15-817A Model Checking and Abstract Interpretation

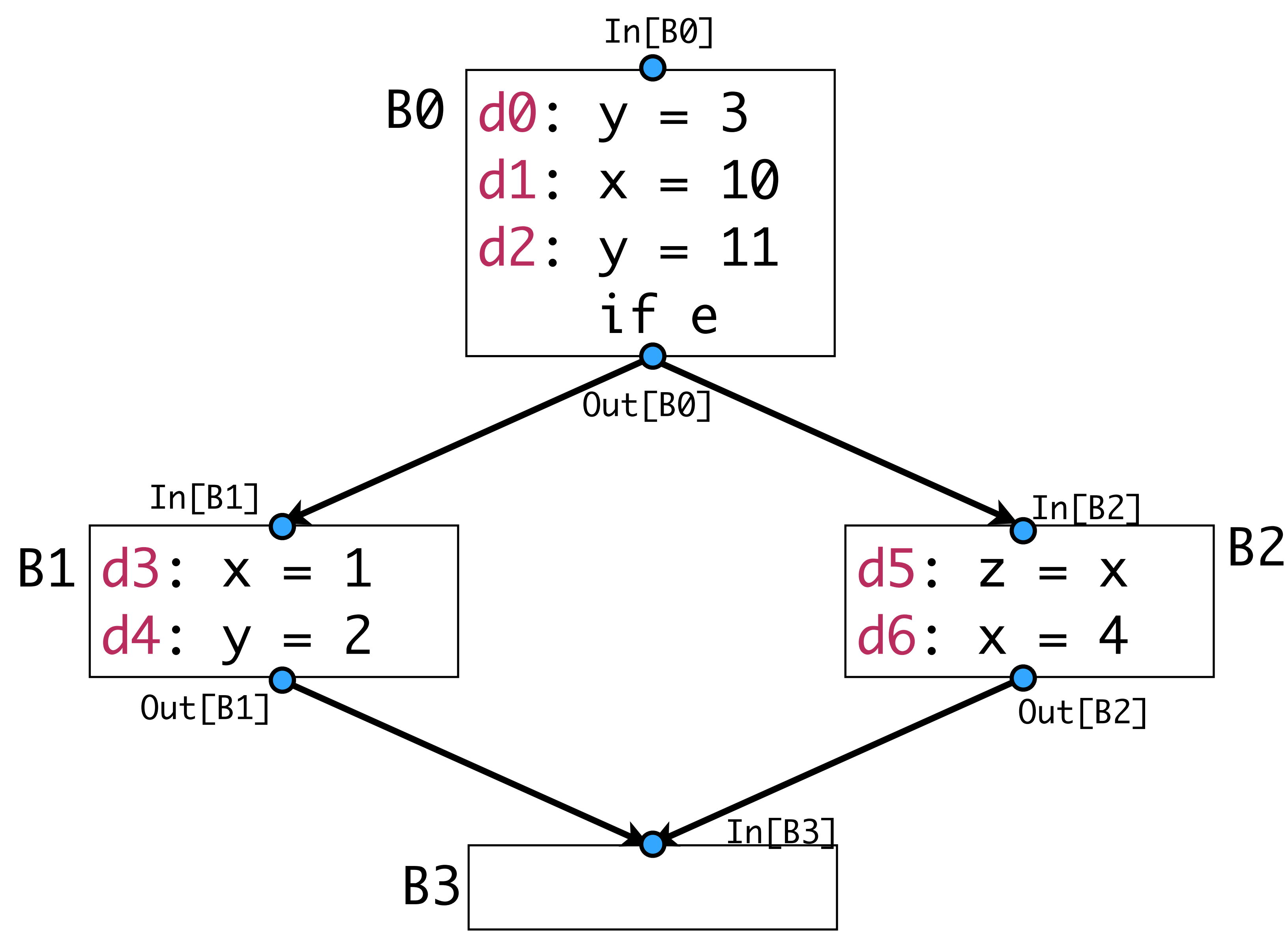
Data Flow Analysis

Soonho Kong
soonhok@cs.cmu.edu

2011/1/26

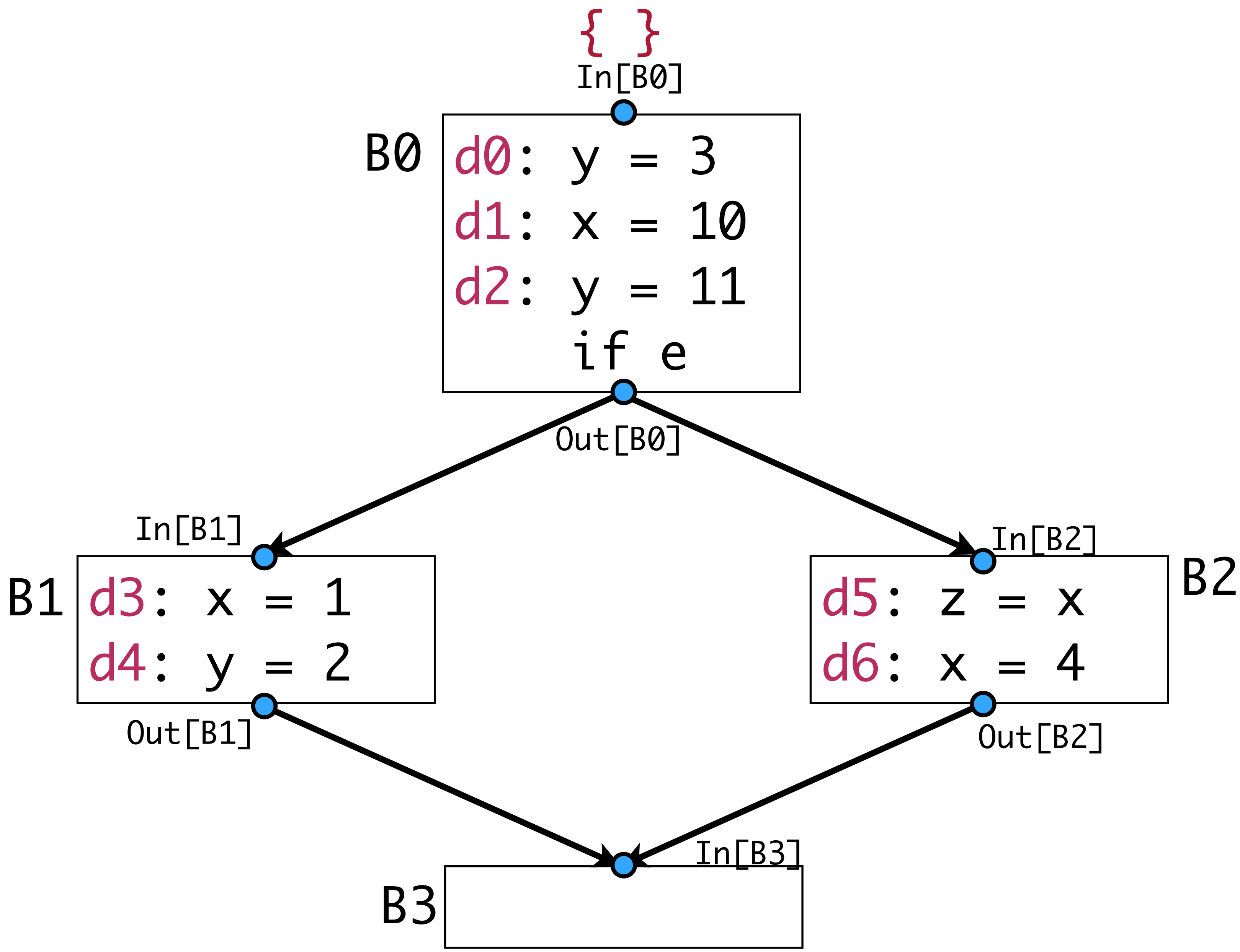
Computer Science Department
Carnegie Mellon University

Motivating Example: Reaching Definitions



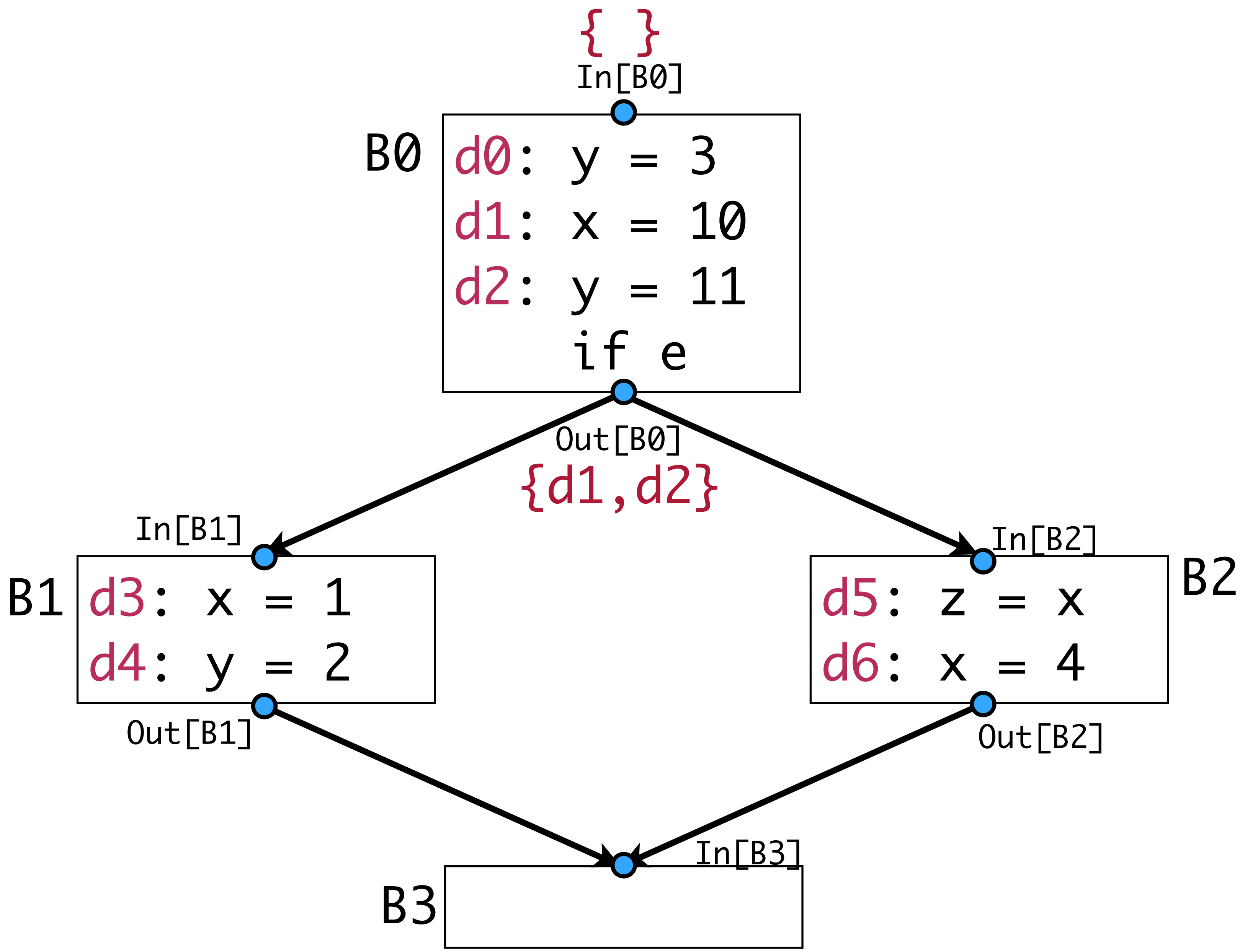
A definition d *reaches* a point p if there is a path from the point immediately following d to p

