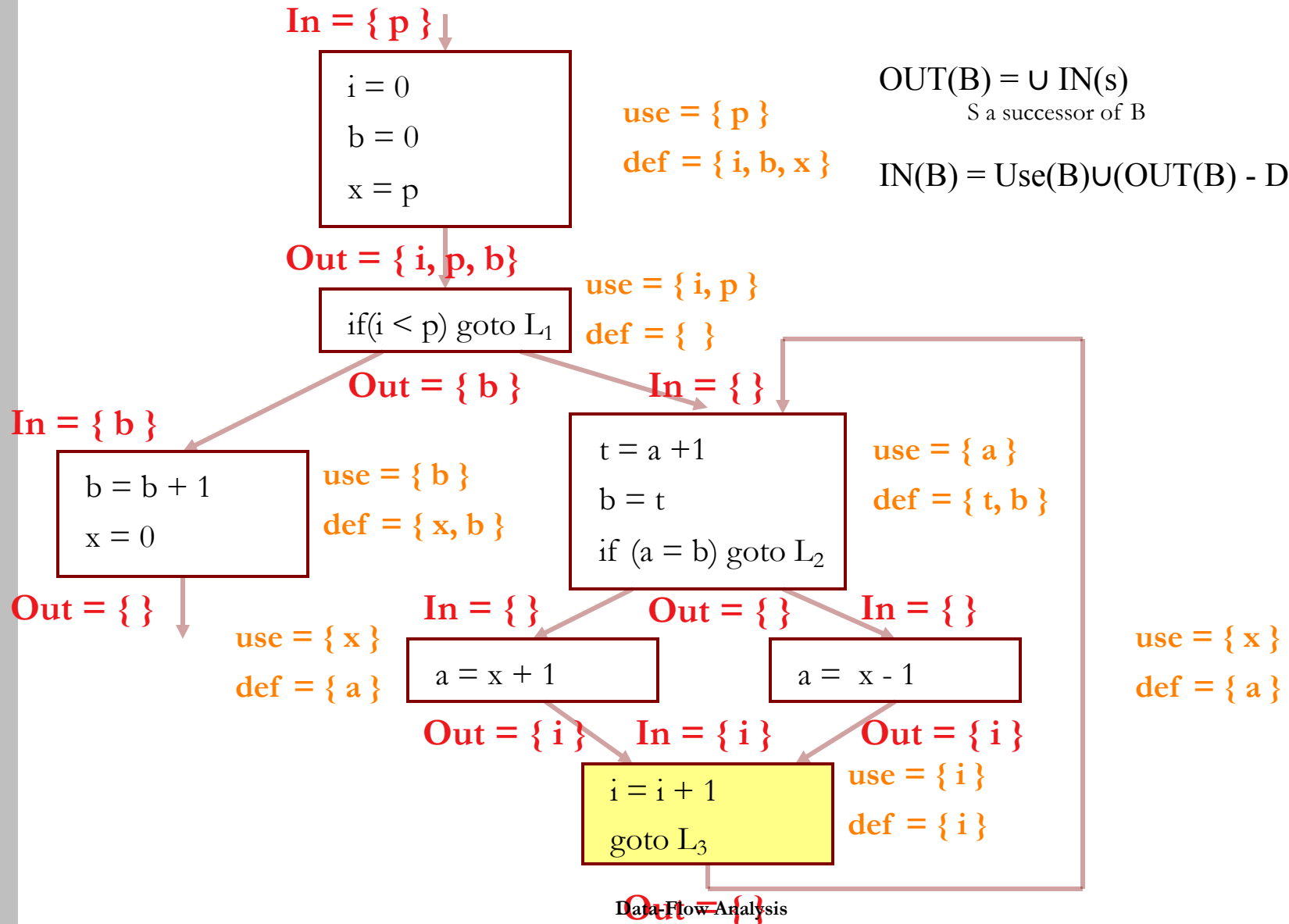
$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$



