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"
  - "positive integer"
  - "integer between 400 and 500"
  - …

1. Choose abstract value (domain)

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"
  - …

1. Choose abstract value (domain)

2. "Execute" the program with abstract values

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

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

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

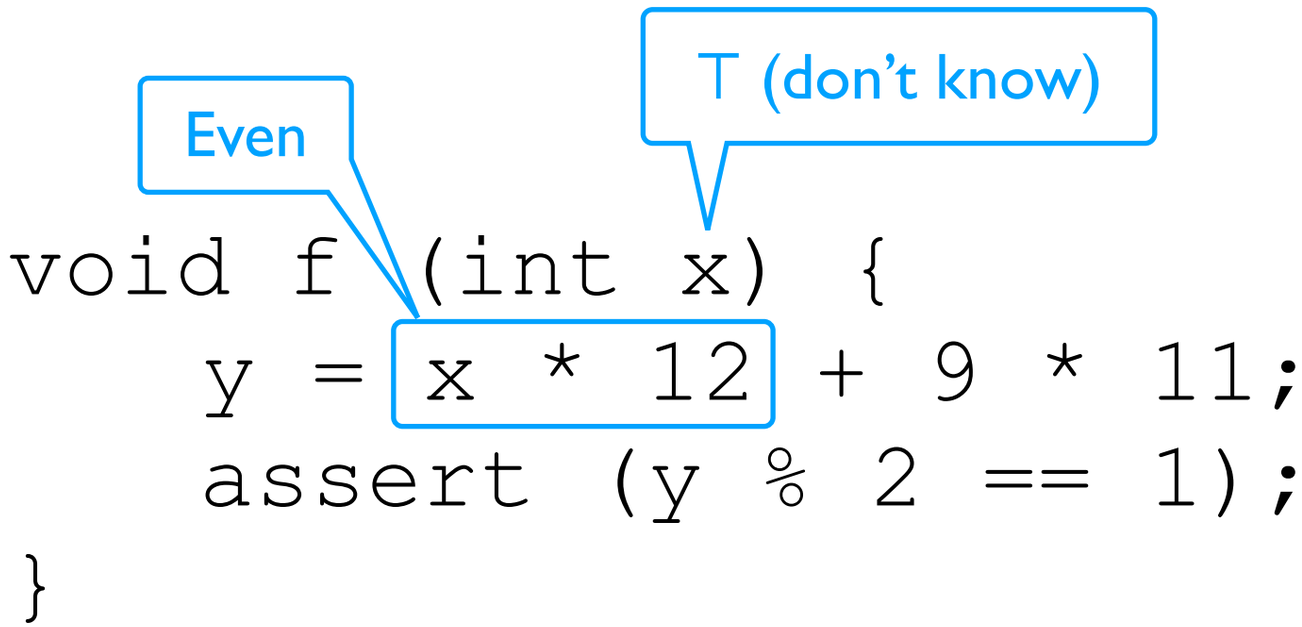
- By contrast to testing, static analysis can prove the absence of bugs

T (don't know)

