

COSE419: Software Verification

Lecture 11 – Static Analysis Overview

Hakjoo Oh
2024 Spring

Principles of Static Analysis

$$30 \times 12 + 11 \times 9 = ?$$

Principles of Static Analysis

$$30 \times 12 + 11 \times 9 = ?$$

- Dynamic analysis (testing): 459

Principles of Static Analysis

$$30 \times 12 + 11 \times 9 = ?$$

- Dynamic analysis (testing): 459
- Static analysis: a variety of answers
 - "integer"
 - "odd integer"
 - "positive integer"
 - "integer between 400 and 500"
 - ...

Principles of Static Analysis

$$30 \times 12 + 11 \times 9 = ?$$

- Dynamic analysis (testing): 459
- Static analysis: a variety of answers
 - "integer"
 - "odd integer" 1. Choose abstract value (domain)
 - "positive integer"
 - "integer between 400 and 500"
 - ...

Principles of Static Analysis

$$30 \times 12 + 11 \times 9 = ?$$

- Dynamic analysis (testing): 459
- Static analysis: a variety of answers
 - "integer"
 - "odd integer" "odd integer"
 - "positive integer"
 - "integer between 400 and 500"
 - ...

2. "Execute" the program with abstract values

$$e \hat{\times} e \hat{+} o \hat{\times} o = o$$

$$e \hat{\times} e = e \quad e \hat{+} e = e$$

$$e \hat{\times} o = e \quad e \hat{+} o = o$$

$$o \hat{\times} e = e \quad o \hat{+} e = o$$

$$o \hat{\times} o = o \quad o \hat{+} o = e$$

1. Choose abstract value (domain)

Strength of Static Analysis

- By contrast to testing, static analysis can prove the absence of bugs

```
void f (int x) {  
    y = x * 12 + 9 * 11;  
    assert (y % 2 == 1);  
}
```

Strength of Static Analysis

- By contrast to testing, static analysis can prove the absence of bugs

```
void f (int x) {  
    y = x * 12 + 9 * 11;  
    assert (y % 2 == 1);  
}
```

T (don't know)

Strength of Static Analysis

- By contrast to testing, static analysis can prove the absence of bugs

```
void f (int x) {  
    y = x * 12 + 9 * 11;  
    assert (y % 2 == 1);  
}
```

Even

T (don't know)

Strength of Static Analysis

- By contrast to testing, static analysis can prove the absence of bugs

```
void f (int x) {  
    y = x * 12 + 9 * 11;  
    assert (y % 2 == 1);  
}
```

Even

T (don't know)

Odd

Strength of Static Analysis

- By contrast to testing, static analysis can prove the absence of bugs

```
void f (int x) {  
    y = x * 12 + 9 * 11;  
    assert (y % 2 == 1);  
}
```

The diagram illustrates the static analysis of a function. The function `f` takes an integer `x` and calculates `y = x * 12 + 9 * 11`. The expression `x * 12` is labeled "Even", `9 * 11` is labeled "Odd", and the entire expression is labeled "T (don't know)". The assertion `(y % 2 == 1)` is labeled "Odd".

Strength of Static Analysis

- By contrast to program verification, static analysis can prove the absence of bugs automatically

```
@pre: n >= 0
@post: rv == n
int SimpleWhile (int n) {
    int i = 0;
    while
    @L: 0 <= i <= n
    (i < n) {
        i = i + 1;
    }
}
```

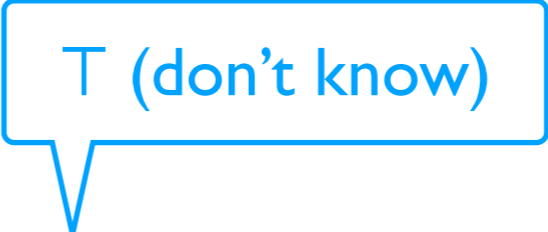
Weakness of Static Analysis

- Instead, static analysis may produce false alarms

```
void f (int x) {  
    y = x + x;  
    assert (y % 2 == 0) ;  
}
```

Weakness of Static Analysis

- Instead, static analysis may produce false alarms



```
void f (int x) {  
    y = x + x;  
    assert (y % 2 == 0);  
}
```

Weakness of Static Analysis

- Instead, static analysis may produce false alarms

```
void f (int x) {  
    y = x + x;  
    assert (y % 2 == 0);  
}
```

T (don't know)

T (don't know)

Weakness of Static Analysis

- Instead, static analysis may produce false alarms

```
void f (int x) {  
    y = x + x;  
    assert (y % 2 == 0);  
}
```

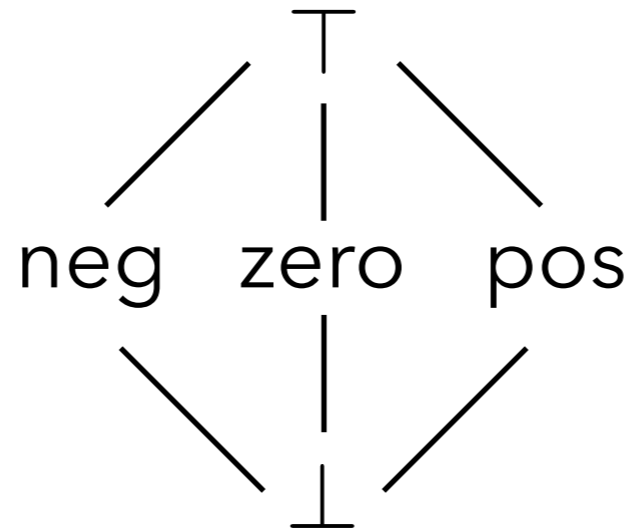
T (don't know)

T (don't know)