Motivating Example: Reaching Definitions



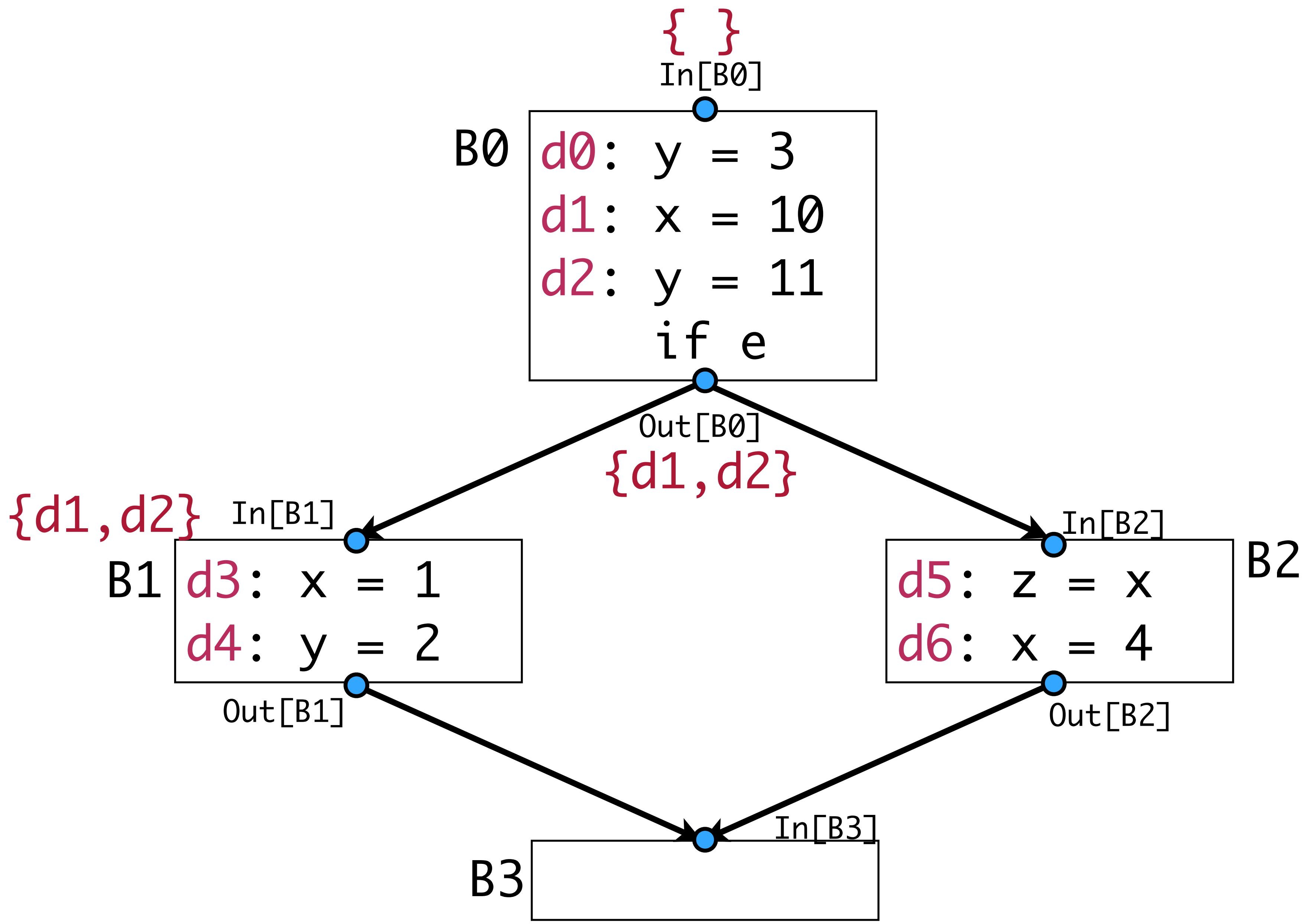
A definition d *reaches* a point p if there is a path from the point immediately following d to p

Motivating Example: Reaching Definitions



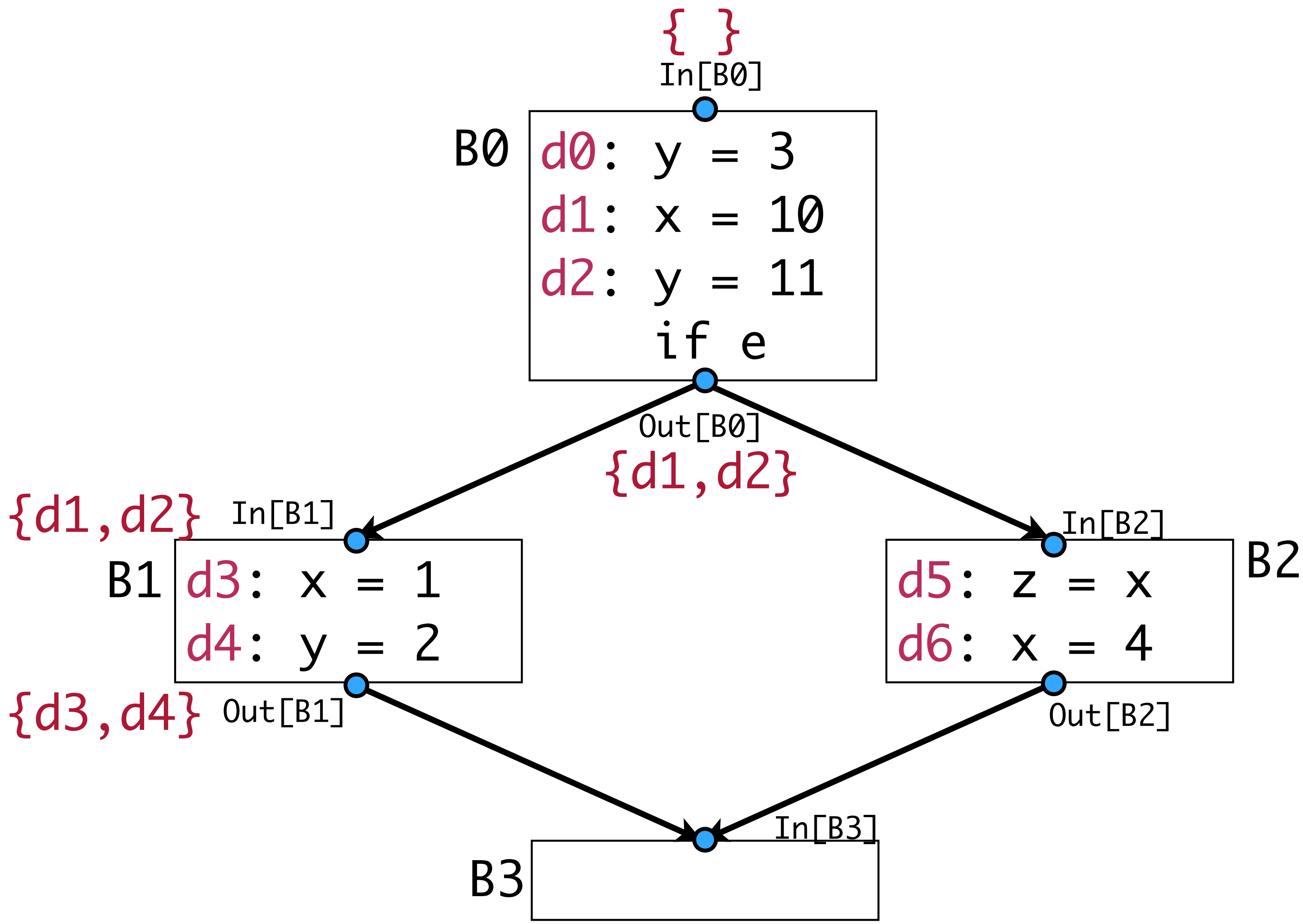
A definition d *reaches* a point p if there is a path from the point immediately following d to p