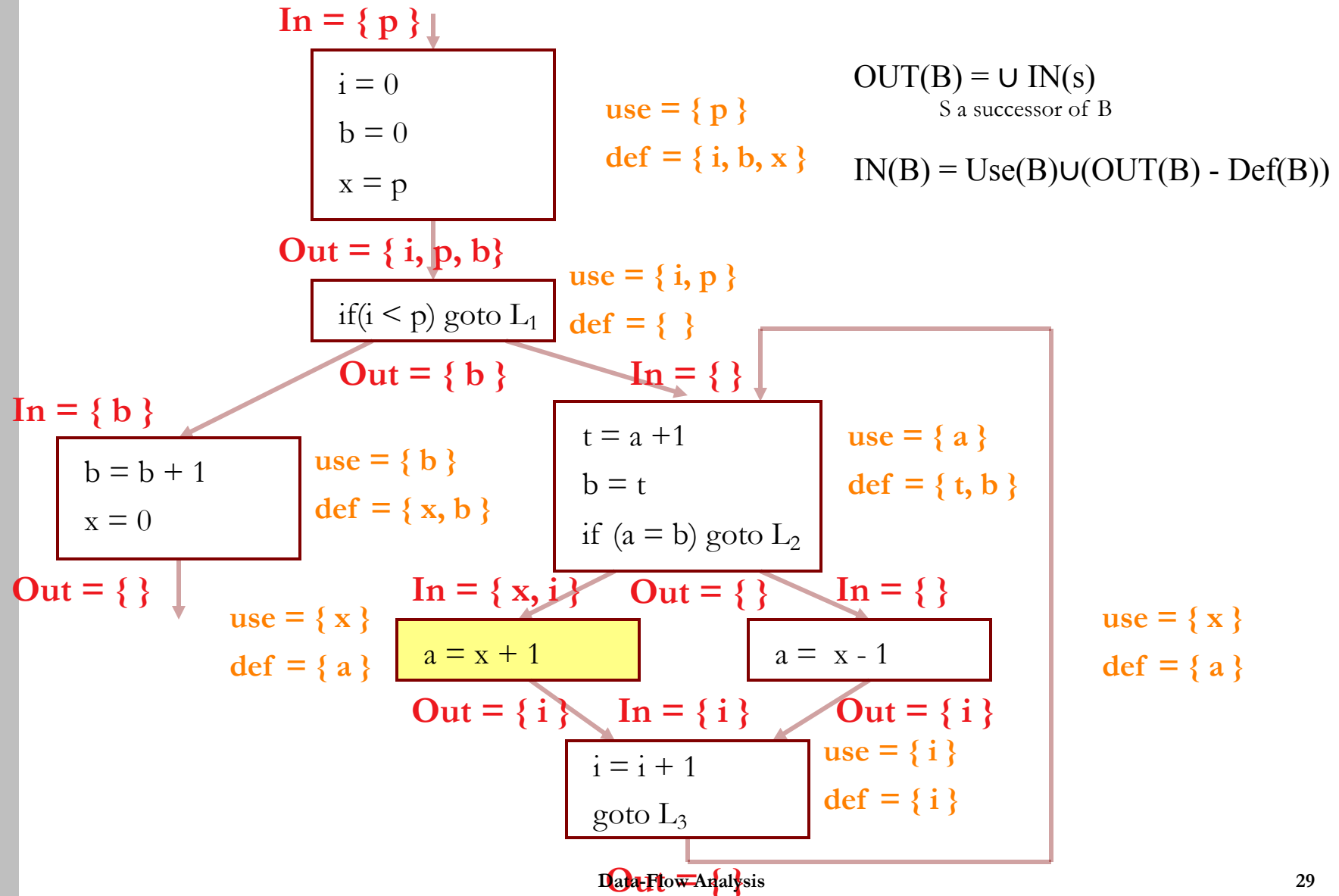
Example

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

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



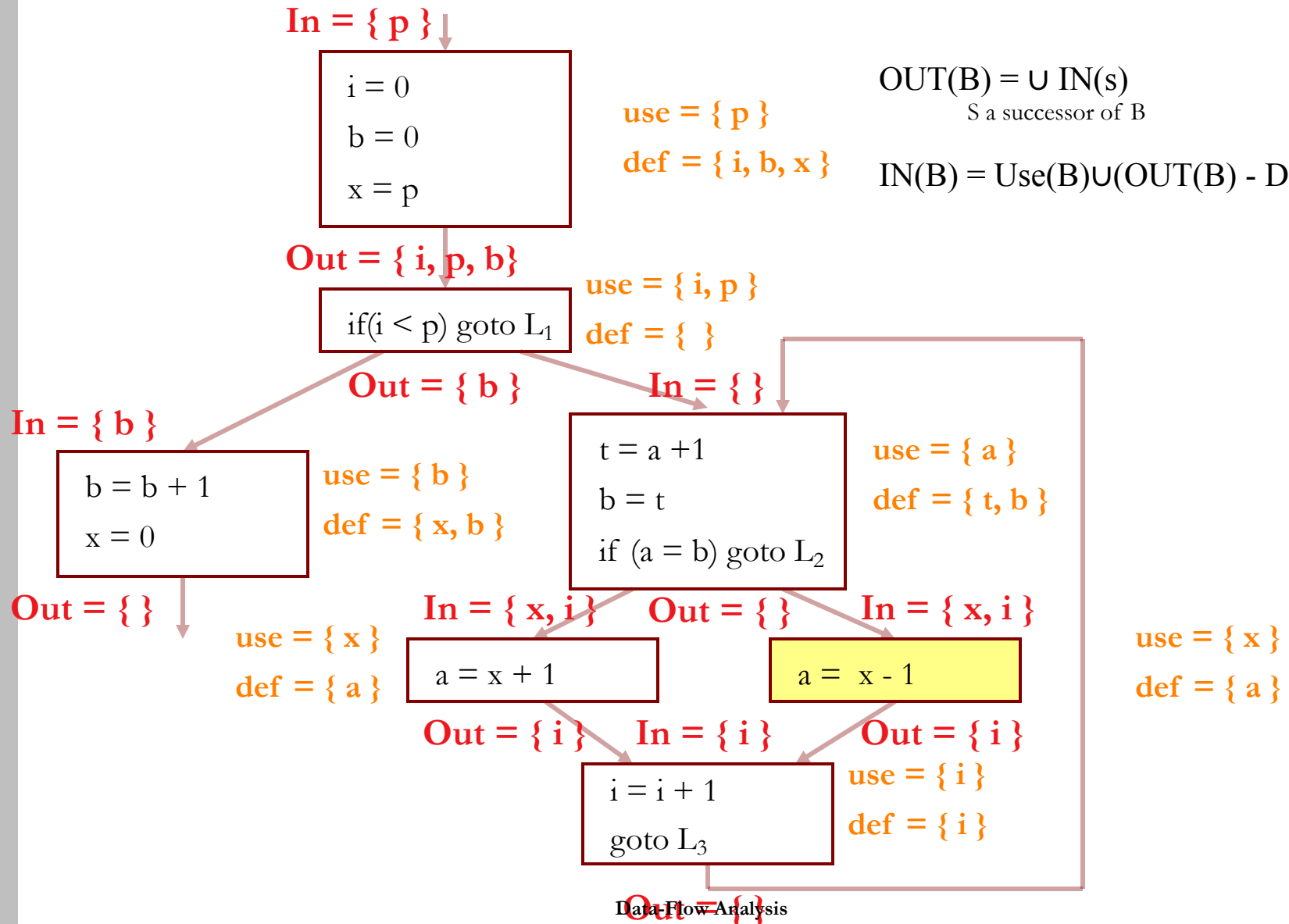
Example



Example

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