false alarm

A Simple Sign Domain

- Abstract values



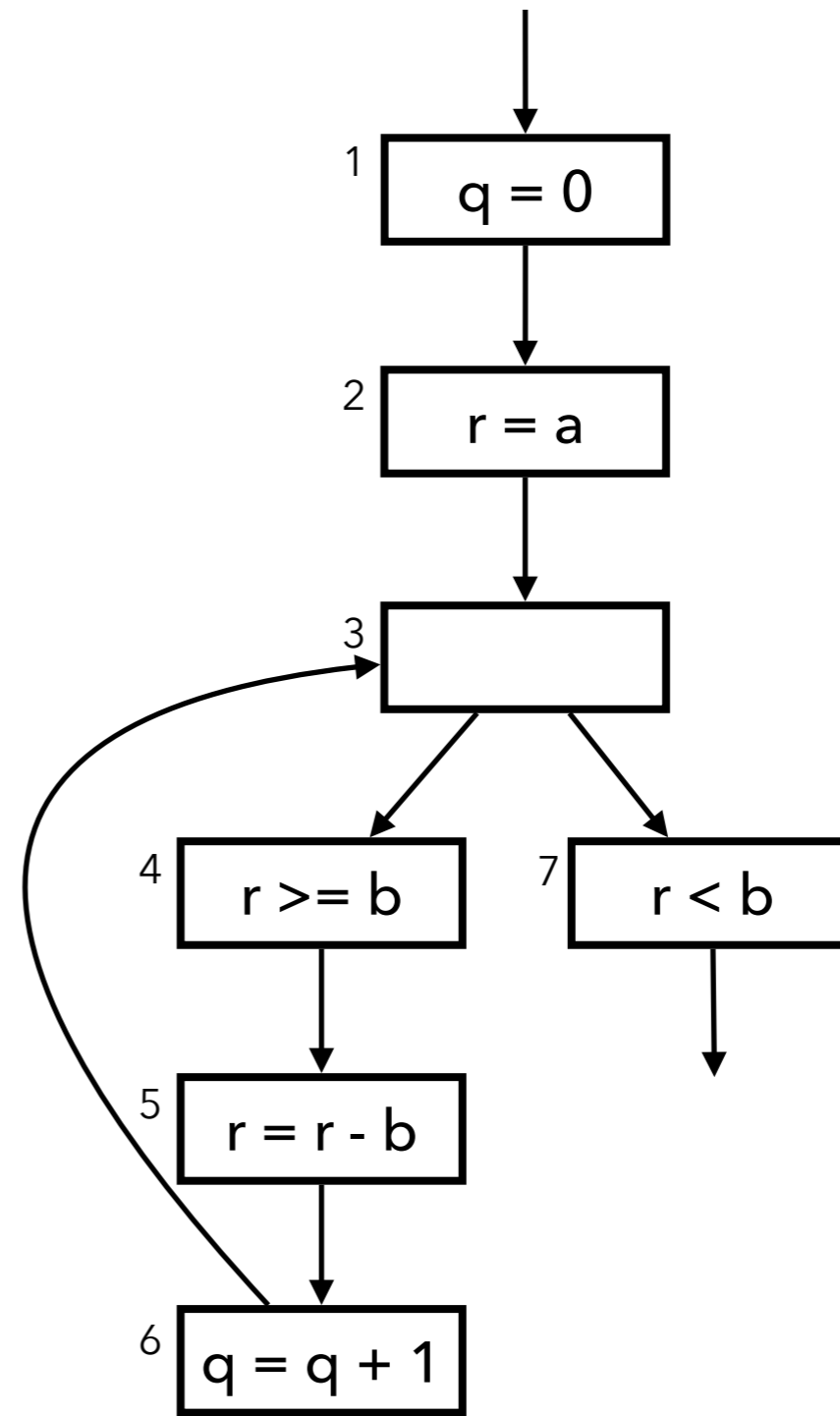
- Abstract operators

+/-	top	neg	zero	pos	bot
top					
neg					
zero					
pos					
bot					

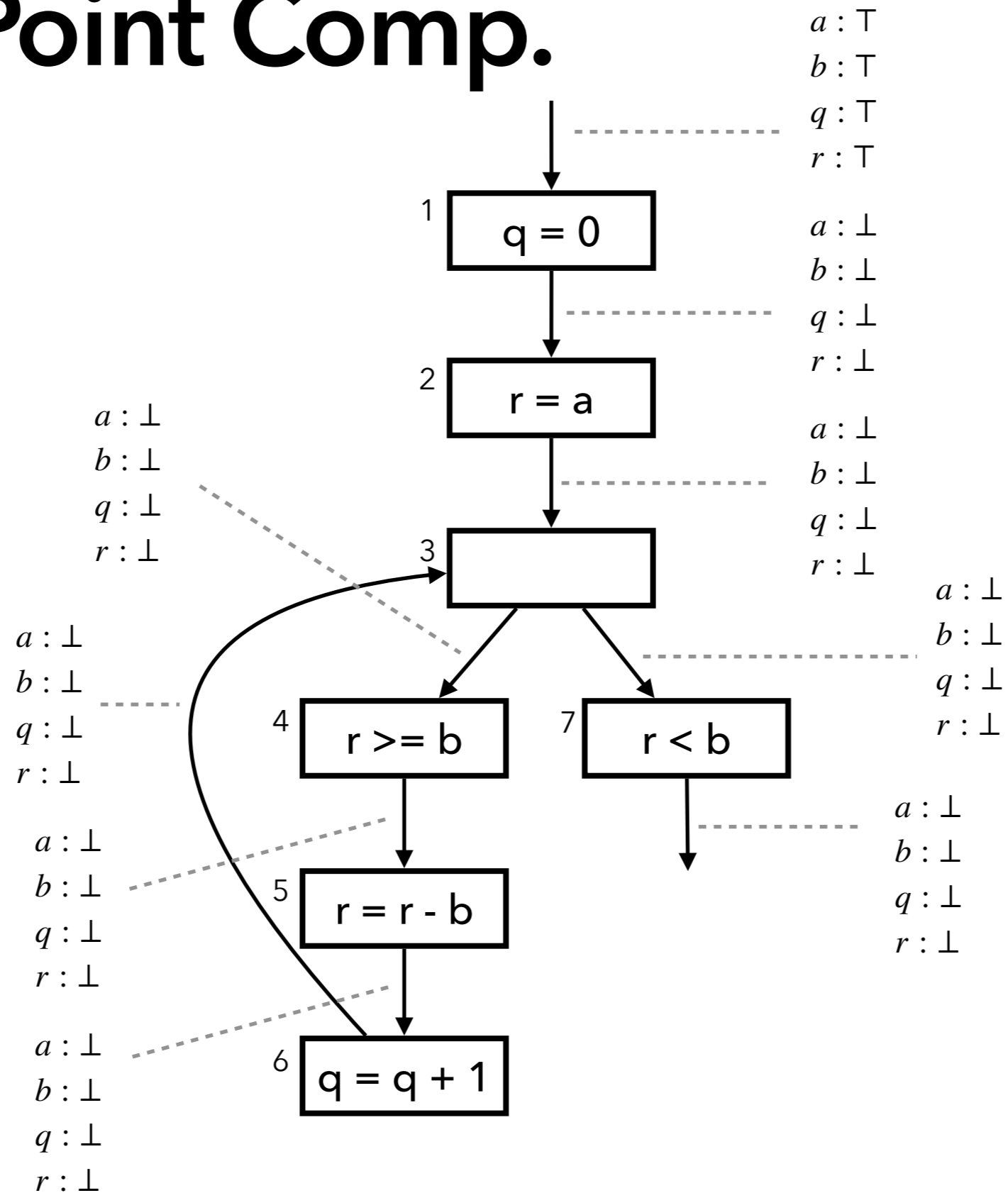
×	top	neg	zero	pos	bot
top					
neg					
zero					
pos					
bot					

Example Program

```
// a >= 0, b >= 0
q = 0;
r = a;
while (r >= b) {
    r = r - b;
    q = q + 1;
}
assert(q >= 0);
assert(r >= 0);
```

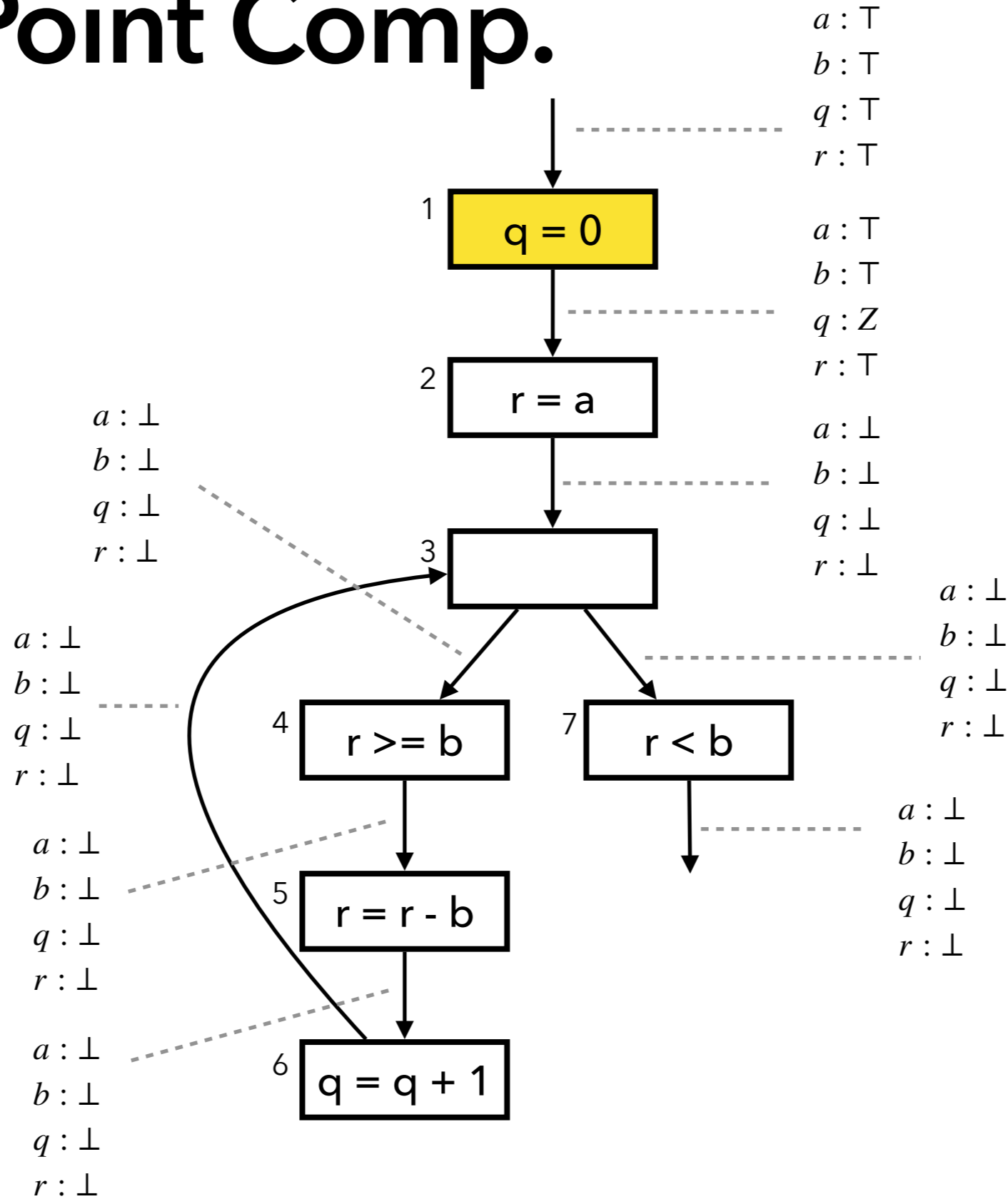


Fixed Point Comp.