Motivating Example: Reaching Definitions



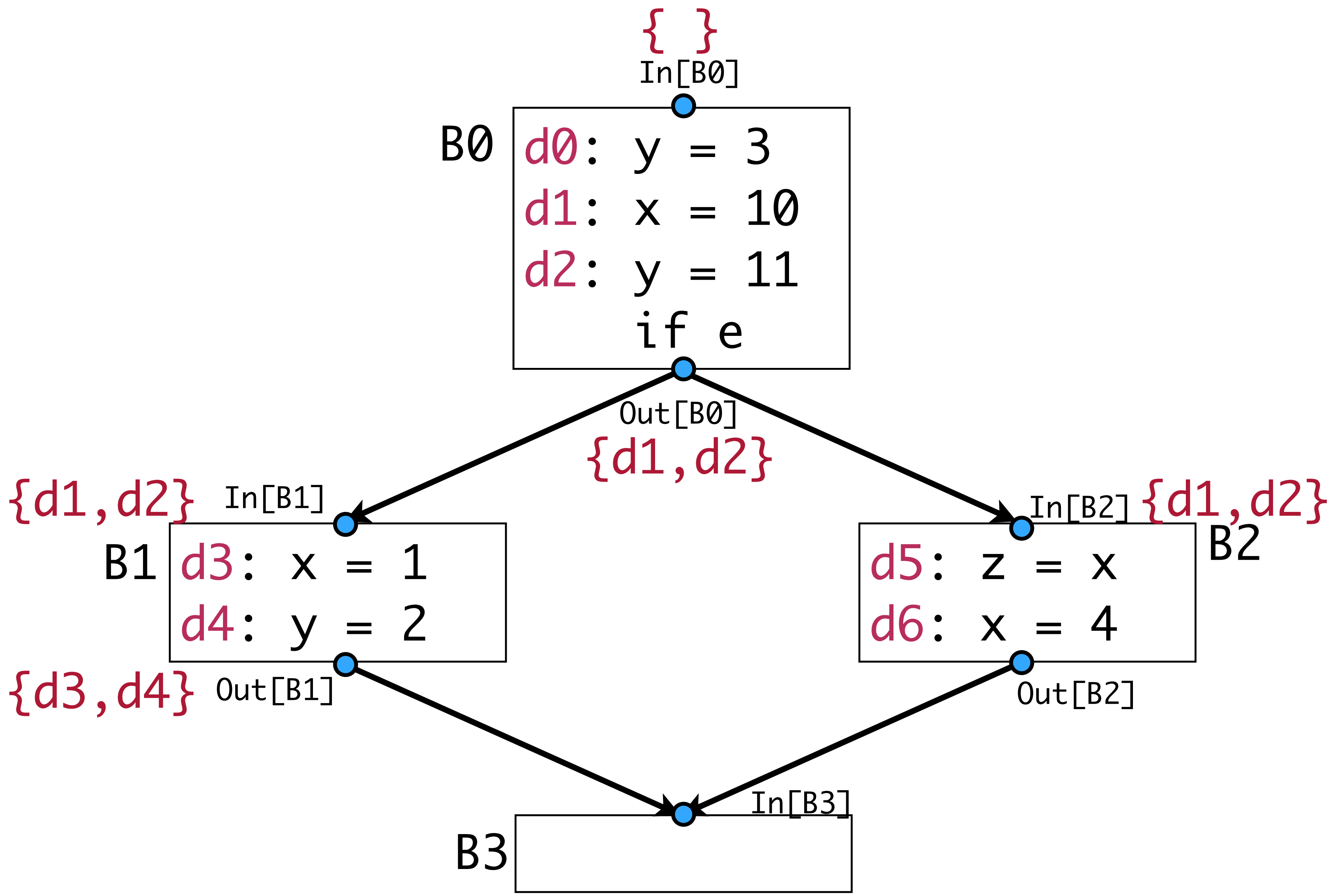
A definition d *reaches* a point p if there is a path from the point immediately following d to p

Motivating Example: Reaching Definitions



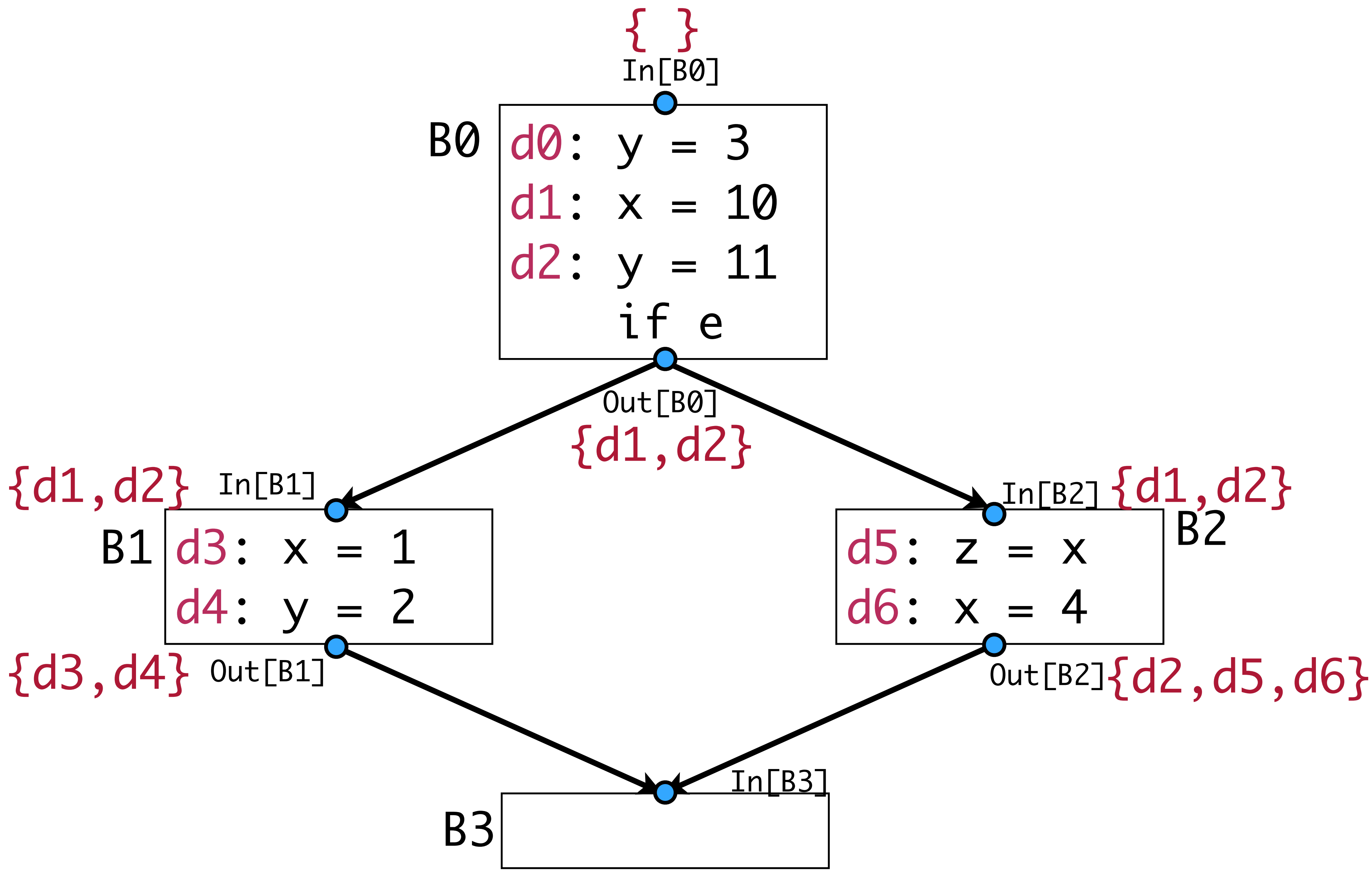
A definition d *reaches* a point p if there is a path from the point immediately following d to p

Motivating Example: Reaching Definitions



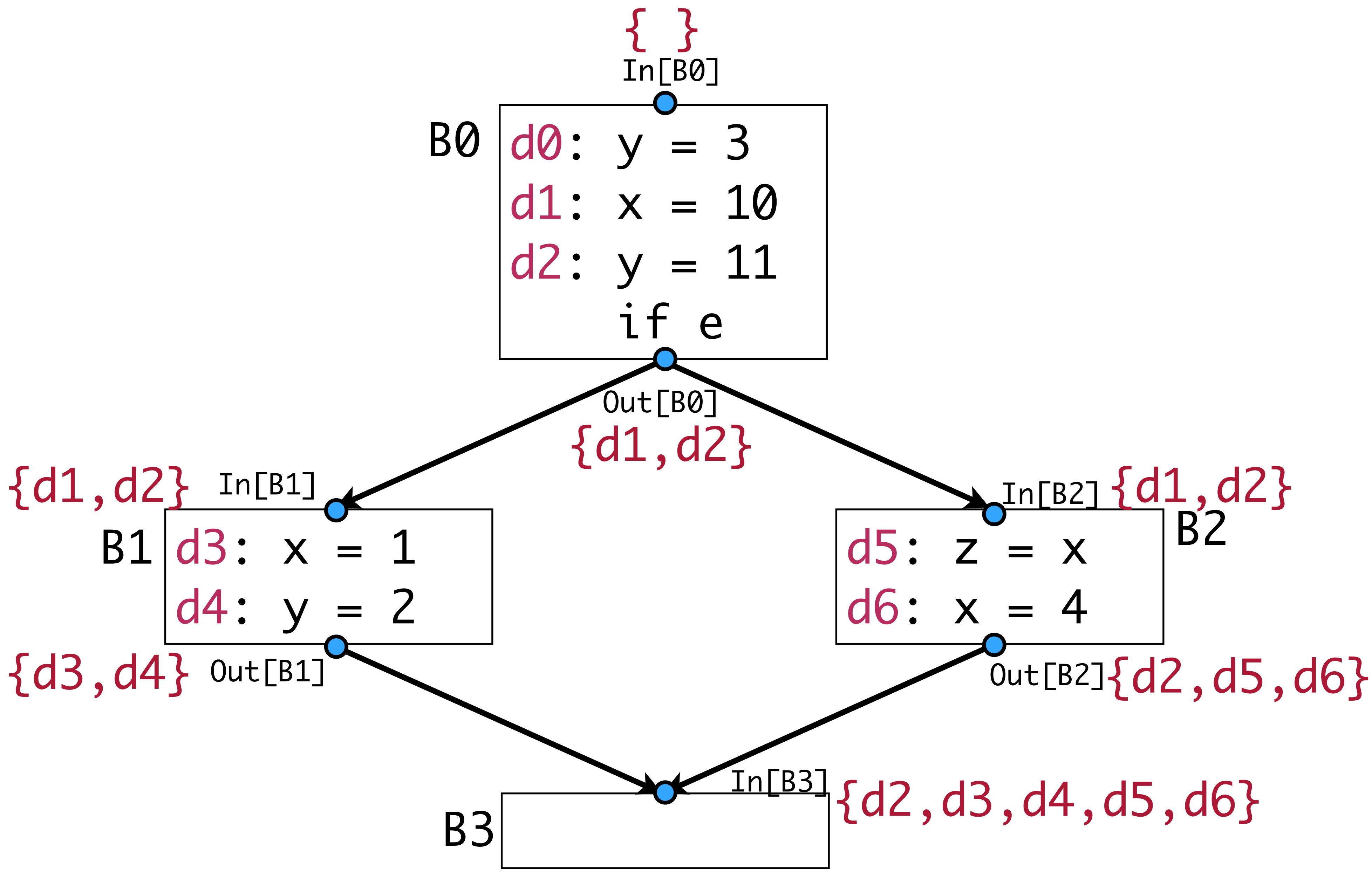
A definition d *reaches* a point p if there is a path from the point immediately following d to p