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

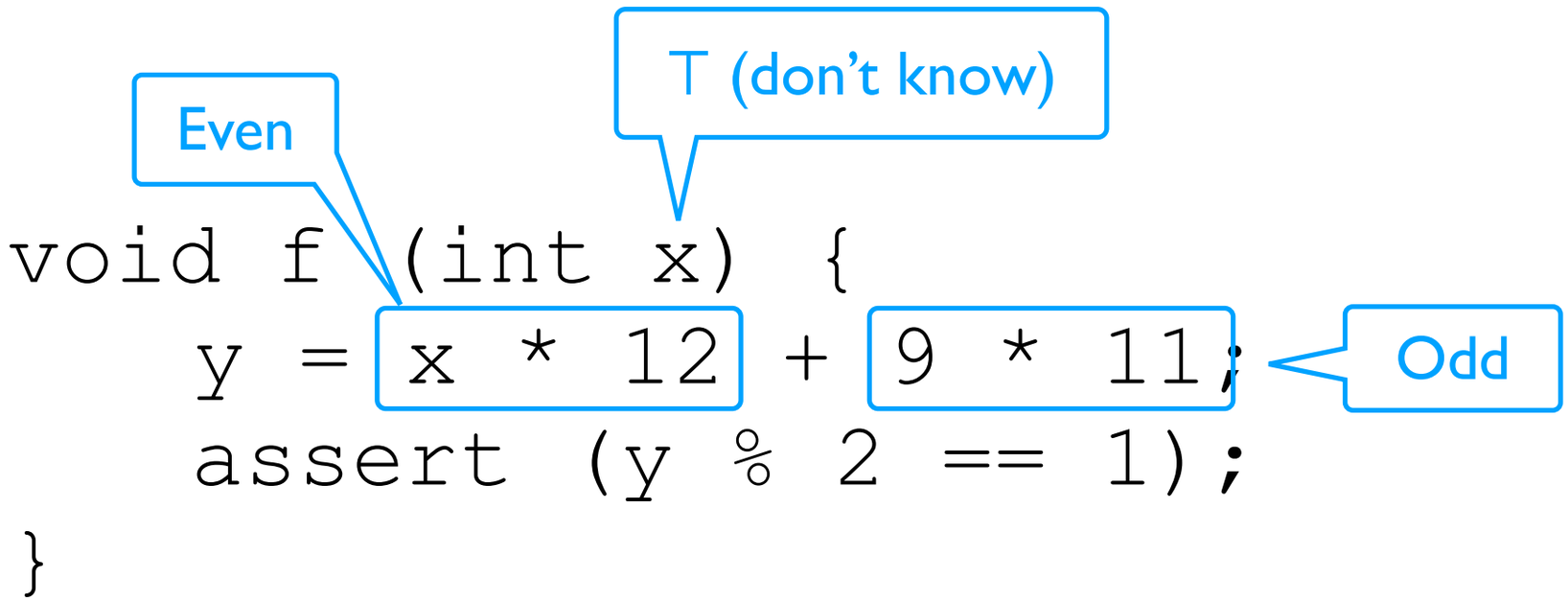
Strength of Static Analysis

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



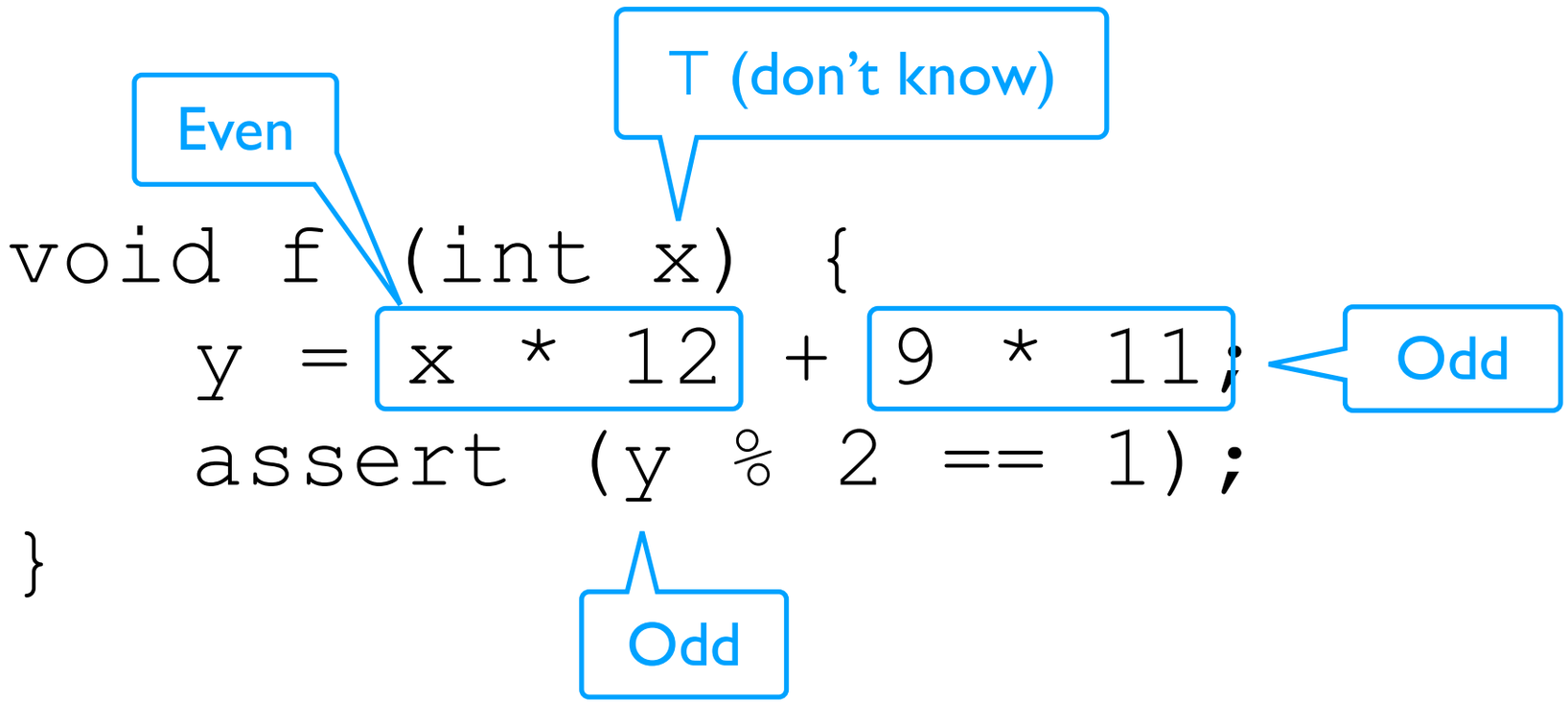
Strength of Static Analysis

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



Strength of Static Analysis

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



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