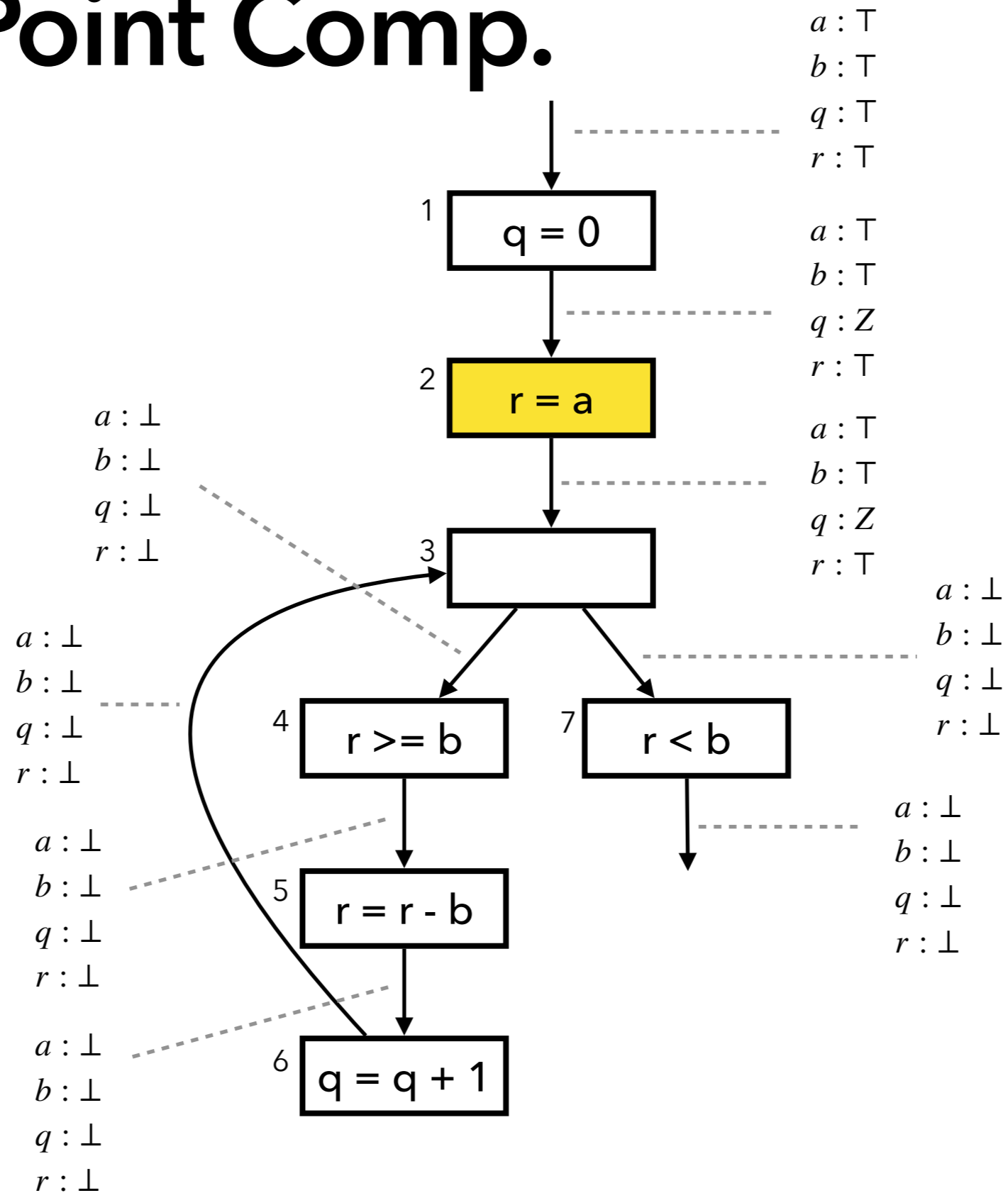
$$W = \{ 1, 2, 3, 4, 5, 6, 7 \}$$

Fixed Point Comp.



$$W = \{ 1, 2, 3, 4, 5, 6, 7 \}$$

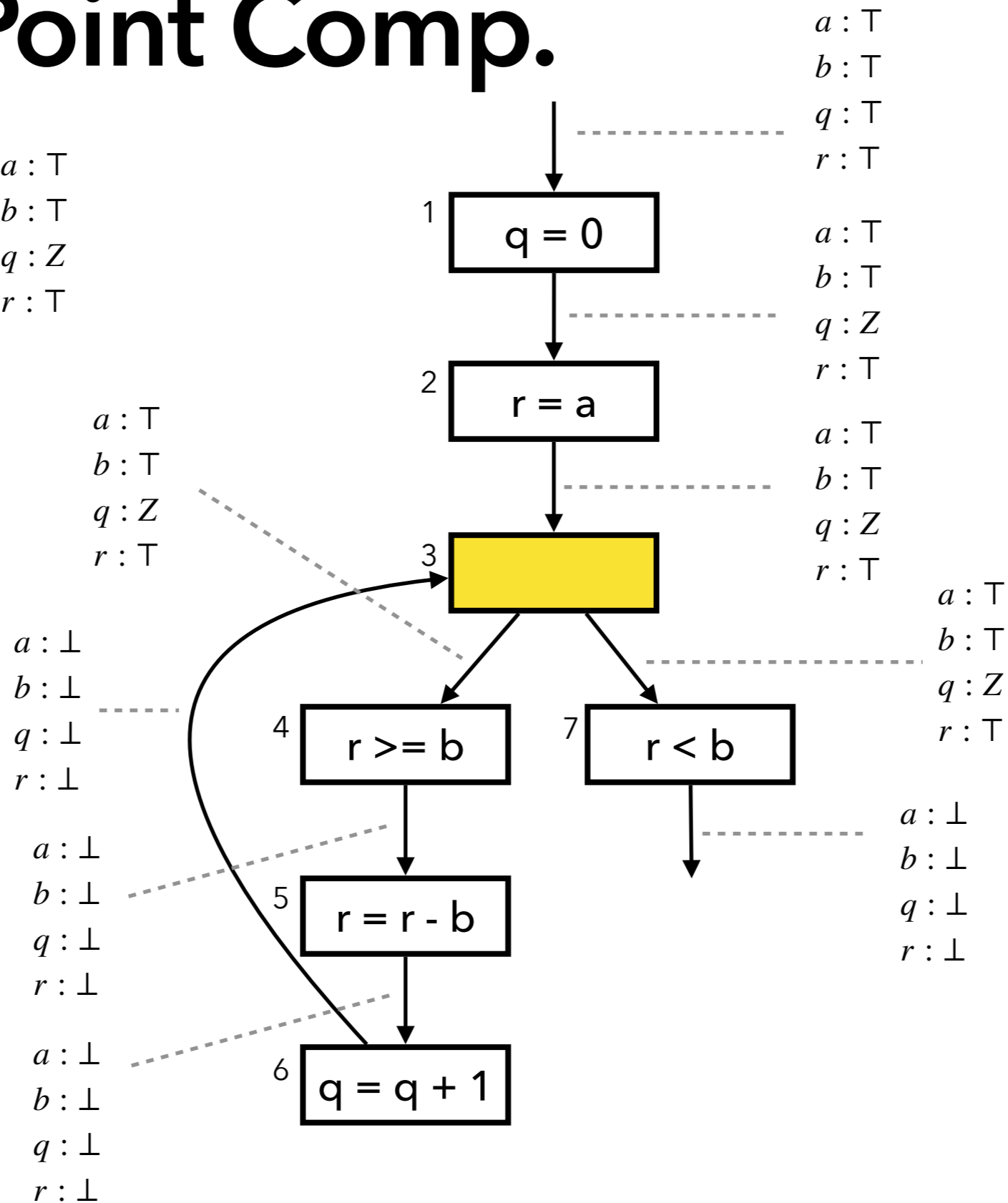
Fixed Point Comp.