Motivating Example: Reaching Definitions



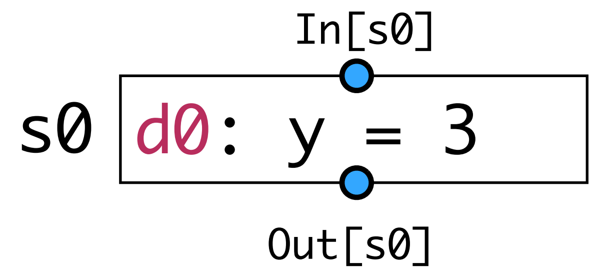
A definition d *reaches* a point p if there is a path from the point immediately following d to p

Motivating Example: Reaching Definitions



A definition d *reaches* a point p if there is a path from the point immediately following d to p

Transfer Function (statement-level)



What is the relation between `In[s0]` and `Out[s0]`?

$$Out[s_0] = f_{d_0}(In[s_0])$$

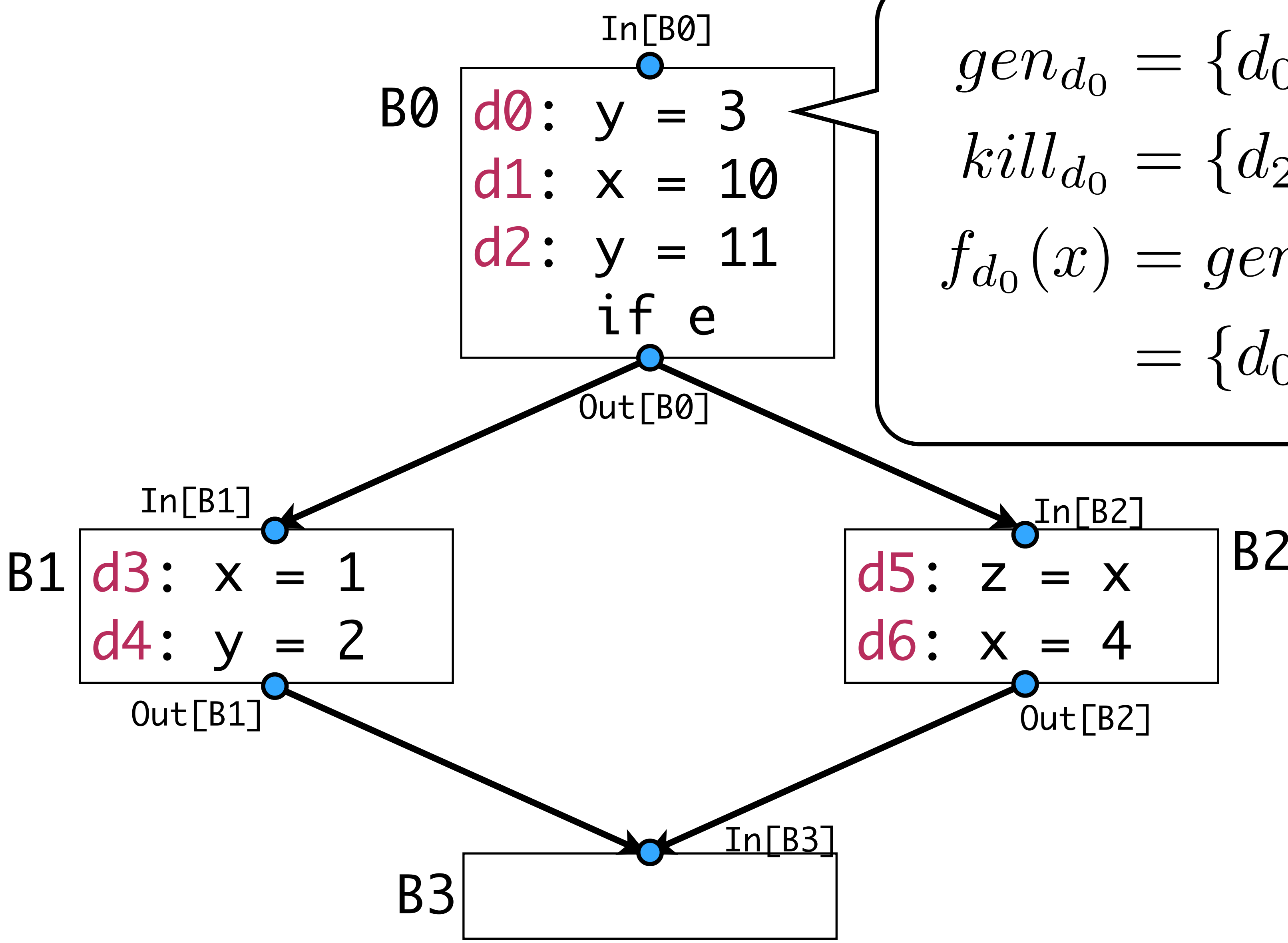
$$f_d(x) = gen_d \cup (x - kill_d)$$

where

gen_d is definition generated: $gen_d = \{d\}$

$kill_d$ is set of all other defs to x in the rest of program

Transfer Function (statement-level)

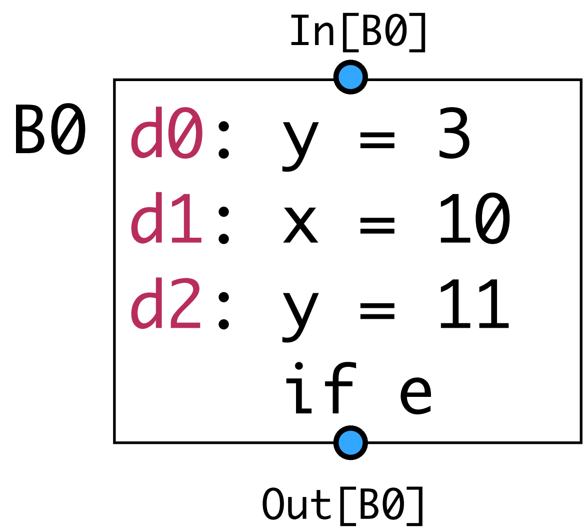


$$gen_{d_0} = \{d_0\}$$

$$kill_{d_0} = \{d_2, d_4\}$$

$$\begin{aligned} f_{d_0}(x) &= gen_{d_0} \cup (x - kill_{d_0}) \\ &= \{d_0\} \cup (x - \{d_2, d_4\}) \end{aligned}$$

Transfer Function (block-level)

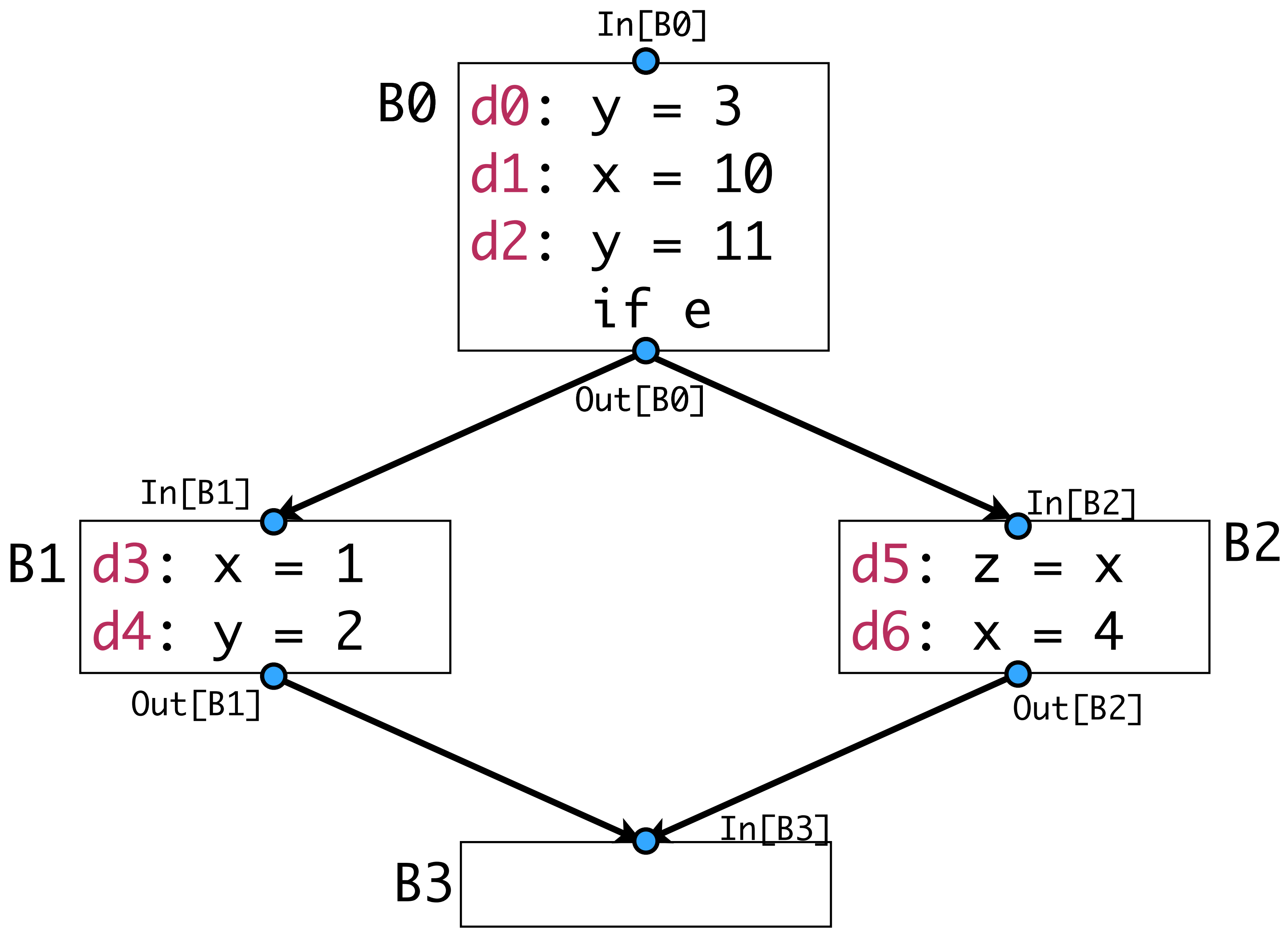


What is the relation between In[B0] and Out[B0]?