T (don't know)

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

Weakness of Static Analysis

- Instead, static analysis may produce false alarms

T (don't know)

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

T (don't know)

Weakness of Static Analysis

- Instead, static analysis may produce false alarms

T (don't know)

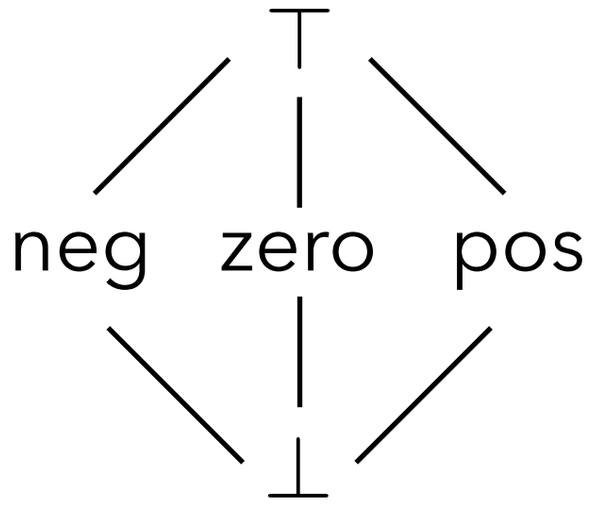
```
void f (int x) {
  y = x + x;
  assert (y % 2 == 0);
}
```