$$W = \{ 1, 2, 3, 4, 5, 6, 7 \}$$

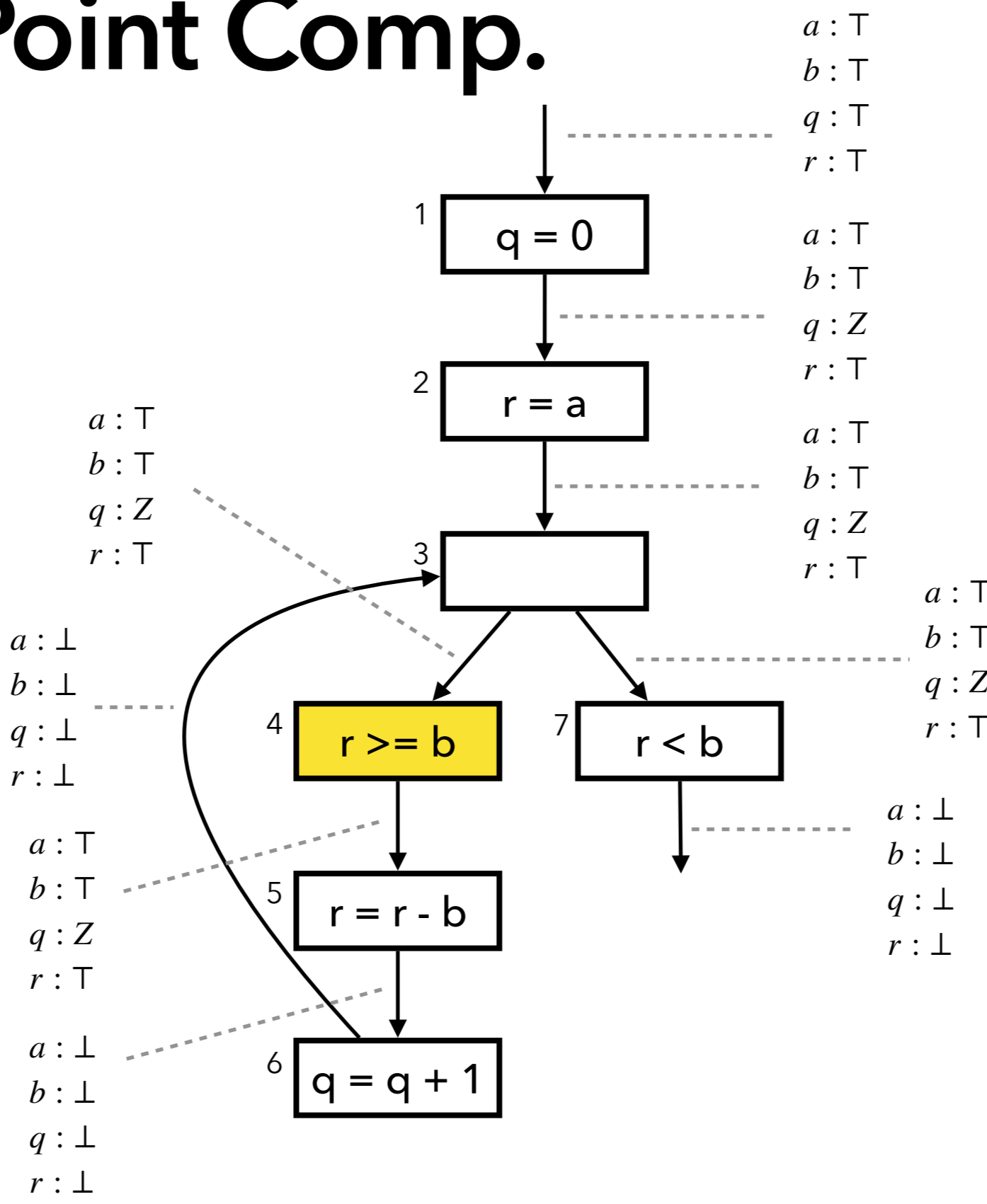
Fixed Point Comp.

$$\begin{array}{l}
 a : \top \\
 b : \top \\
 q : \mathbb{Z} \\
 r : \top
 \end{array}
 \sqcup
 \begin{array}{l}
 a : \perp \\
 b : \perp \\
 q : \perp \\
 r : \perp
 \end{array}
 =
 \begin{array}{l}
 a : \top \\
 b : \top \\
 q : \mathbb{Z} \\
 r : \top
 \end{array}$$



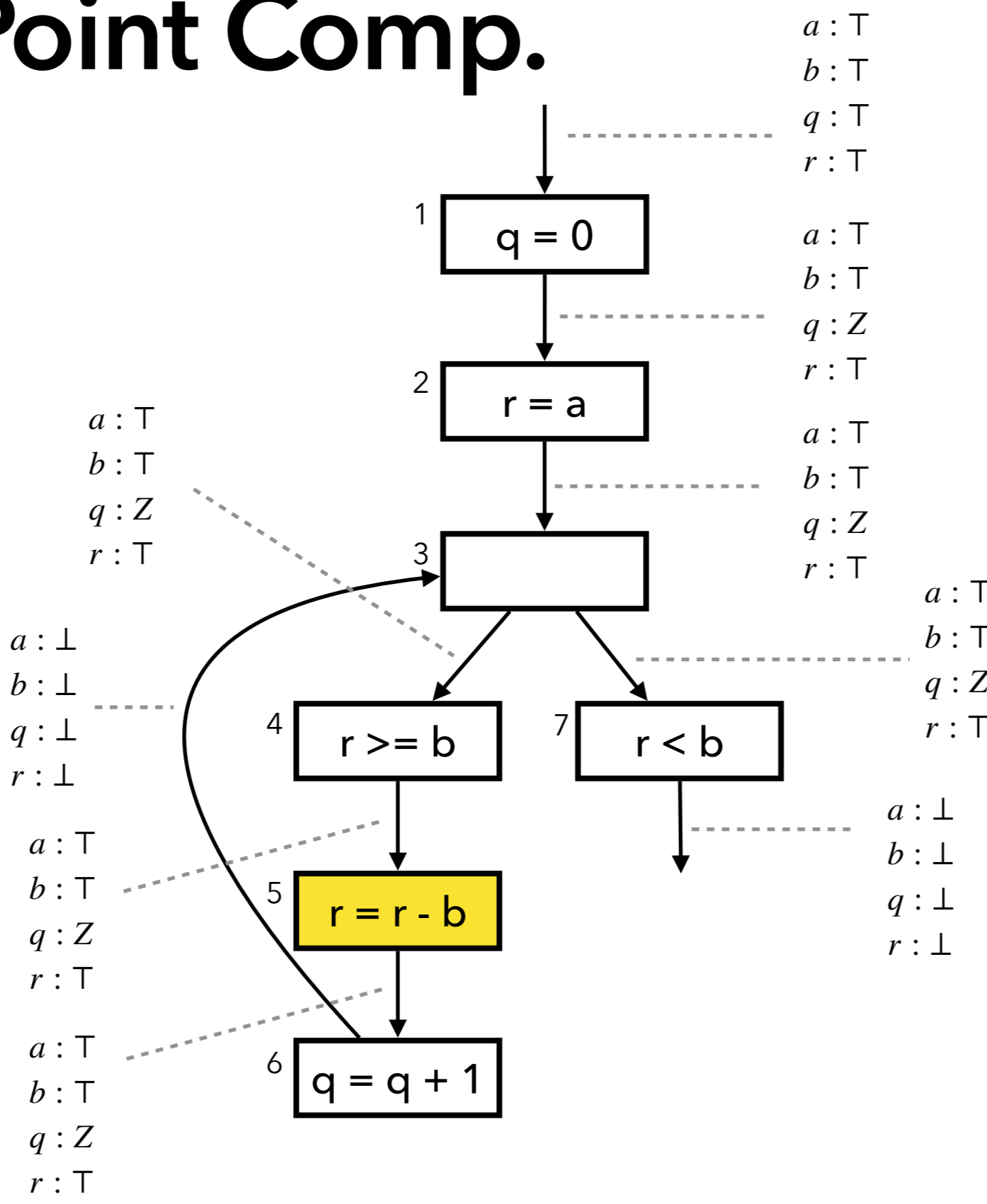
$$W = \{1, 2, 3, 4, 5, 6, 7\}$$

Fixed Point Comp.