$$Out[s_0] = f_B(In[s_0])$$

$$f_B(x) = f_{d_n} \circ \cdots \circ f_{d_0}(x)$$

Transfer Function (block-level)

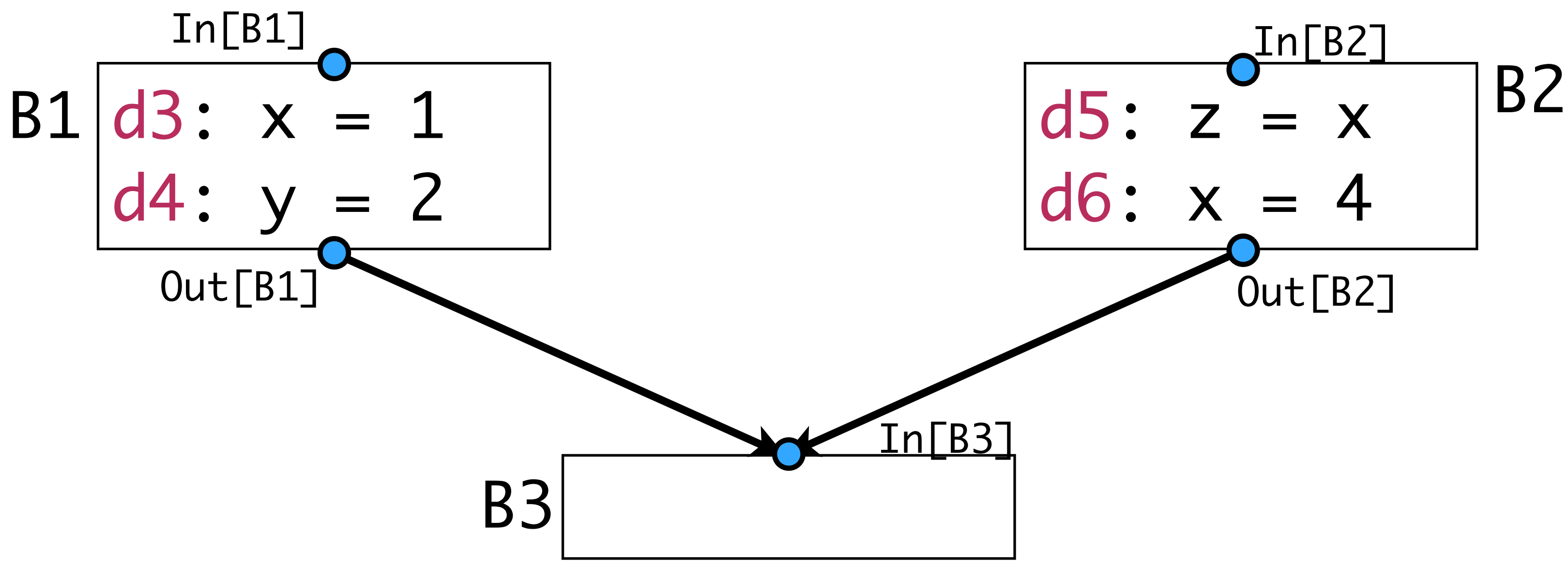


$$Out[B_0] = \{d_1, d_2\} \cup (In[B_0] - \{d_3, d_4, d_6, d_0\})$$

$$Out[B_1] = \{d_3, d_4\} \cup (In[B_1] - \{d_0, d_1, d_2, d_6\})$$

$$Out[B_2] = \{d_5, d_6\} \cup (In[B_2] - \{d_1, d_3\})$$

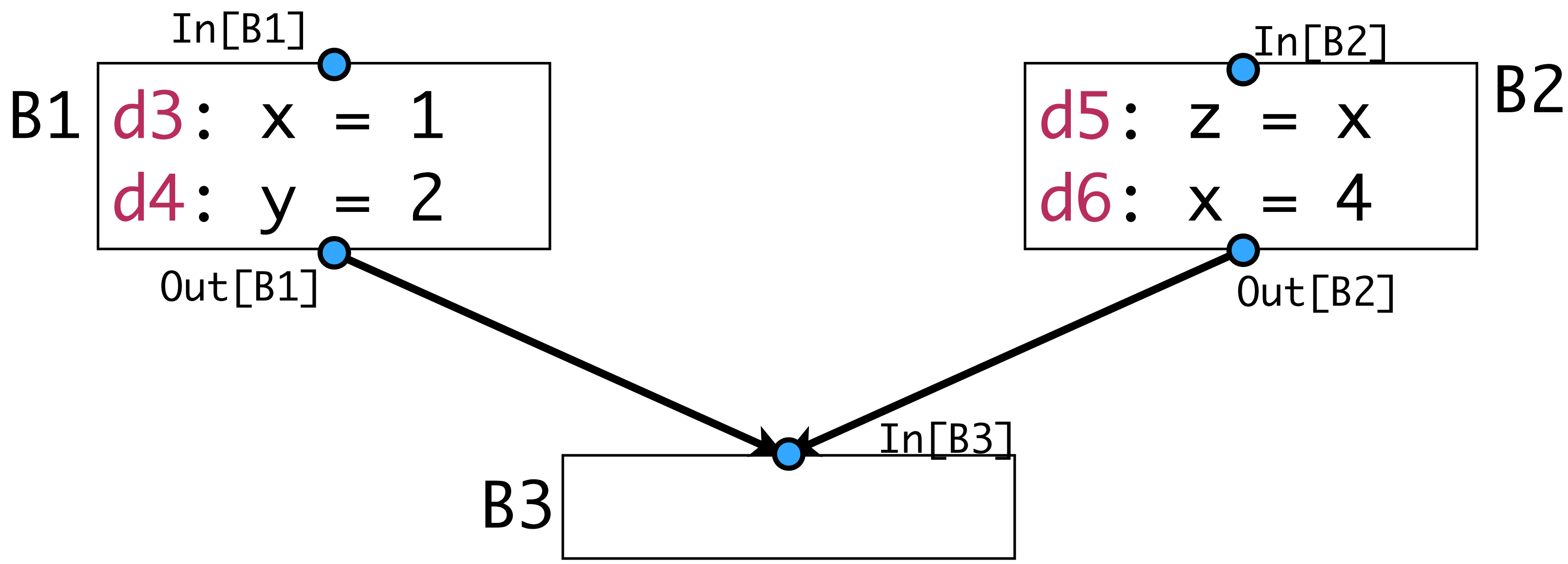
Meet Operator



What is the relation between `Out[B1]`, `Out[B2]`, and `Out[B0]`?

$$In[B] = \bigcup_{B':pred(B)} Out[B']$$

Meet Operator



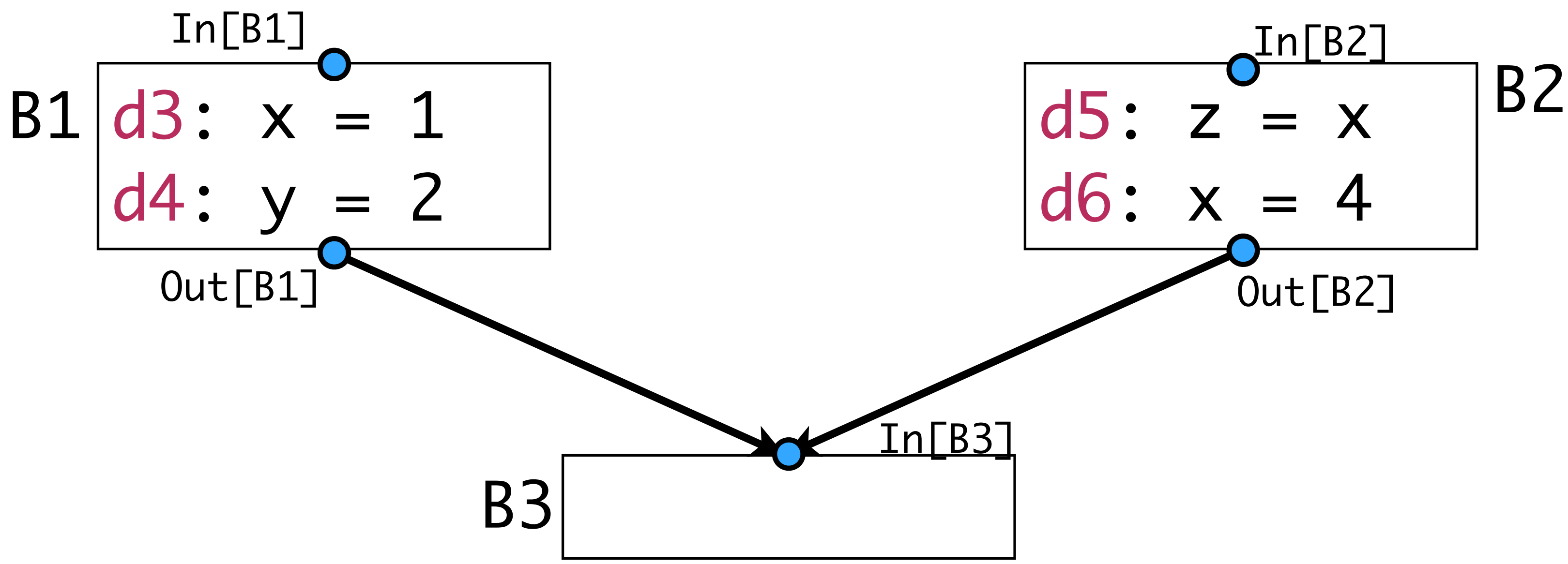
What is the relation between `Out[B1]`, `Out[B2]`, and `Out[B0]`?

$$In[B] = \bigcup_{B':pred(B)} Out[B']$$

Why $\bigcup$?

"A definition d *reaches* a point p if there is a path from the point immediately following d to p"

Meet Operator



What is the relation between `Out[B1]`, `Out[B2]`, and `Out[B0]`?