T (don't know)

false alarm

A Simple Sign Domain

- Abstract values



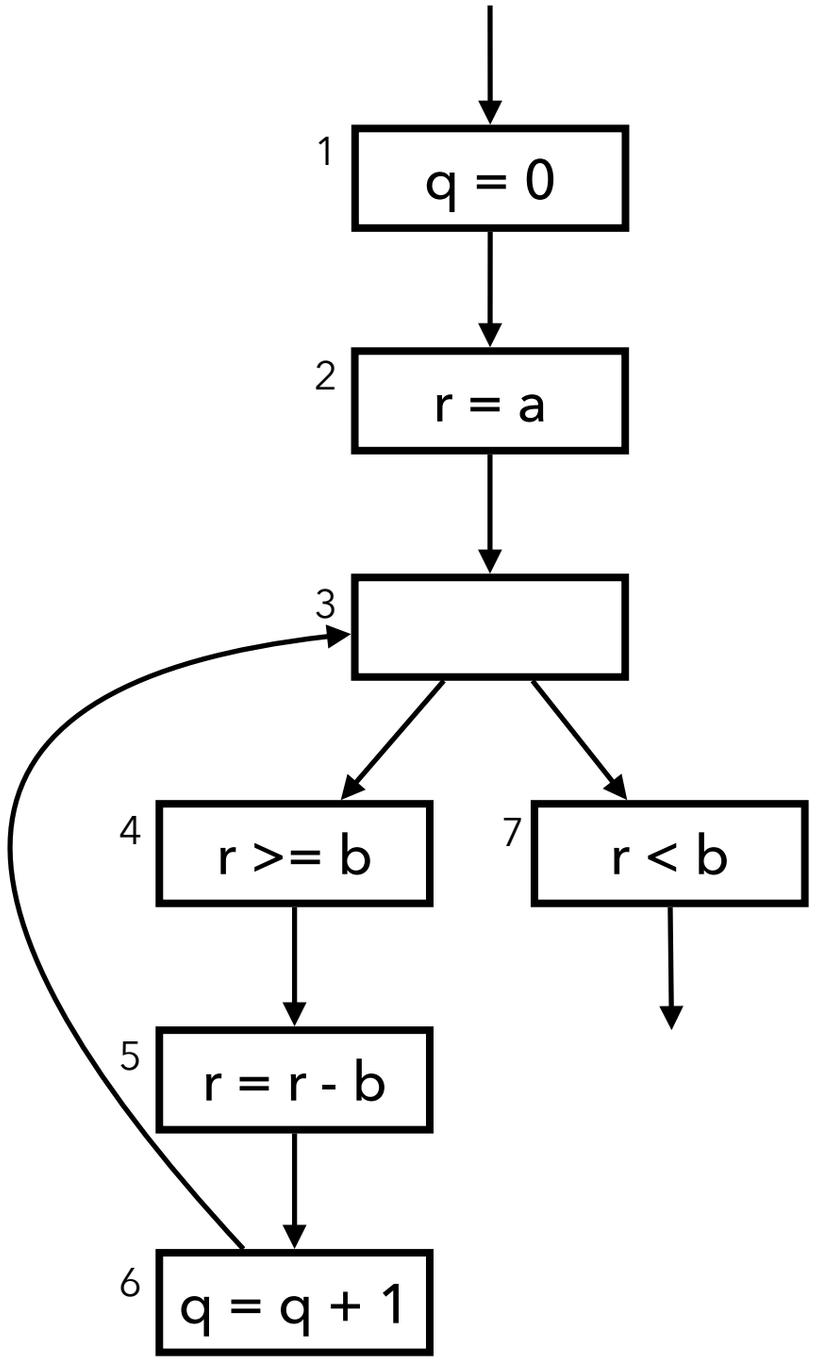
- Abstract operators

+/−	top	neg	zero	pos	bot