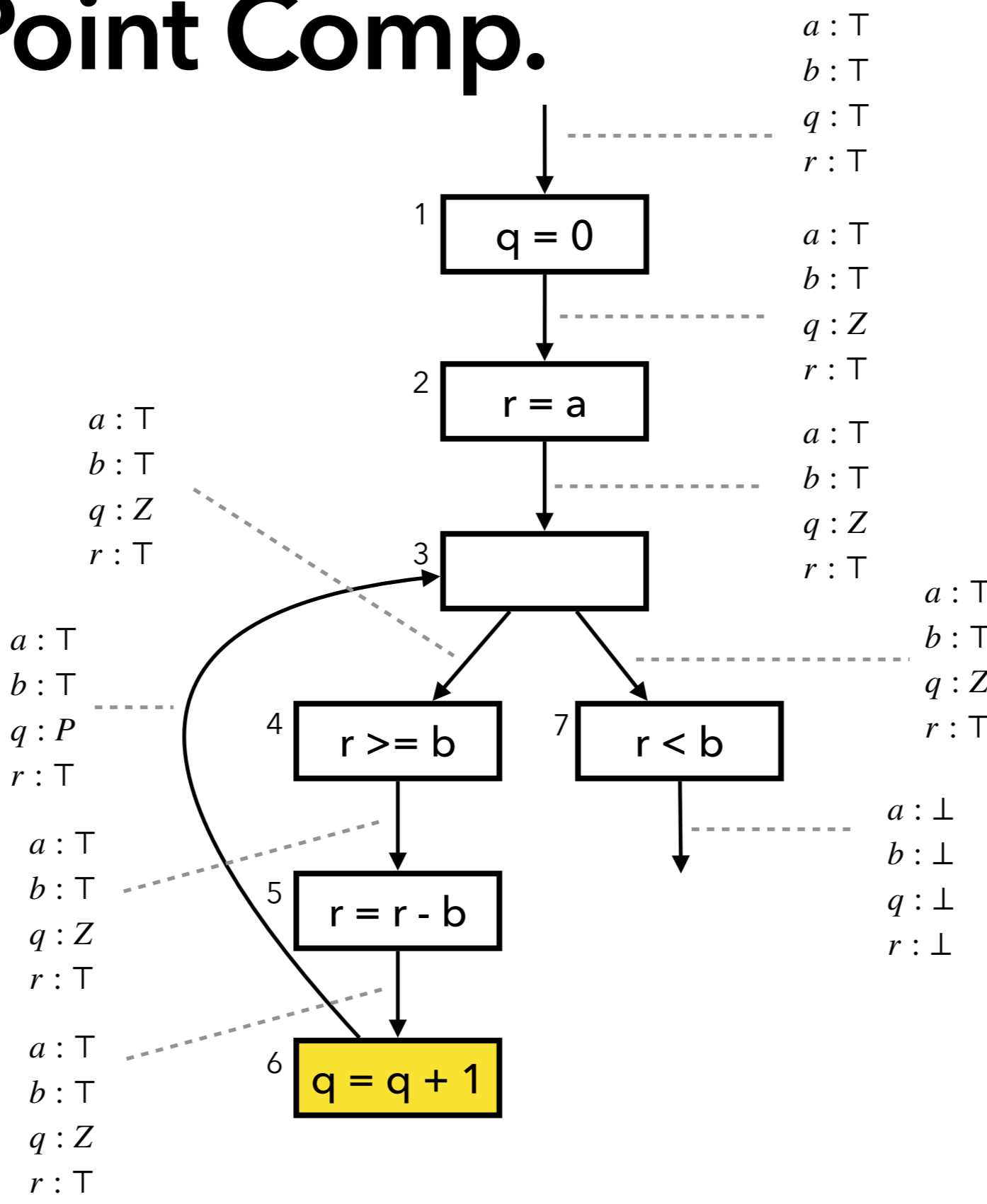
$$W = \{ 1, 2, 3, 4, 5, 6, 7 \}$$

Fixed Point Comp.



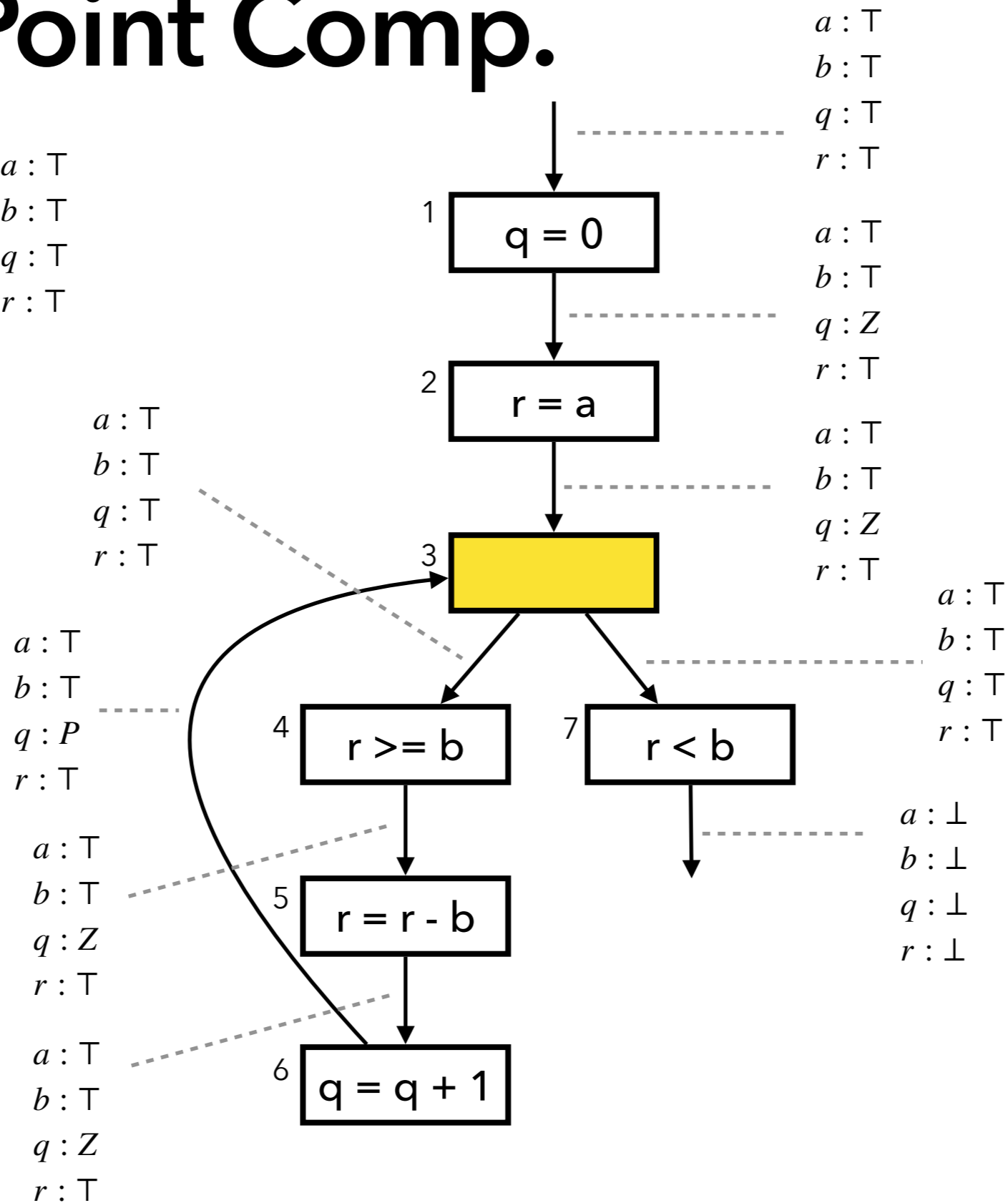
$$W = \{ 1, 2, 3, 4, 5, 6, 7 \}$$

Fixed Point Comp.