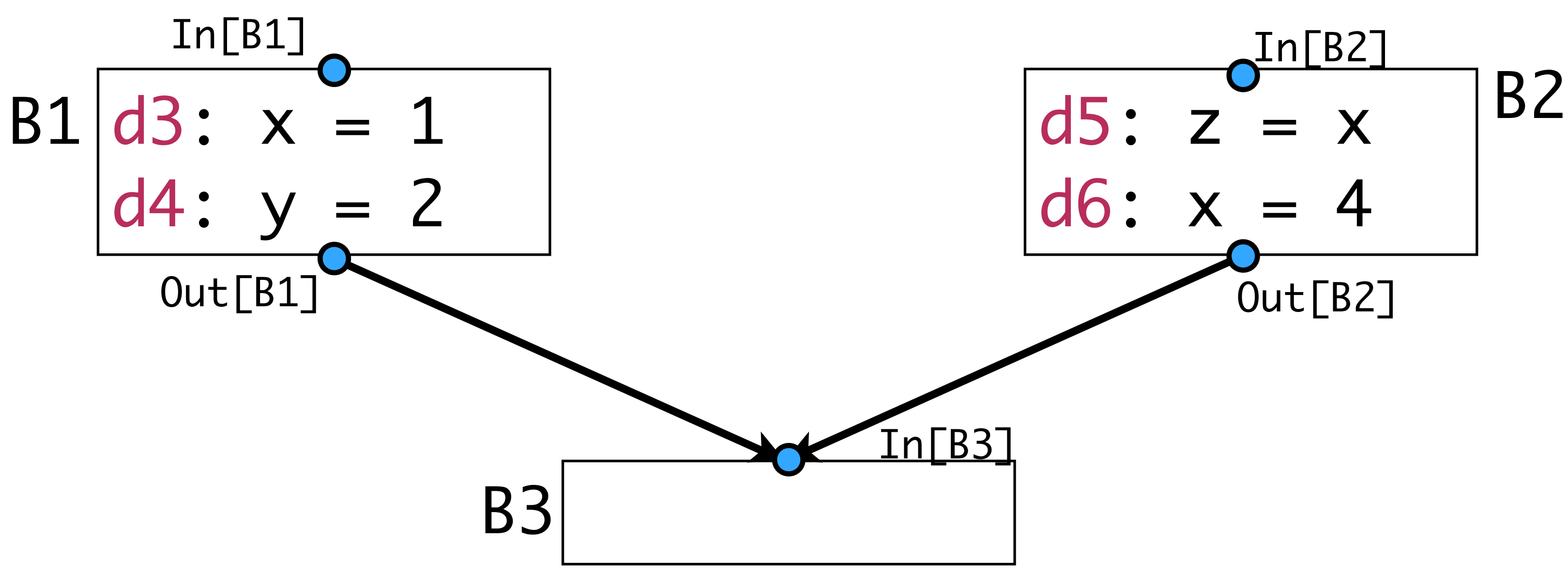
$$In[B] = \bigcup_{B':pred(B)} Out[B']$$

What if "MUST reach definition"?

"A definition d *reaches* a point p if for all path from the point immediately following d to p"

$$In[B] = \bigcap_{B':pred(B)} Out[B']$$

Meet Operator



What is the relation between `Out[B1]`, `Out[B2]`, and `Out[B0]`?

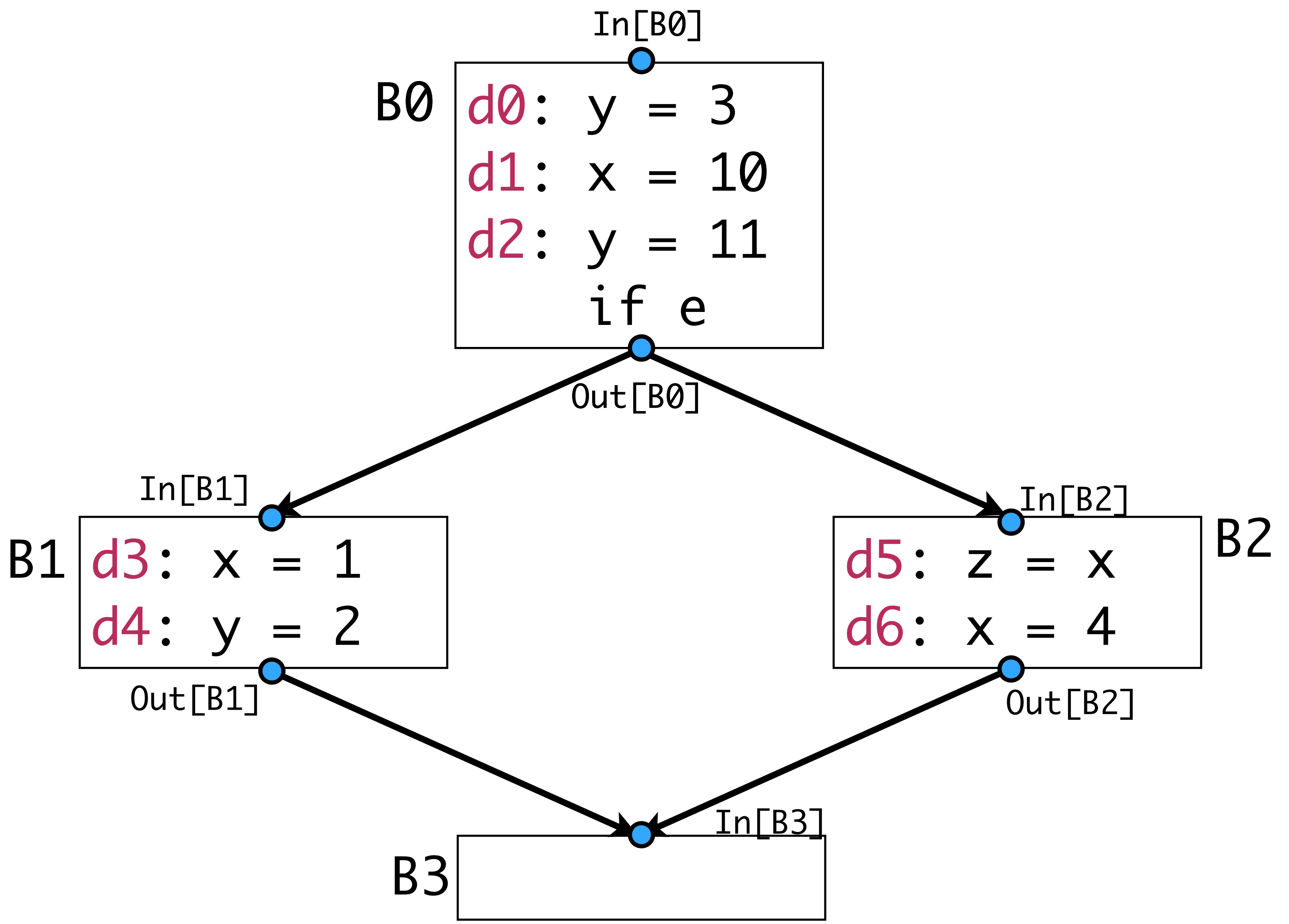
$$In[B] = \bigcup_{B':pred(B)} Out[B']$$

In general

$$In[B] = \bigsqcap_{B':pred(B)} Out[B']$$

and $\bigsqcap$ depends on the problem.