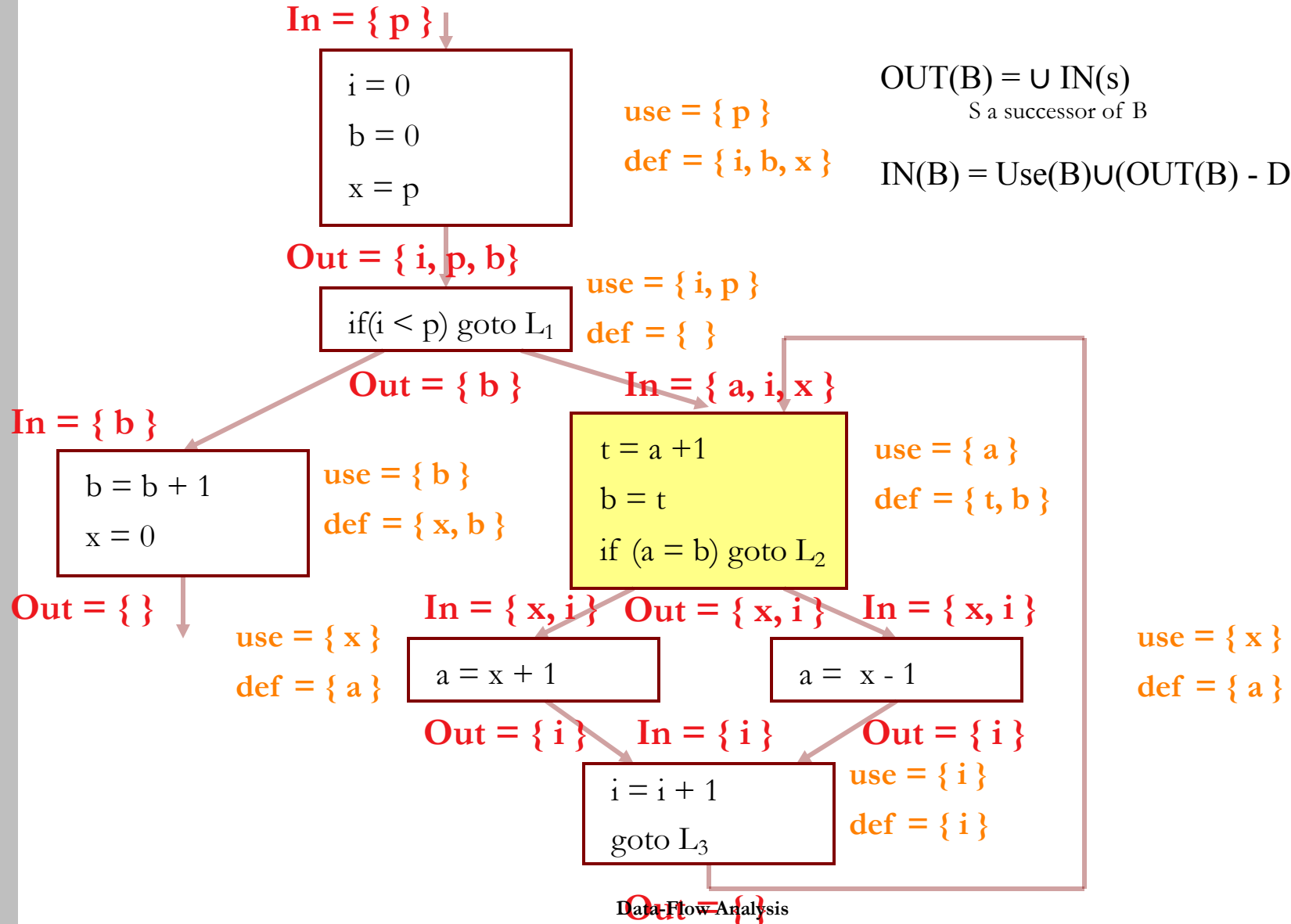
$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$



Example

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

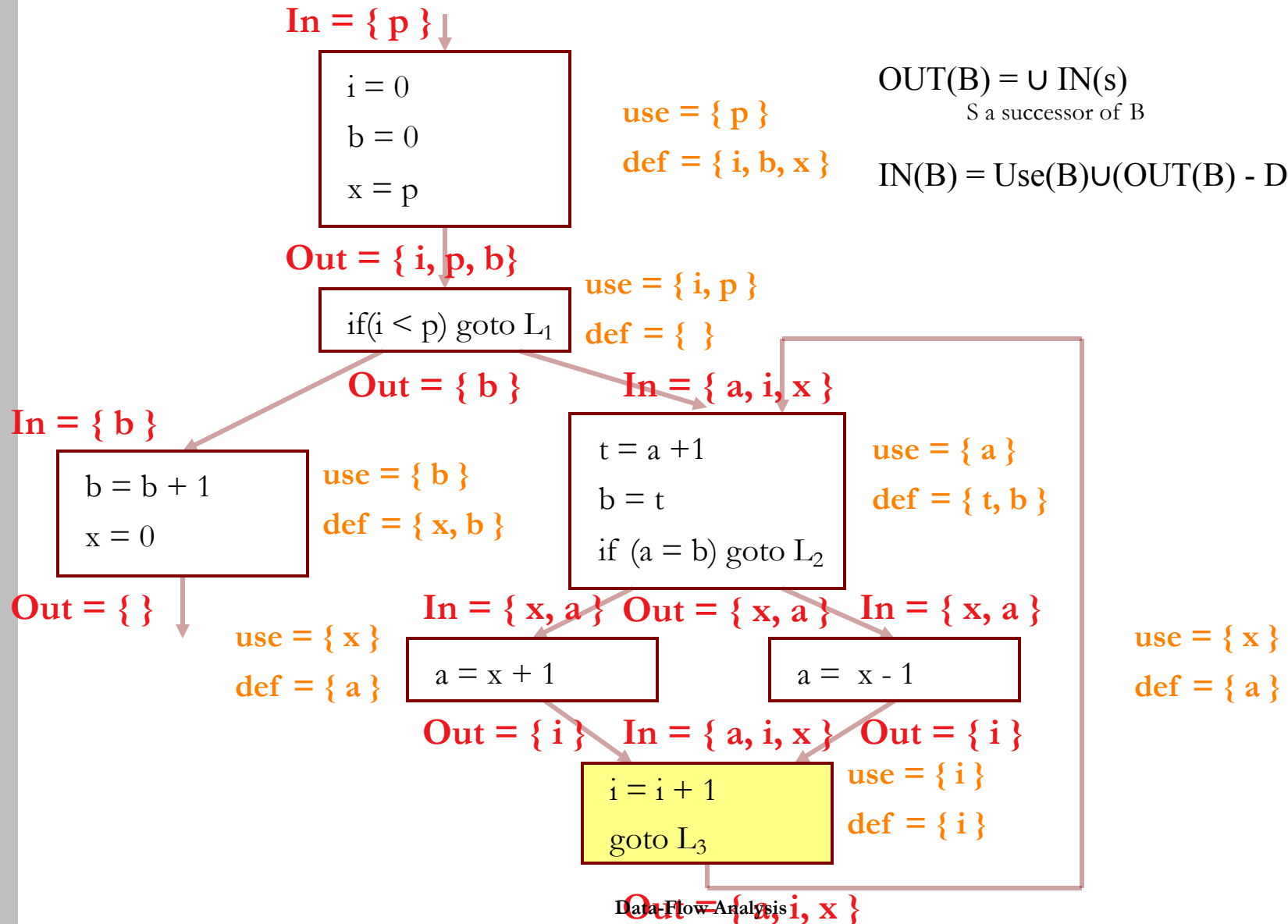
$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$