top					
neg					
zero					
pos					
bot					

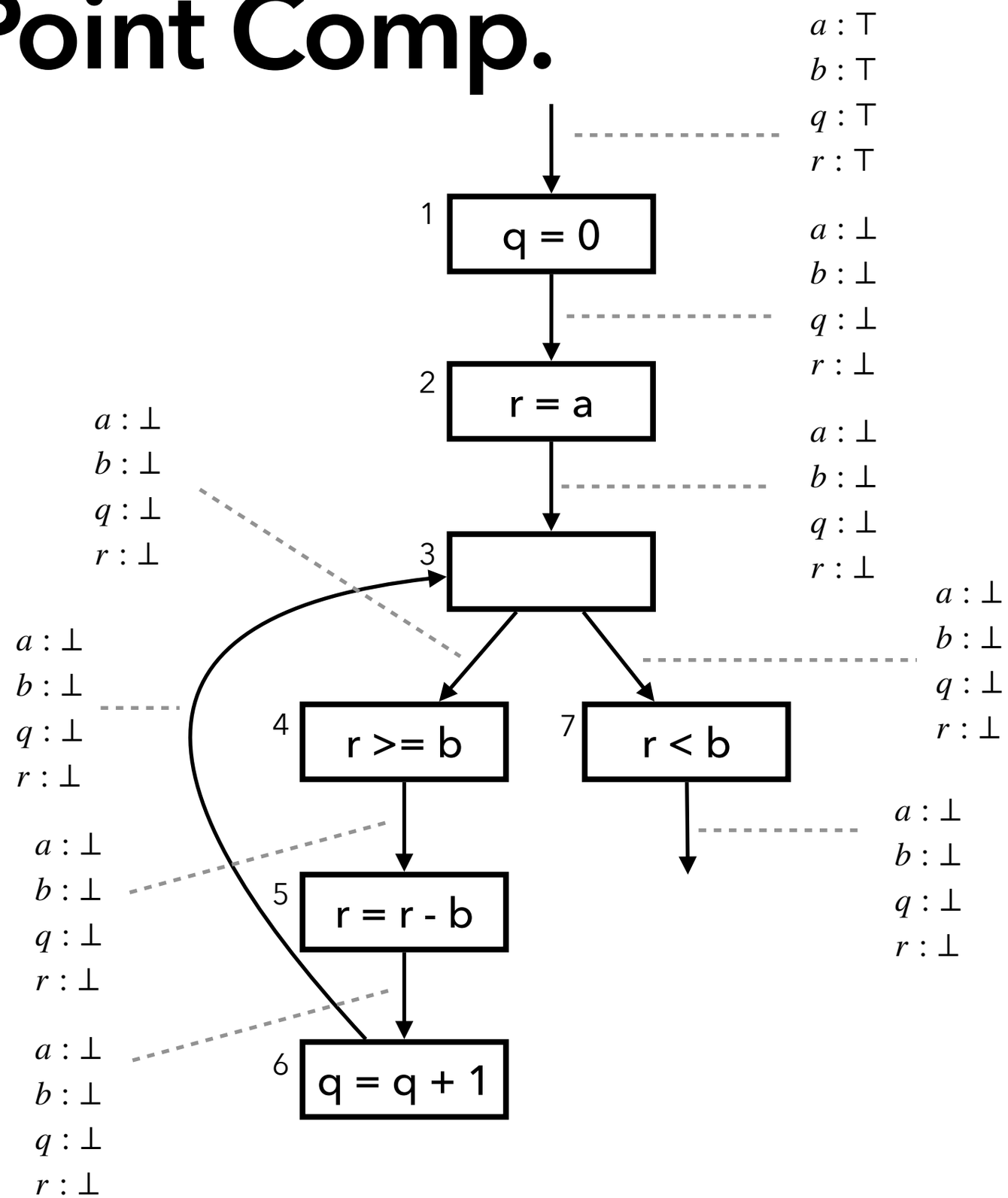
×	top	neg	zero	pos	bot
top					
neg					
zero					
pos					
bot					