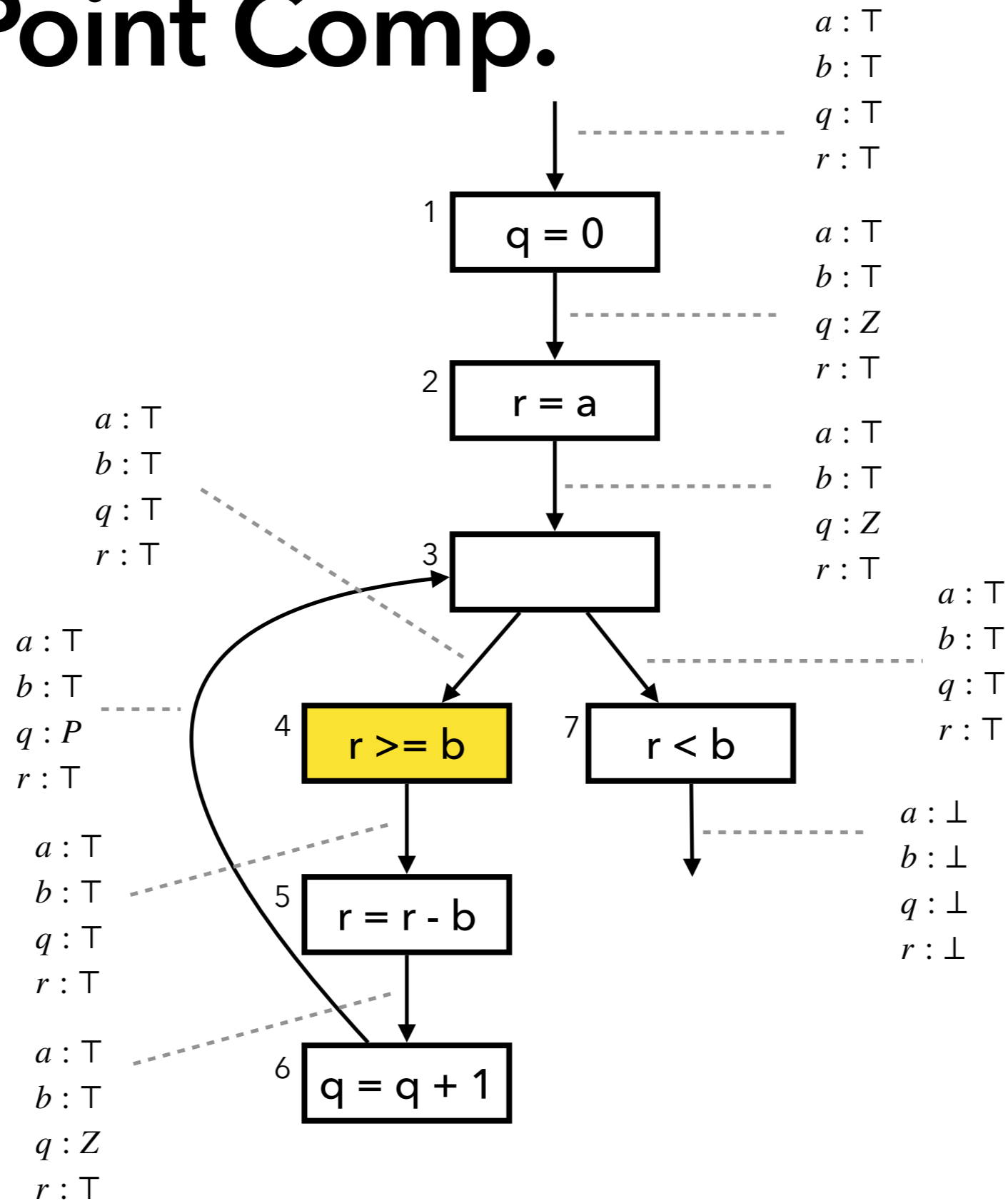
$$W = \{ 1, 2, 3, 4, 5, 6, 7 \}$$

Fixed Point Comp.

$$\begin{array}{l} a : \top \\ b : \top \\ q : \mathbb{Z} \\ r : \top \end{array} \sqcup \begin{array}{l} a : \top \\ b : \top \\ q : P \\ r : \top \end{array} = \begin{array}{l} a : \top \\ b : \top \\ q : \top \\ r : \top \end{array}$$


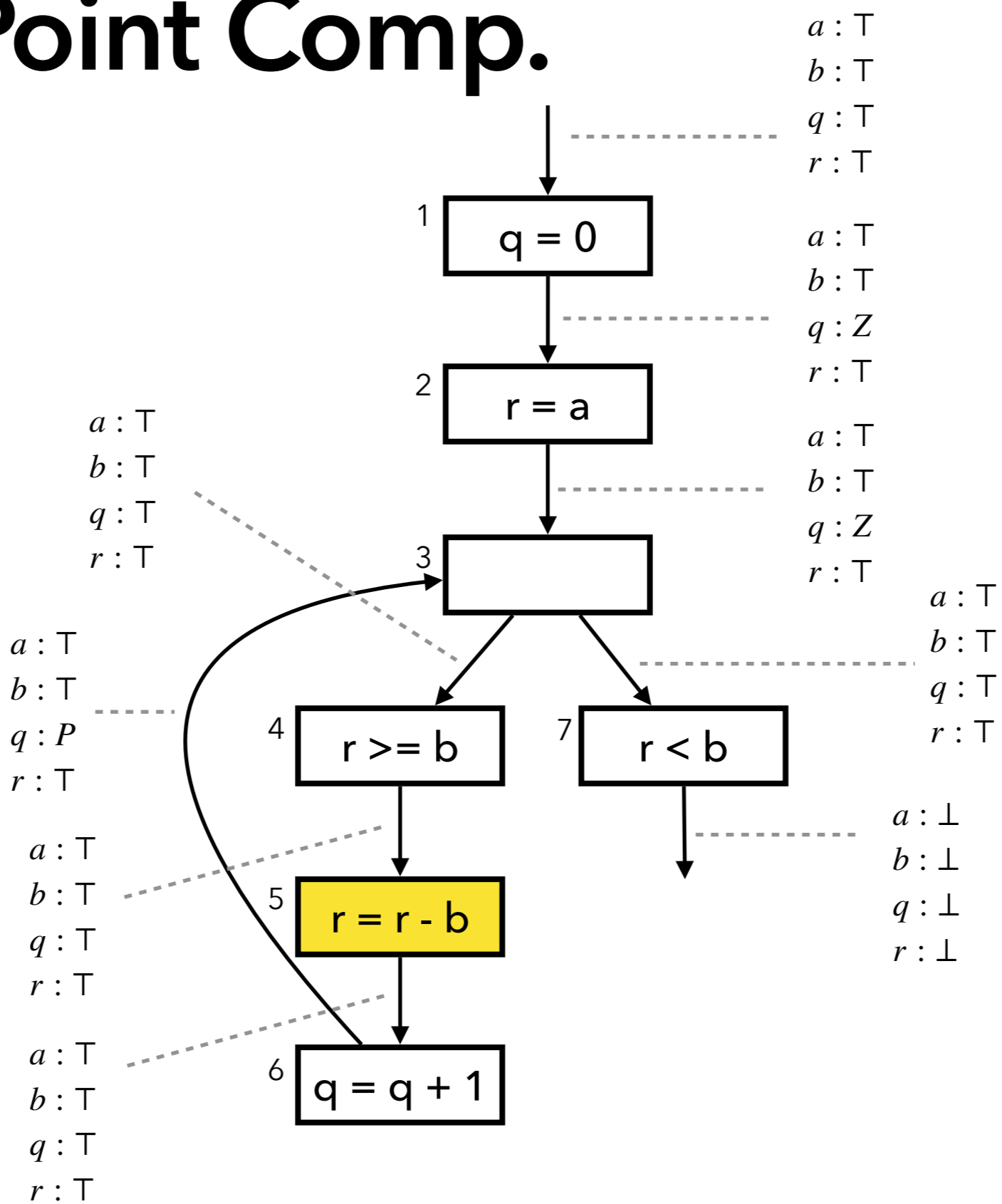
$W = \{ 1, 2, 3, 4, 5, 6, 7 \}$

Fixed Point Comp.