Example

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

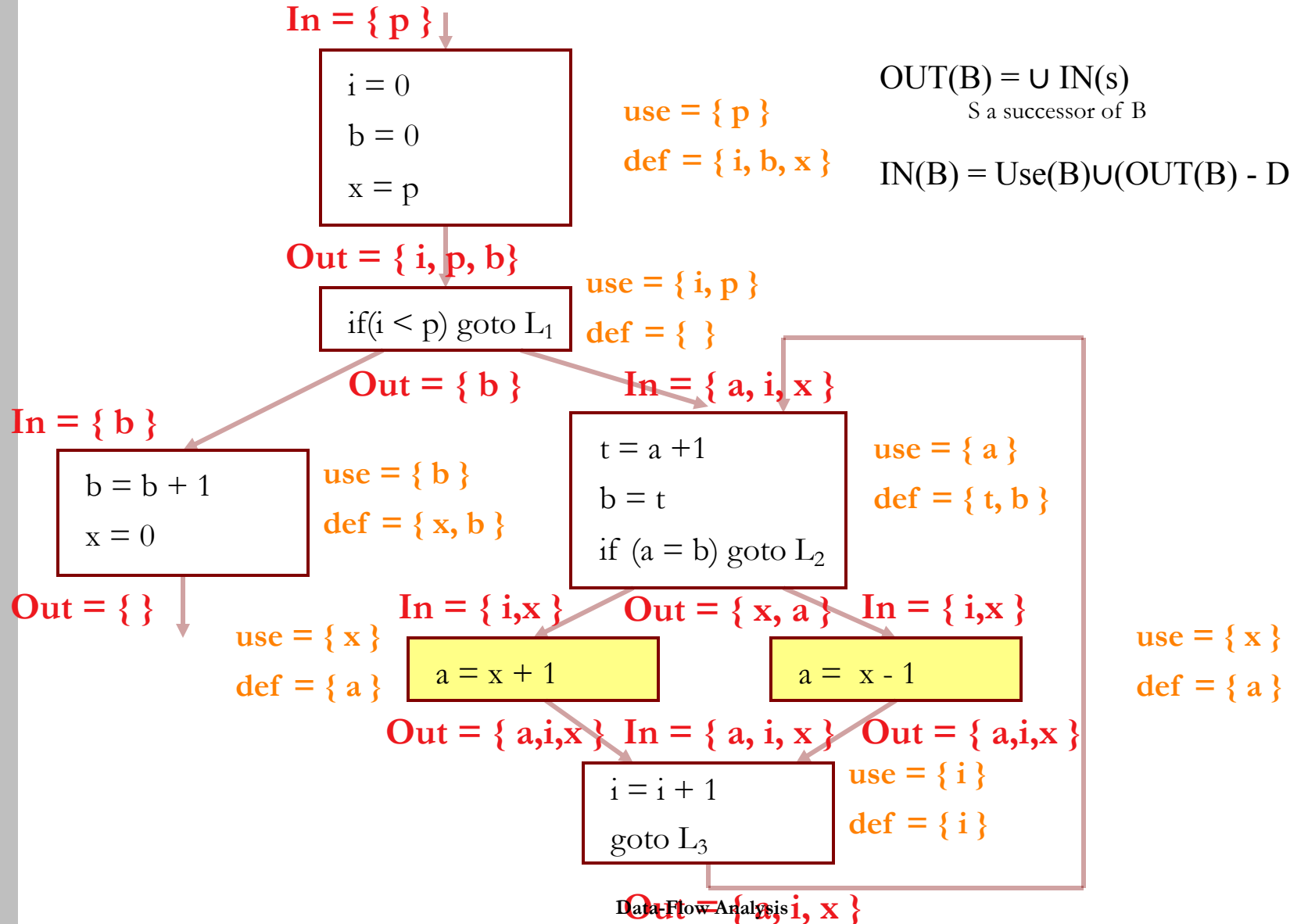
$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$