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);
```

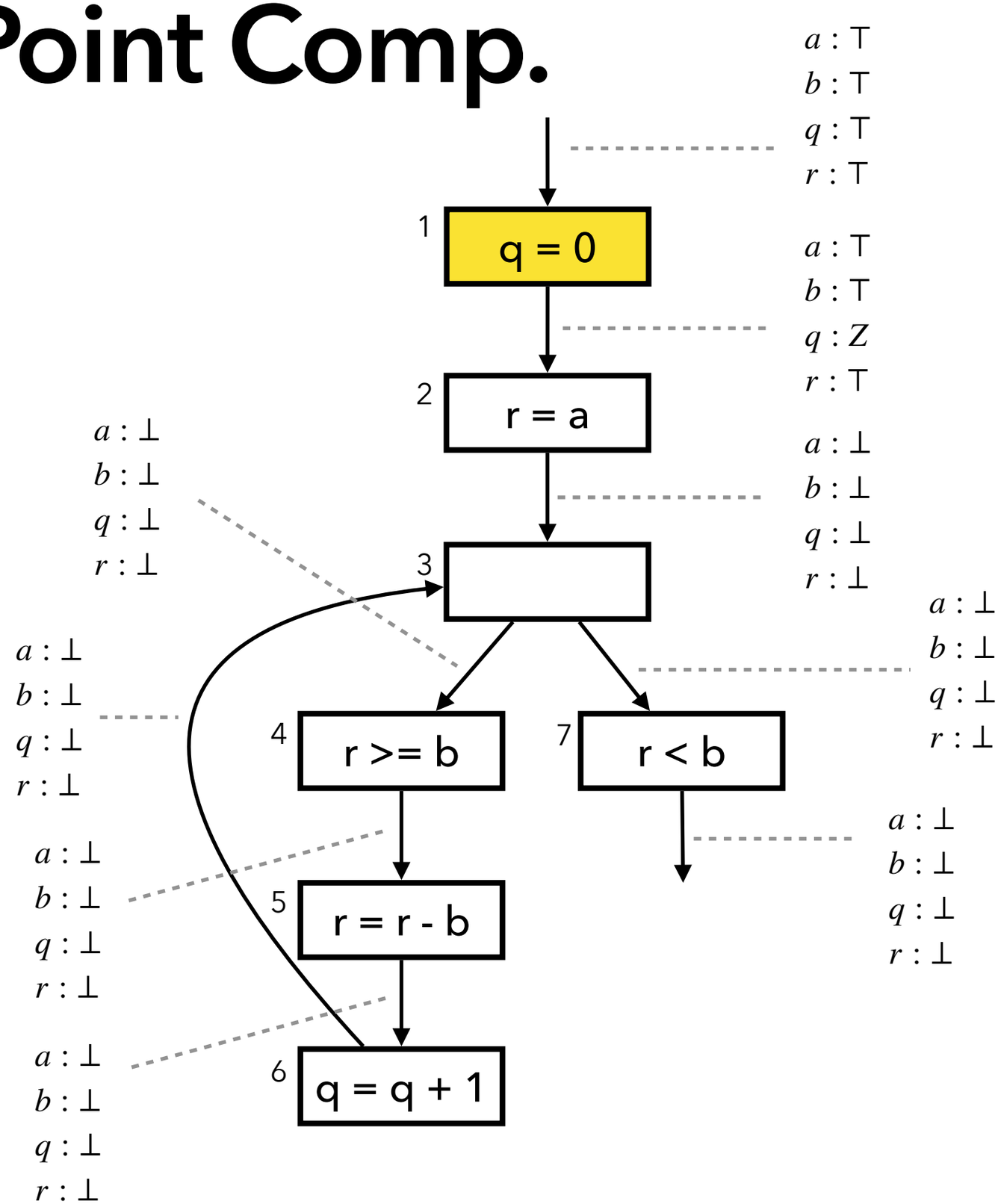


Fixed Point Comp.