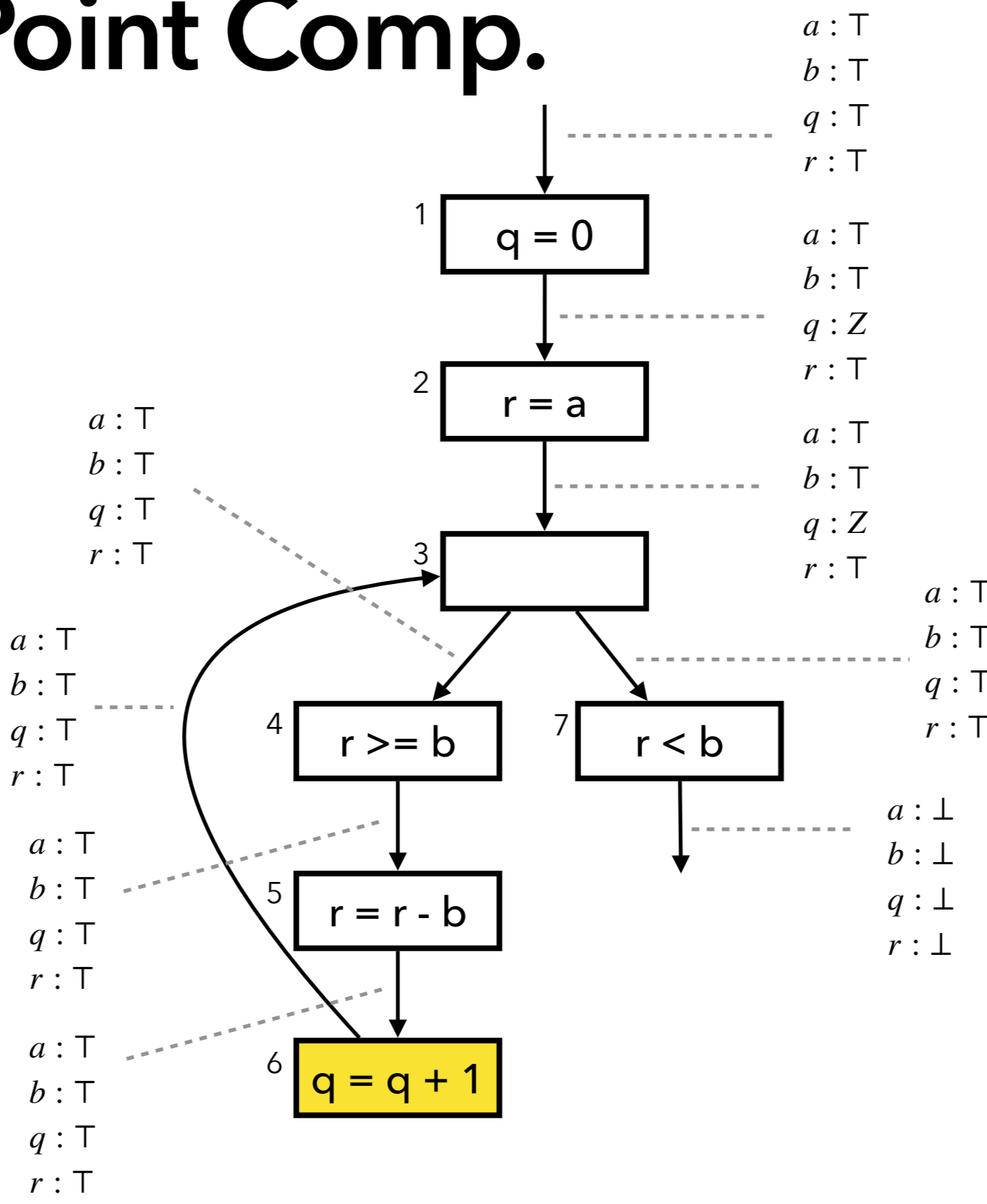
$$W = \{ 1, 2, 3, 4, 5, 6, 7 \}$$

Fixed Point Comp.



$$W = \{ 1, 2, 3, 4, 5, 6, 7 \}$$

Fixed Point Comp.



$$W = \{ 1, 2, 3, 4, 5, 6, 7 \}$$

Fixed Point Comp.

$a : \top$	$a : \top$	$a : \top$
$b : \top$	$b : \top$	$b : \top$
$q : \mathbb{Z}$	$q : \top$	$q : \top$
$r : \top$	$r : \top$	$r : \top$

(fixed point)

$a : \top$
 $b : \top$
 $q : \top$
 $r : \top$

$a : \top$
 $b : \top$
 $q : \top$
 $r : \top$

$a : \top$
 $b : \top$
 $q : \top$
 $r : \top$

$a : \top$
 $b : \top$
 $q : \top$
 $r : \top$

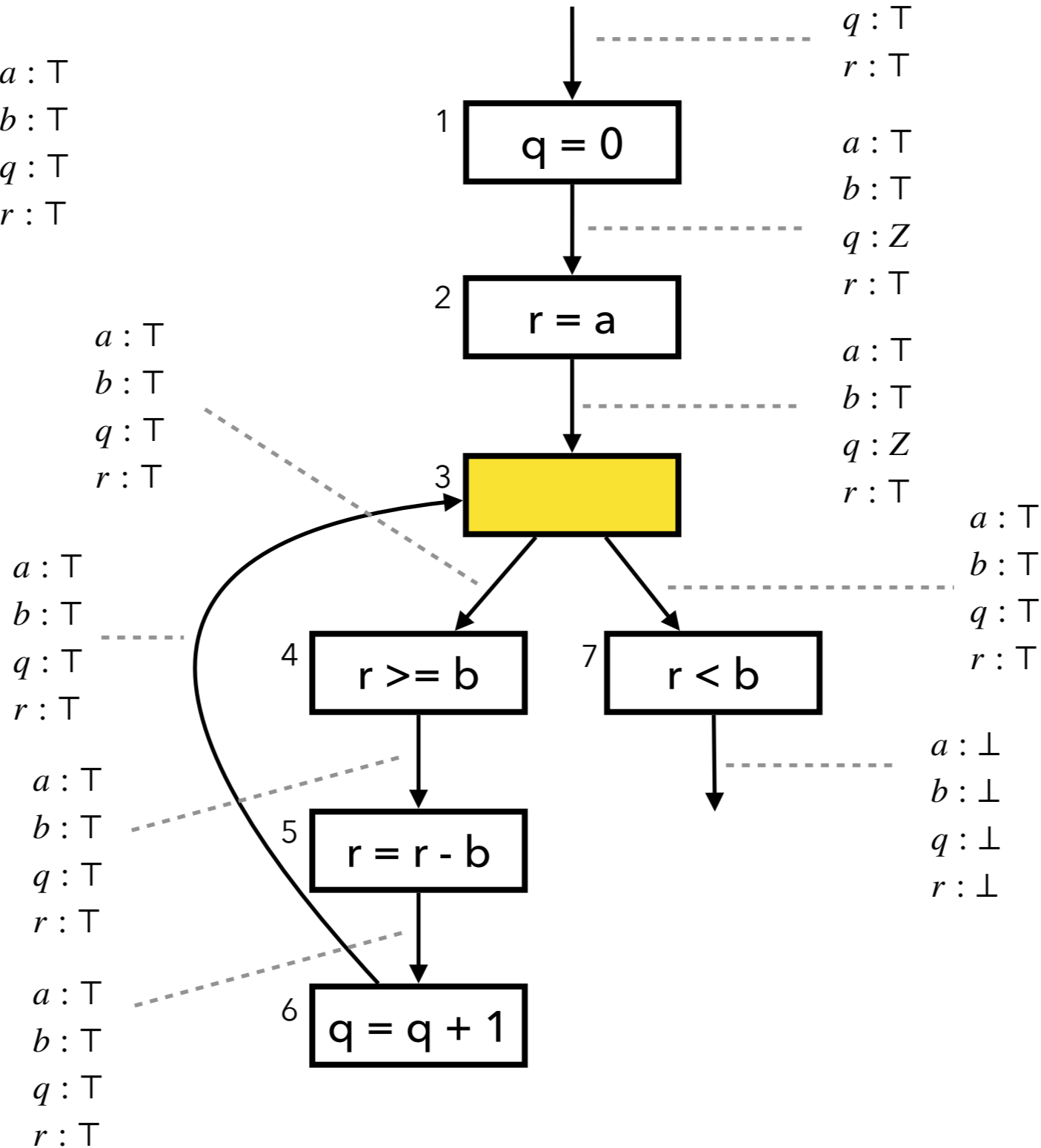
$a : \top$
 $b : \top$
 $q : \top$
 $r : \top$