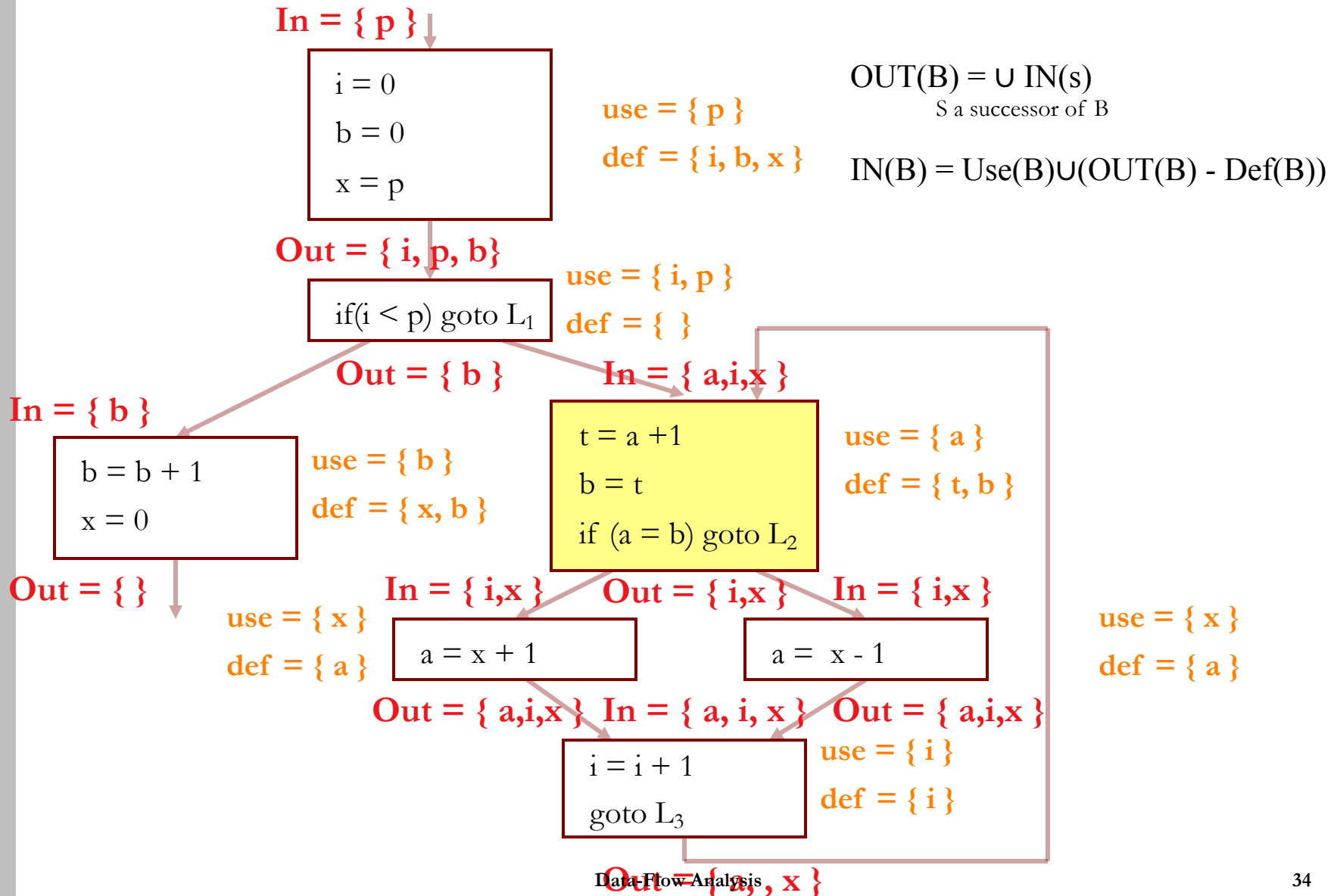
Example

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

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



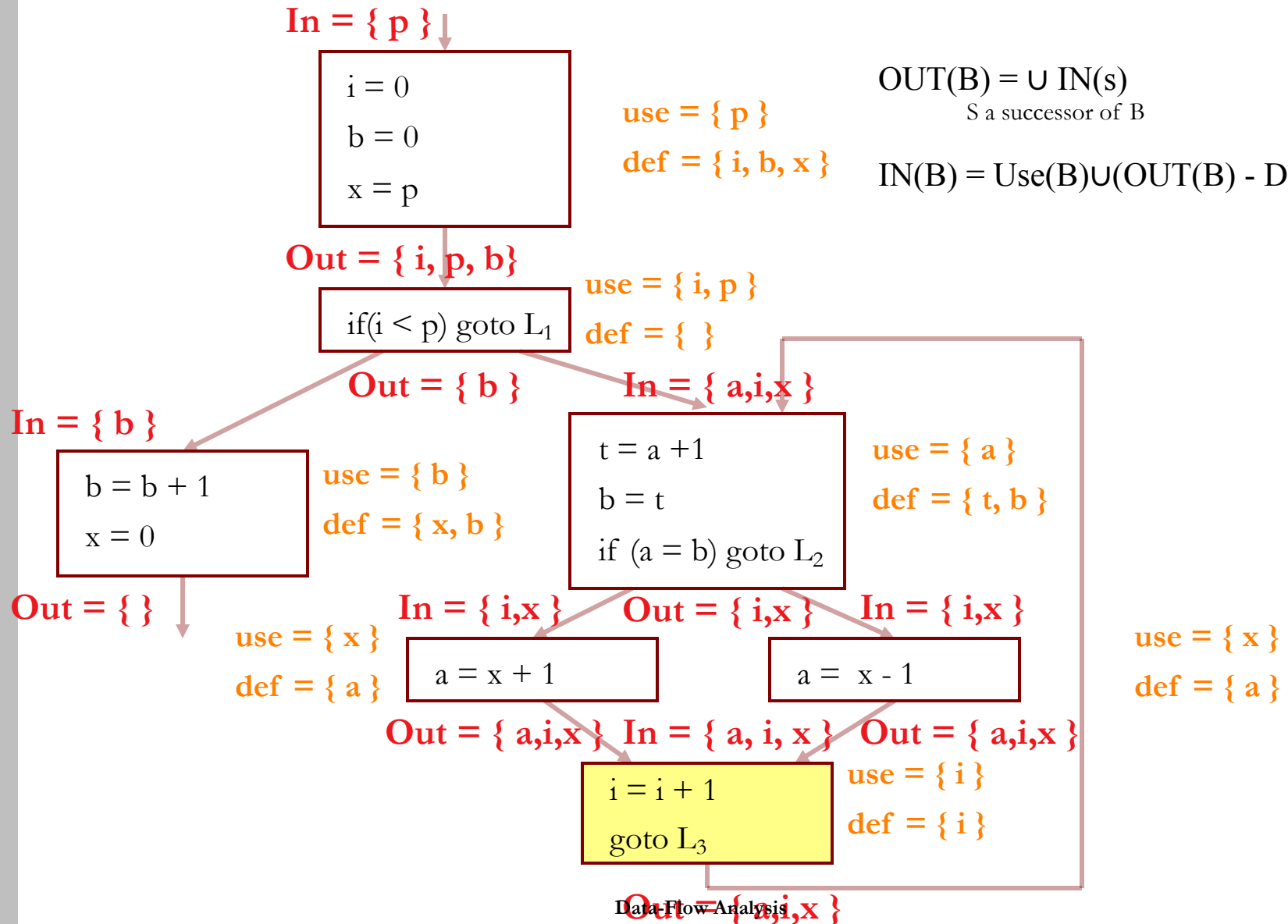