Meet Operator

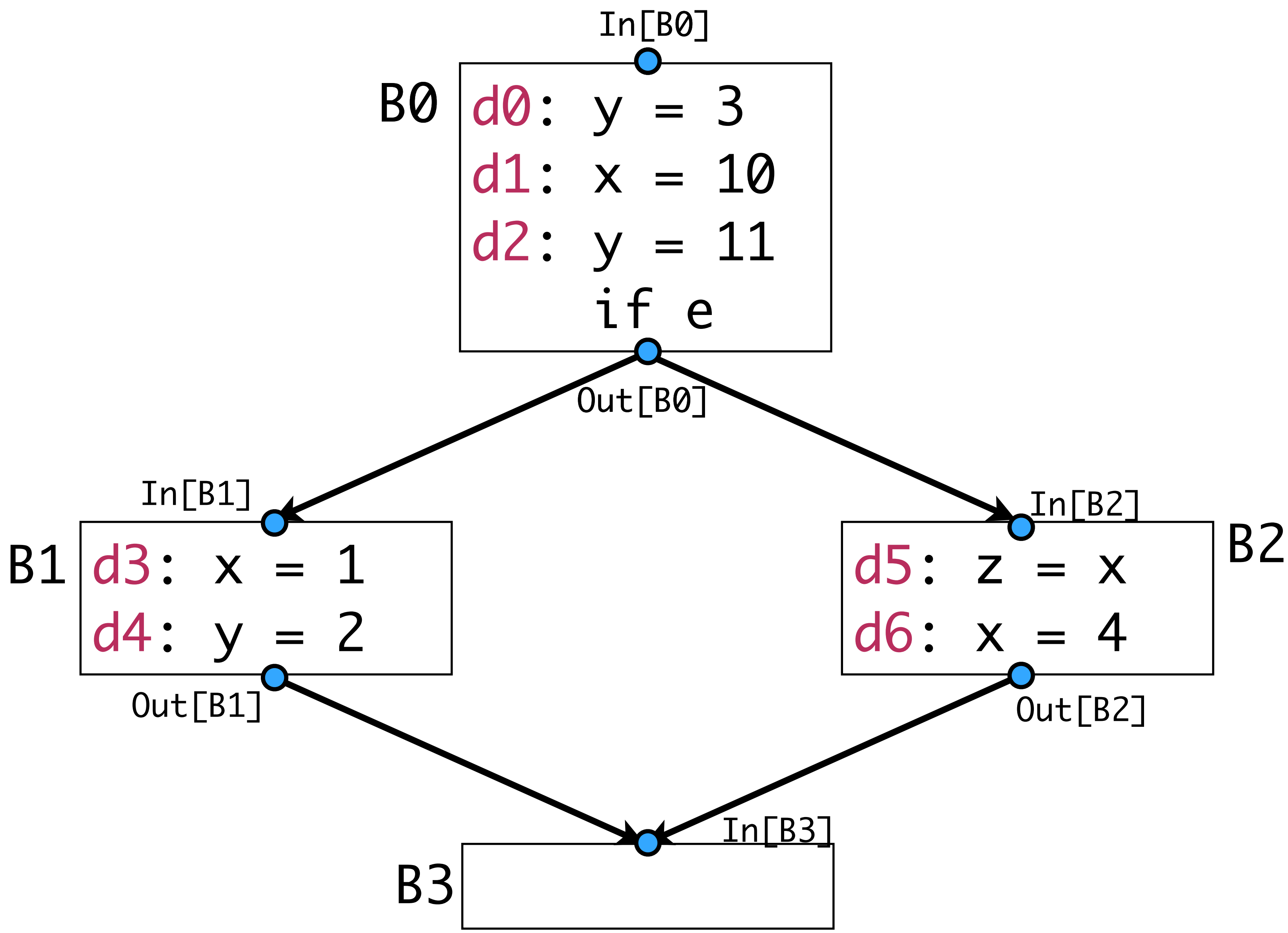


$$In[B_1] = Out[B_0]$$

$$In[B_2] = Out[B_0]$$

$$In[B_3] = Out[B_1] \cup Out[B_2]$$

Boundary Condition



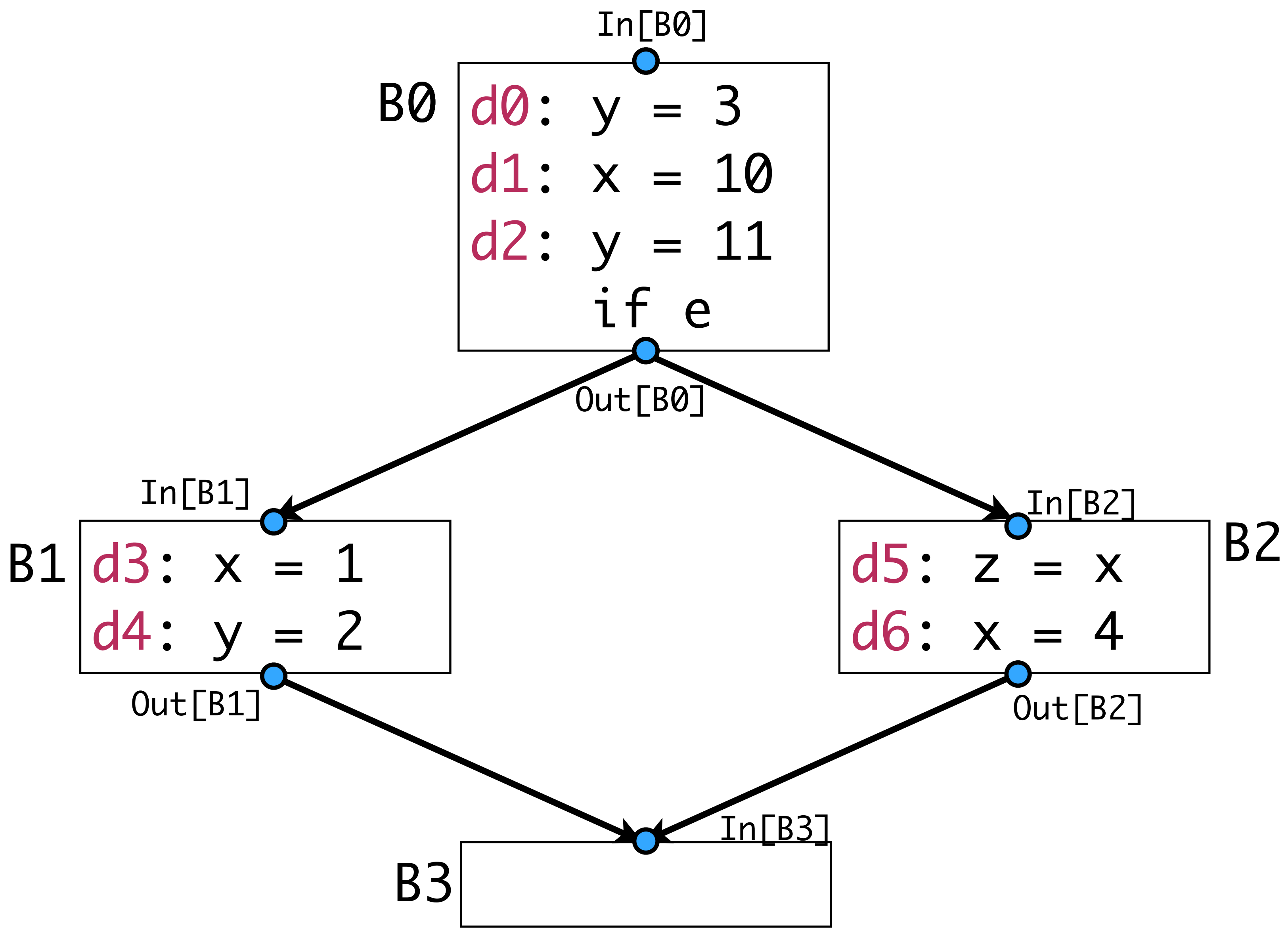
So far, we have...

$$Out[B_0] = \{d_1, d_2\} \cup (In[B_0] - \{d_3, d_4, d_6, d_0\}) \qquad In[B_1] = Out[B_0]$$

$$Out[B_1] = \{d_3, d_4\} \cup (In[B_1] - \{d_0, d_1, d_2, d_6\}) \qquad In[B_2] = Out[B_0]$$

$$Out[B_2] = \{d_5, d_6\} \cup (In[B_2] - \{d_1, d_3\}) \qquad In[B_3] = Out[B_1] \cup Out[B_2]$$

Boundary Condition

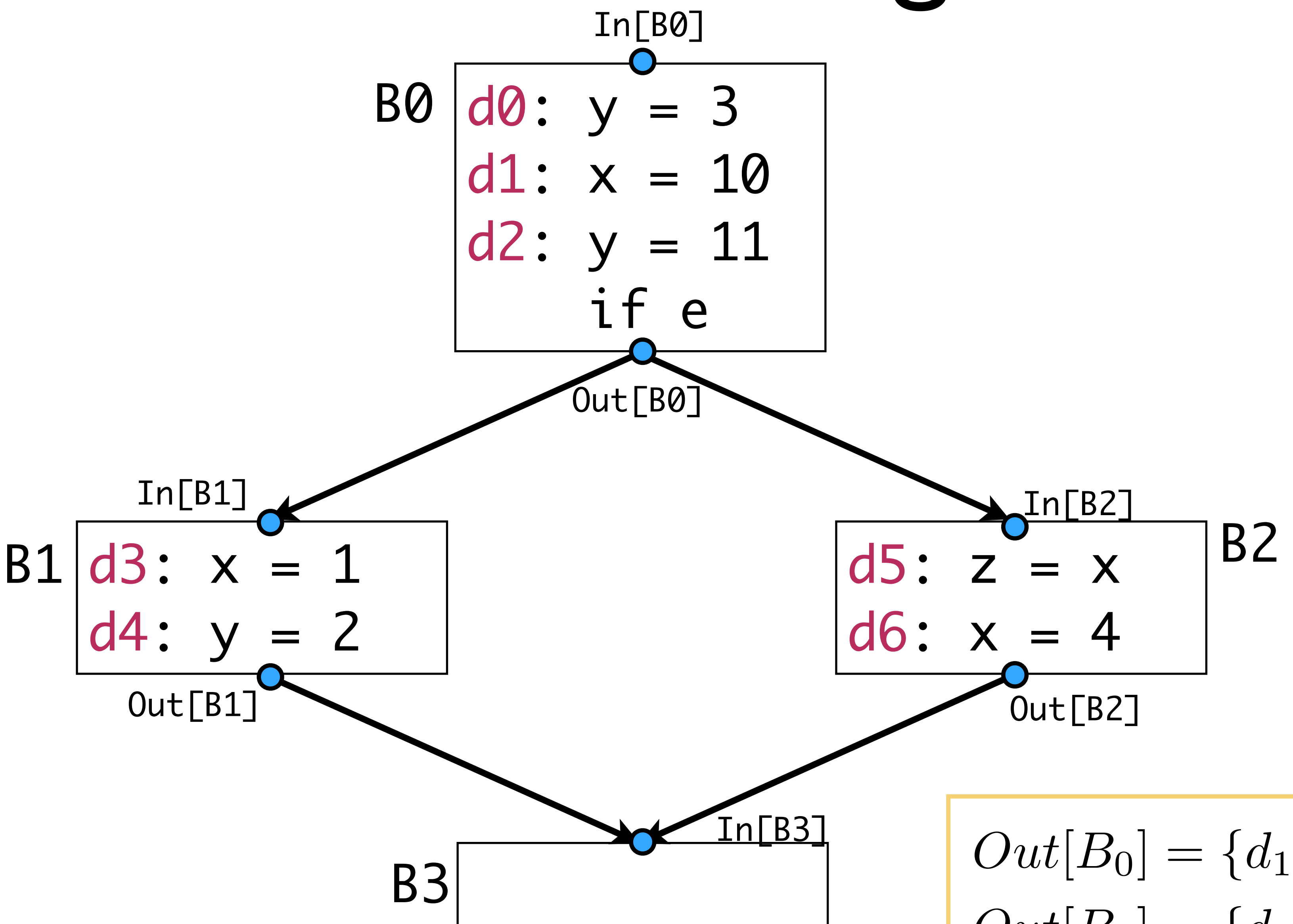


What `In[B0]` should be ?

$$In[B_0] = \{\}$$

It depends on the problem.

Extracting Constraints



Transfer Function

$$Out[B_0] = \{d_1, d_2\} \cup (In[B_0] - \{d_3, d_4, d_6, d_0\})$$

$$Out[B_1] = \{d_3, d_4\} \cup (In[B_1] - \{d_0, d_1, d_2, d_6\})$$

$$Out[B_2] = \{d_5, d_6\} \cup (In[B_2] - \{d_1, d_3\})$$

$$In[B_1] = Out[B_0]$$

$$In[B_2] = Out[B_0]$$

$$In[B_3] = Out[B_1] \cup Out[B_2]$$

Meet Operator

$$In[B_0] = \{\}$$

Boundary Condition

Solving Constraints

Transfer Function

$$Out[B_0] = \{d_1, d_2\} \cup (In[B_0] - \{d_3, d_4, d_6, d_0\})$$
$$Out[B_1] = \{d_3, d_4\} \cup (In[B_1] - \{d_0, d_1, d_2, d_6\})$$
$$Out[B_2] = \{d_5, d_6\} \cup (In[B_2] - \{d_1, d_3\})$$

$$In[B_1] = Out[B_0]$$
$$In[B_2] = Out[B_0]$$
$$In[B_3] = Out[B_1] \cup Out[B_2]$$

Meet Operator

$$In[B_0] = \{\}$$

Boundary Condition

Goal: Find $(In[B_0], In[B_1], In[B_2], In[B_3], Out[B_0], Out[B_1], Out[B_2])$ satisfying the above constraints.