$a : \top$
 $b : \top$
 $q : \mathbb{Z}$
 $r : \top$

$a : \top$
 $b : \top$
 $q : \mathbb{Z}$
 $r : \top$

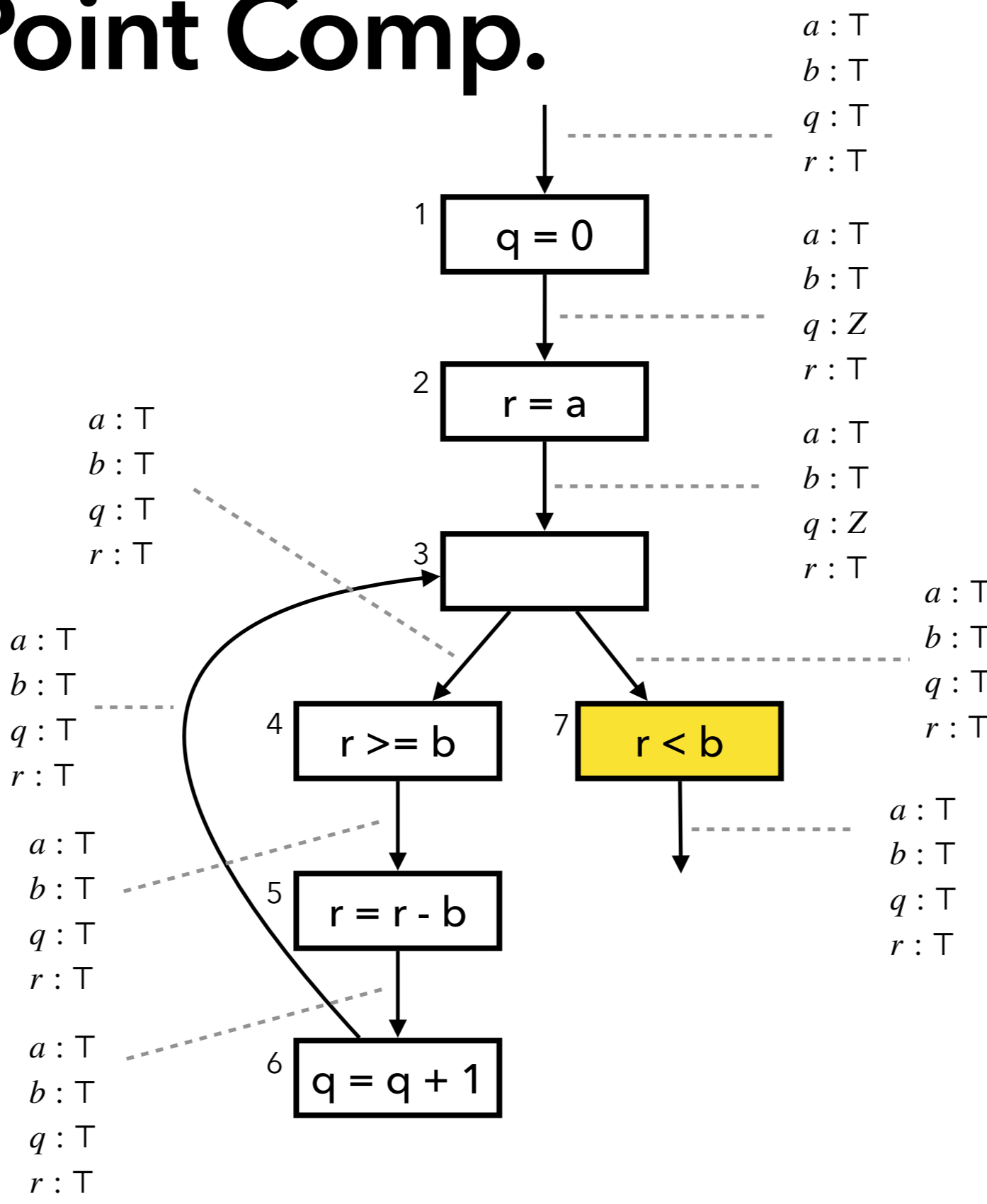
$a : \top$
 $b : \top$
 $q : \top$
 $r : \top$

$a : \perp$
 $b : \perp$
 $q : \perp$
 $r : \perp$



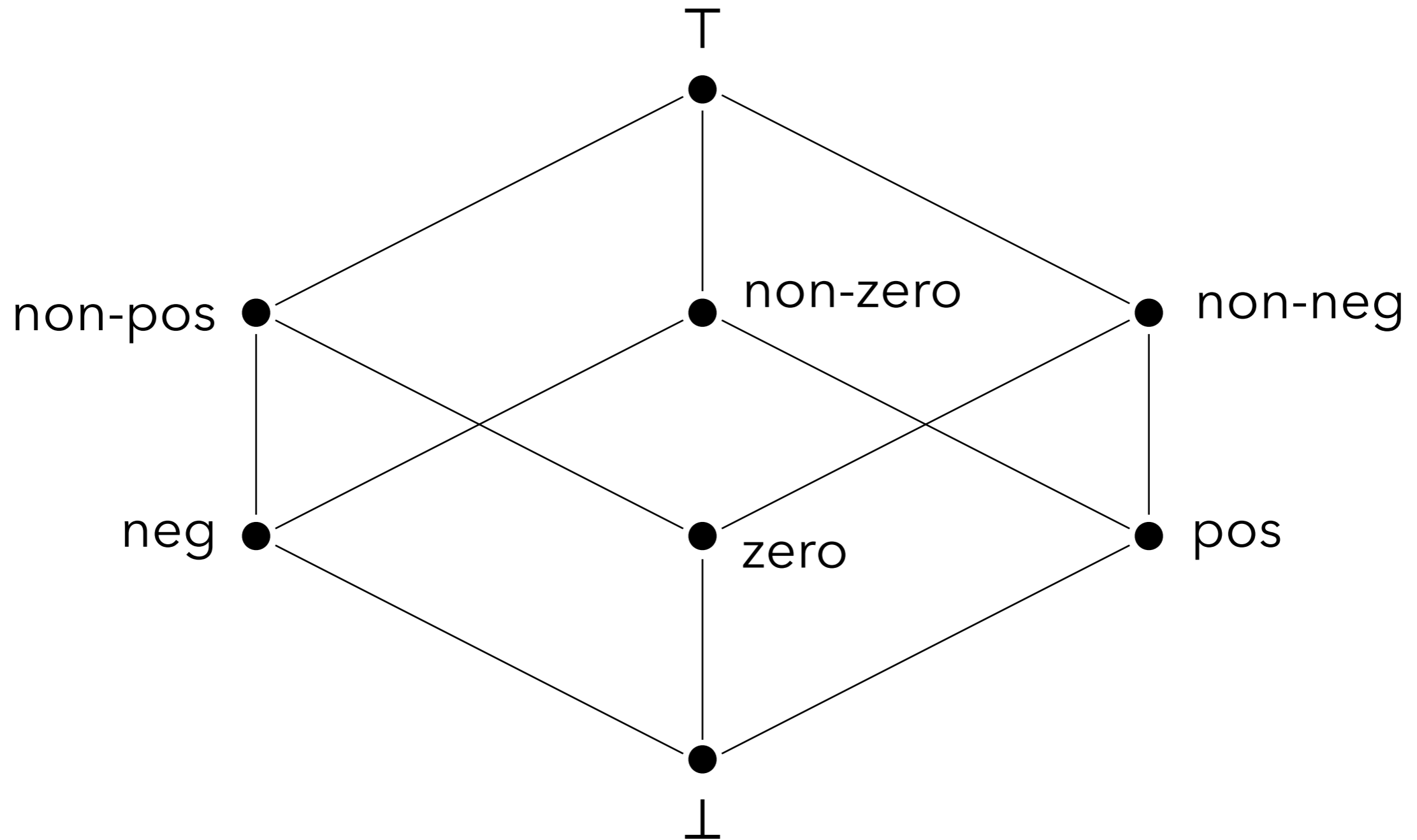
$W = \{ 1, 2, 3, 4, 5, 6, 7 \}$

Fixed Point Comp.



$$W = \{ 1, 2, 3, 4, 5, 6, 7 \}$$

An Extended Sign Domain



+	top	neg	zero	pos	non-pos	non-zero	non-neg	bot
top								
neg								
zero								
pos								
non-pos								
non-zero								
non-neg								
bot								

—	top	neg	zero	pos	non-pos	non-zero	non-neg	bot
top								
neg								
zero								
pos								
non-pos								
non-zero								
non-neg								
bot								

Exercise (1)

Describe the result of the analysis with the extended sign domain

```
// a >= 0, b >= 0
q = 0;
r = a;
while (r >= b) {
    r = r - b;
    q = q + 1;
}
assert (q >= 0);
assert (r >= 0);
```