Example



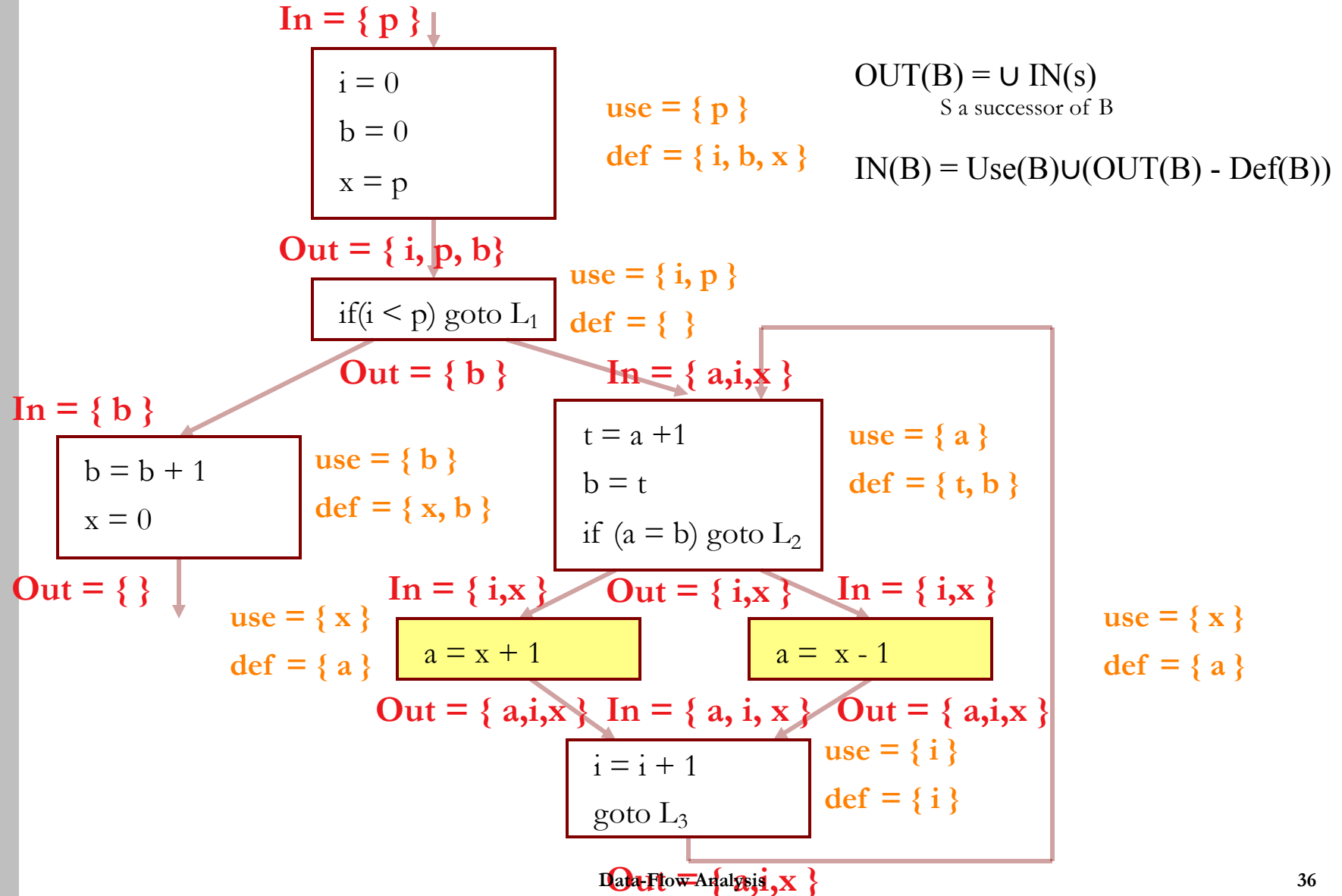
Example

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

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