Solving Constraints

Transfer Function

$$Out[B_0] = \{d_1, d_2\} \cup (In[B_0] - \{d_3, d_4, d_6, d_0\})$$
$$Out[B_1] = \{d_3, d_4\} \cup (In[B_1] - \{d_0, d_1, d_2, d_6\})$$
$$Out[B_2] = \{d_5, d_6\} \cup (In[B_2] - \{d_1, d_3\})$$

$$In[B_1] = Out[B_0]$$
$$In[B_2] = Out[B_0]$$
$$In[B_3] = Out[B_1] \cup Out[B_2]$$

Meet Operator

$$In[B_0] = \{\}$$

Boundary Condition

Goal: Find $(In[B_0], In[B_1], In[B_2], In[B_3], Out[B_0], Out[B_1], Out[B_2])$
satisfying the above constraints.

THIS IS A FIXED POINT PROBLEM

$$\mathcal{F}(In[B_0], In[B_1], In[B_2], In[B_3], Out[B_0], Out[B_1], Out[B_2]) =$$
$$(In[B_0], In[B_1], In[B_2], In[B_3], Out[B_0], Out[B_1], Out[B_2])$$

Solving Constraints

Transfer Function

$$Out[B_0] = \{d_1, d_2\} \cup (In[B_0] - \{d_3, d_4, d_6, d_0\})$$
$$Out[B_1] = \{d_3, d_4\} \cup (In[B_1] - \{d_0, d_1, d_2, d_6\})$$
$$Out[B_2] = \{d_5, d_6\} \cup (In[B_2] - \{d_1, d_3\})$$

$$In[B_1] = Out[B_0]$$
$$In[B_2] = Out[B_0]$$
$$In[B_3] = Out[B_1] \cup Out[B_2]$$

Meet Operator

$$In[B_0] = \{\}$$

Boundary Condition

Goal: Find $(In[B_0], In[B_1], In[B_2], In[B_3], Out[B_0], Out[B_1], Out[B_2])$ satisfying the above constraints.

THIS IS A FIXED POINT PROBLEM

$$\mathcal{F}(In[B_0], In[B_1], In[B_2], In[B_3], Out[B_0], Out[B_1], Out[B_2]) = (In[B_0], In[B_1], In[B_2], In[B_3], Out[B_0], Out[B_1], Out[B_2])$$

Do we have a solution for this problem? If it is, how can we compute it?...

Basic Fixpoint Theorems

from the last lecture...

If τ is monotonic, then it has a least fixpoint, **lfp** $Z\left[\tau(Z)\right]$, and a greatest fixpoint, **gfp** $Z\left[\tau(Z)\right]$.

lfp $Z\left[\tau(Z)\right] = \cap\{Z \mid \tau(Z) = Z\}$ whenever τ is monotonic.

lfp $Z\left[\tau(Z)\right] = \cup_i \tau^i(False)$ whenever τ is also $\cup$-continuous;

gfp $Z\left[\tau(Z)\right] = \cup\{Z \mid \tau(Z) = Z\}$ whenever τ is monotonic.

gfp $Z\left[\tau(Z)\right] = \cap_i \tau^i(True)$ whenever τ is also $\cap$-continuous.

Least Fixpoint Algorithm

from the last lecture...

As a consequence of the preceding lemmas, if τ is monotonic, its least fixpoint can be computed by the following program.

```
function Lfp(Tau : PredicateTransformer)
begin
    Q := False;
    Q' := Tau(Q);
    while (Q ≠ Q') do
    begin
        Q := Q';
        Q' := Tau(Q')
    end;
    returnQ
end
```

Solving Constraints

THIS IS A FIXED POINT PROBLEM

$$\mathcal{F}(In[B_0], In[B_1], In[B_2], In[B_3], Out[B_0], Out[B_1], Out[B_2]) = (In[B_0], In[B_1], In[B_2], In[B_3], Out[B_0], Out[B_1], Out[B_2])$$

Do we have a solution for this problem? If it is, how can we compute it? Can we get a solution within finite time?

Solving Constraints

THIS IS A FIXED POINT PROBLEM

$$\mathcal{F}(In[B_0], In[B_1], In[B_2], In[B_3], Out[B_0], Out[B_1], Out[B_2]) = (In[B_0], In[B_1], In[B_2], In[B_3], Out[B_0], Out[B_1], Out[B_2])$$

Do we have a solution for this problem? If it is, how can we compute it? Can we get a solution within finite time?

Yes, we have a solution only if $\mathcal{F}$ is monotone.

Yes, we can compute a solution only if $\mathcal{F}$ is continuous.

Yes, we can compute a solution in finite time only if we the domain has descending chain condition(DCC).

Solving Constraints

THIS IS A FIXED POINT PROBLEM

$$\mathcal{F}(In[B_0], In[B_1], In[B_2], In[B_3], Out[B_0], Out[B_1], Out[B_2]) = (In[B_0], In[B_1], In[B_2], In[B_3], Out[B_0], Out[B_1], Out[B_2])$$

Do we have a solution for this problem? If it is, how can we compute it? Can we get a solution within finite time?

Yes, we have a solution only if $\mathcal{F}$ is monotone.

Yes, we can compute a solution only if $\mathcal{F}$ is continuous.

Yes, we can compute a solution in finite time only if we the domain has descending chain condition(DCC).

$\mathcal{F} : V^7 \rightarrow V^7$ is monotone, continuous, and the domain has DCC.

Why?

Solving Constraints

THIS IS A FIXED POINT PROBLEM

$$\mathcal{F}(In[B_0], In[B_1], In[B_2], In[B_3], Out[B_0], Out[B_1], Out[B_2]) = (In[B_0], In[B_1], In[B_2], In[B_3], Out[B_0], Out[B_1], Out[B_2])$$

Do we have a solution for this problem? If it is, how can we compute it? Can we get a solution within finite time?

Yes, we have a solution only if $\mathcal{F}$ is monotone.

Yes, we can compute a solution only if $\mathcal{F}$ is continuous.

Algorithm

```
// Boundary Condition
In[B0] = { }

// Initialization for iterative algorithm
For each basic block B
        Out[B] = { }

// iterate
while(Changes to any In[], Out[] occur) {
        For each basic block B {
                In[B] = meet(Out[p_0], ... Out[p_1])
                Out[B] = f_B(In[B])
        }
}
```

lfp $Z\left[\tau(Z)\right] = \cup_i \tau^i(\underline{False})$ whenever τ is also $\cup$-continuous;

$$In[B] = \bigcup_{B':pred(B)} Out[B']$$

$$Out[s_0] = f_B(In[s_0])$$

Solving Constraints

```
// Boundary Condition
In[B0] = { }

// Initialization for iterative algorithm
For each basic block B
        Out[B] = { }

// iterate
while(Changes to any In[], Out[] occur) {
        For each basic block B {
                In[B] = meet(Out[p_0], ... Out[p_1])
                Out[B] = f_B(In[B])
        }
}
```

lfp $Z\left[\tau(Z)\right] = \cup_i \tau^i(\underline{False})$ whenever τ is also $\cup$-continuous;

$$In[B] = \bigcup_{B':pred(B)} Out[B']$$

$$Out[s_0] = f_B(In[s_0])$$

Does this algorithm terminate? If so, what is complexity?

Solving Constraints

```
// Boundary Condition
In[B0] = { }

// Initialization for iterative algorithm
For each basic block B
        Out[B] = { }

// iterate
while(Changes to any In[], Out[] occur) {
        For each basic block B {
                In[B] = meet(Out[p_0], ... Out[p_1])
                Out[B] = f_B(In[B])
        }
}
```

lfp $Z\left[\tau(Z)\right] = \cup_i \tau^i(\underline{False})$ whenever τ is also $\cup$-continuous;

$$In[B] = \bigcup_{B':pred(B)} Out[B']$$

$$Out[s_0] = f_B(In[s_0])$$

Does this algorithm terminate? If so, what is complexity?

Yes, why? Since $\mathcal{F} : V^7 \rightarrow V^7$ is monotone, and the domain has DCC property.

Summary: Reaching Definition

Reaching definition problem is defined by

- Domain of values: $V = 2^{\{d_0, d_1, d_2, d_3, d_4, d_5, d_6\}}$
- Meet operator: $\cup : V \rightarrow V$
- Boundary Condition: $In[B_0] = \{\}$
- Set of Transfer Functions:

$$\{f_{B_0}, f_{B_1}, f_{B_2}, f_{B_3}, f_{B_4}, f_{B_5}, f_{B_6}\} : 2^{V \rightarrow V}$$

A Unified Framework

Data flow problems are defined by

- Domain of values: V — meet-semilattice is enough.
- Meet operator: $\prod : V \to V$
- Boundary Condition: value of entry/exit node
- Set of Transfer Functions: $2^{V \to V}$

Meet-semilattice

Reminder

A set S partially ordered by the binary relation $\sqsubseteq$ if for all x, y, z:

reflexive	$x \sqsubseteq x$
anti-symmetric	$x \sqsubseteq y \wedge y \sqsubseteq x \implies x = y$
transitive	$x \sqsubseteq y \wedge y \sqsubseteq z \implies x \sqsubseteq z$

A poset S is meet-semilattice if for all elements x and y of S, the greatest lower bound of the set {x, y} exists.

$$\forall x, y \in S : \exists \sqcap \{x, y\} \in S$$

Semi-lattice has the top element.

$$\exists \top \in S : \forall x \in S : x \sqsubseteq \top$$

Meet-semilattice

- Example: ({d0, d1, d2, d3, d4, d5, d6}, $\supseteq$)
  What is $\sqcap$ and $\top$?

General Iterative Data Flow Analysis Algorithm (Forward)

```
// Boundary Condition
Out[Entry] = V_Entry

// Initialization for iterative algorithm
For each basic block B
        Out[B] = Top

// iterate
while(Changes to any Out[], In[] occur) {
        For each basic block B {
                In[B] = meet(Out[p_0], ... Out[p_1])
                Out[B] = f_B(In[B])
        }
}
```

Thank you