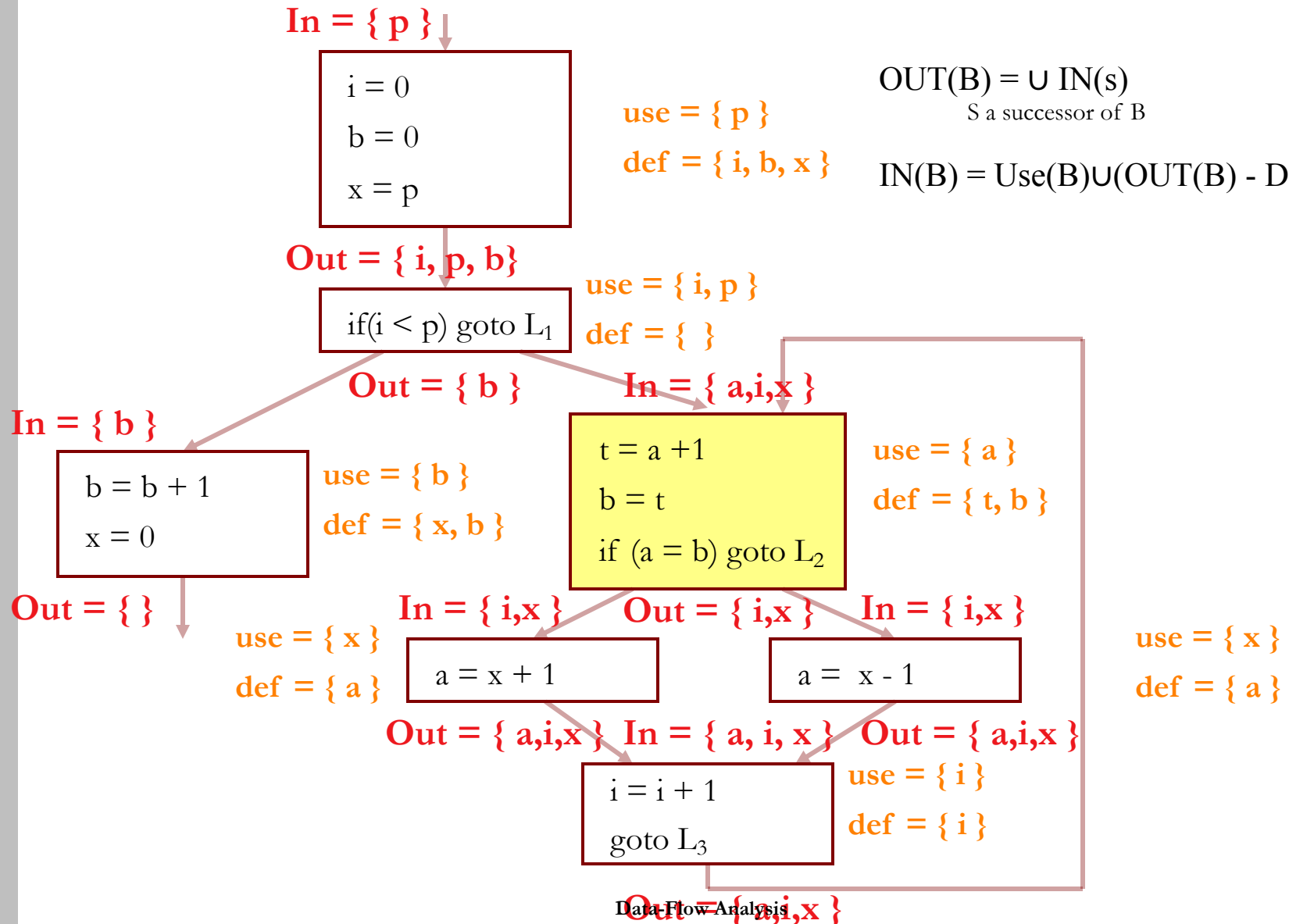
Example



Example

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

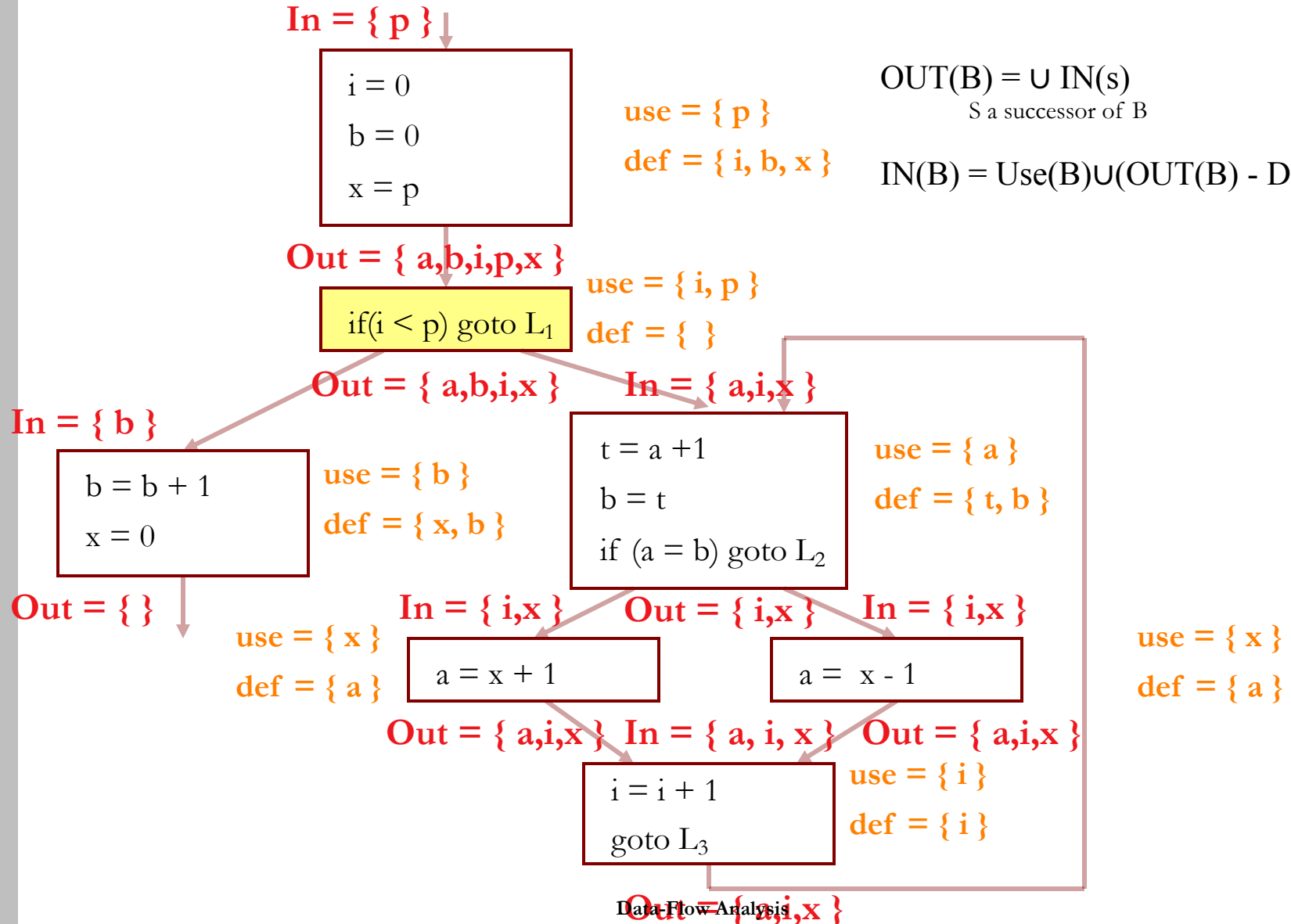
$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$



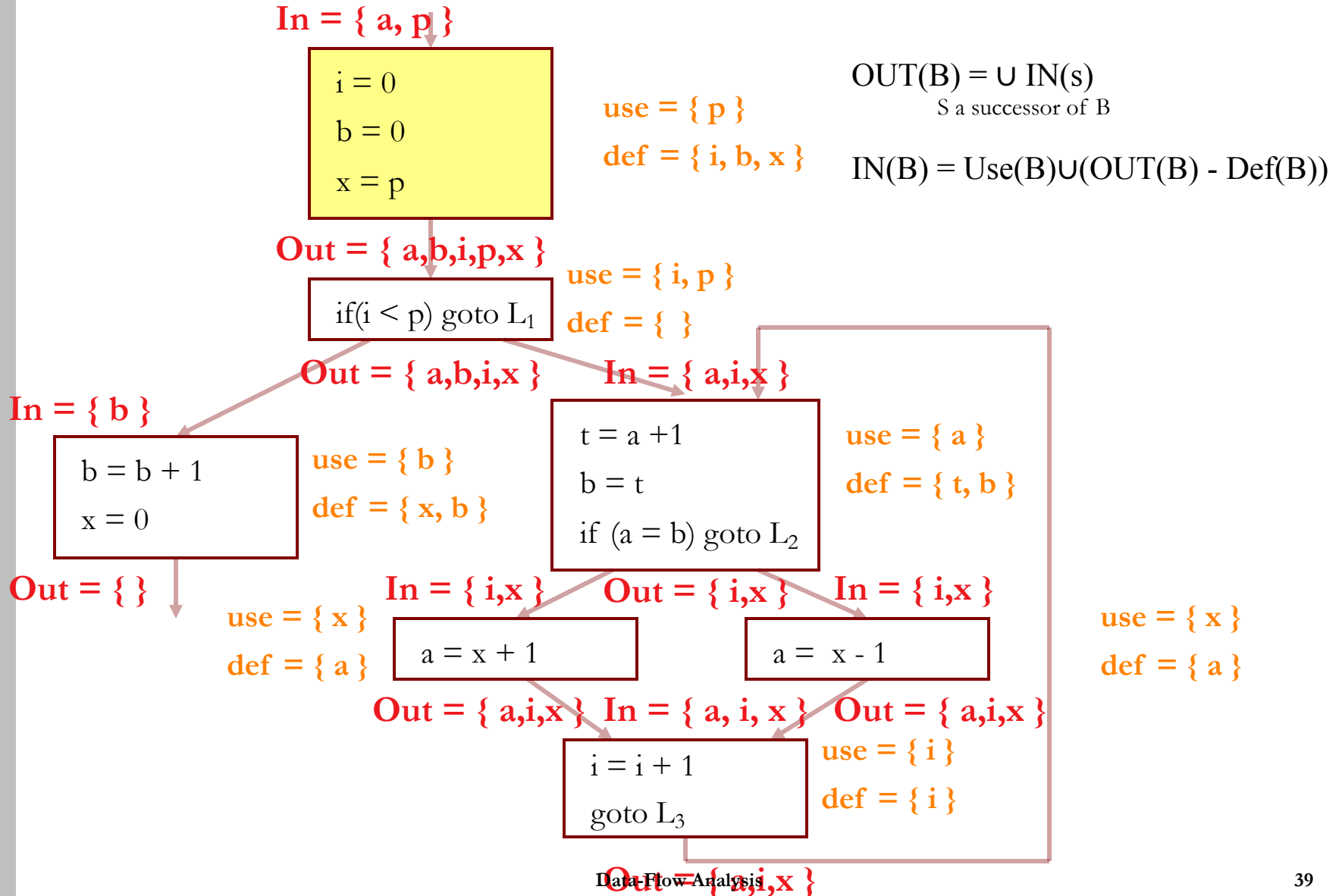
Example

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

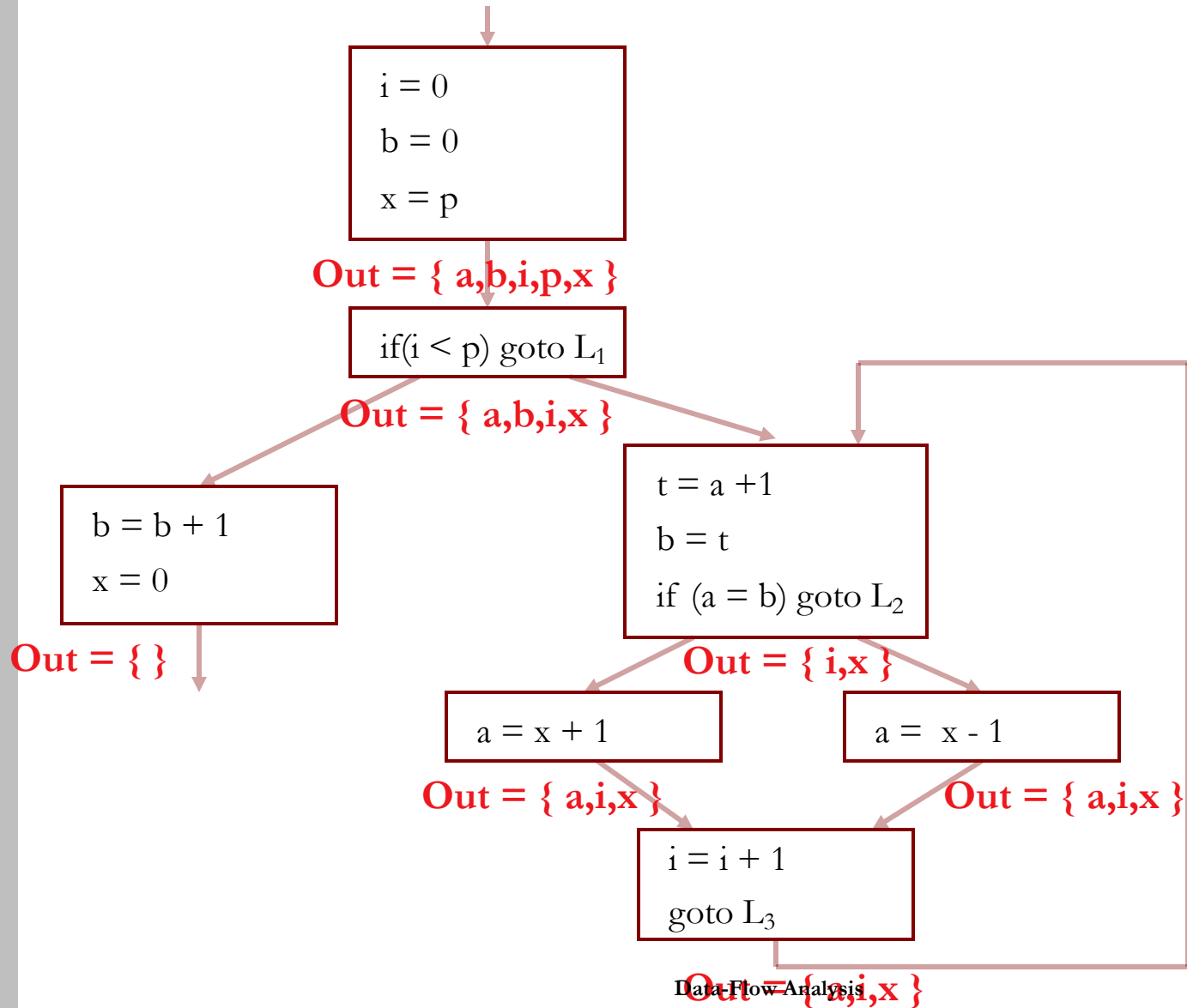
$$IN(B) = Use(B) \cup (OUT(B) - Def(B))$$



Example



Example



Summary

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