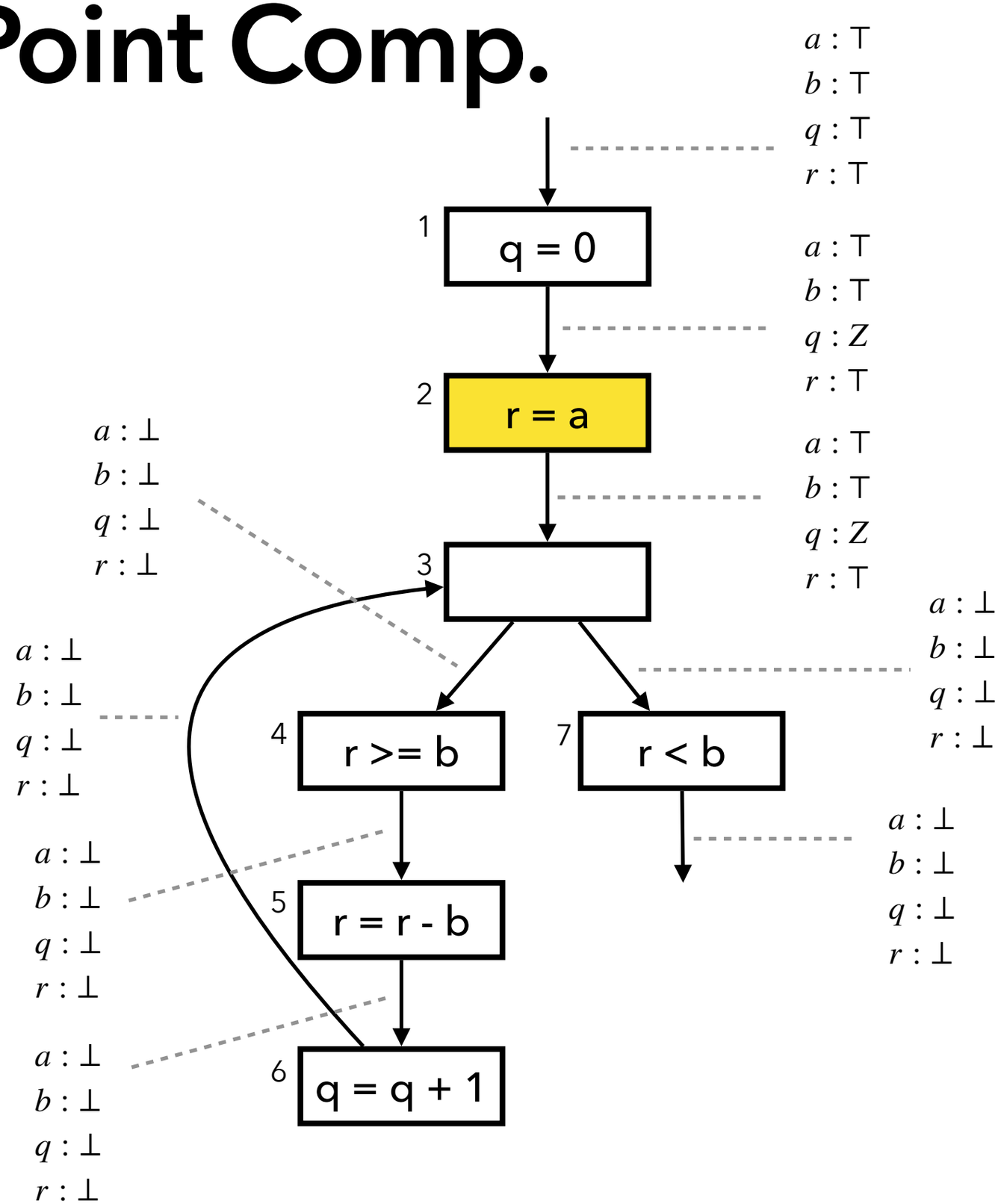
W = { 1, 2, 3, 4, 5, 6, 7 }

Fixed Point Comp.



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

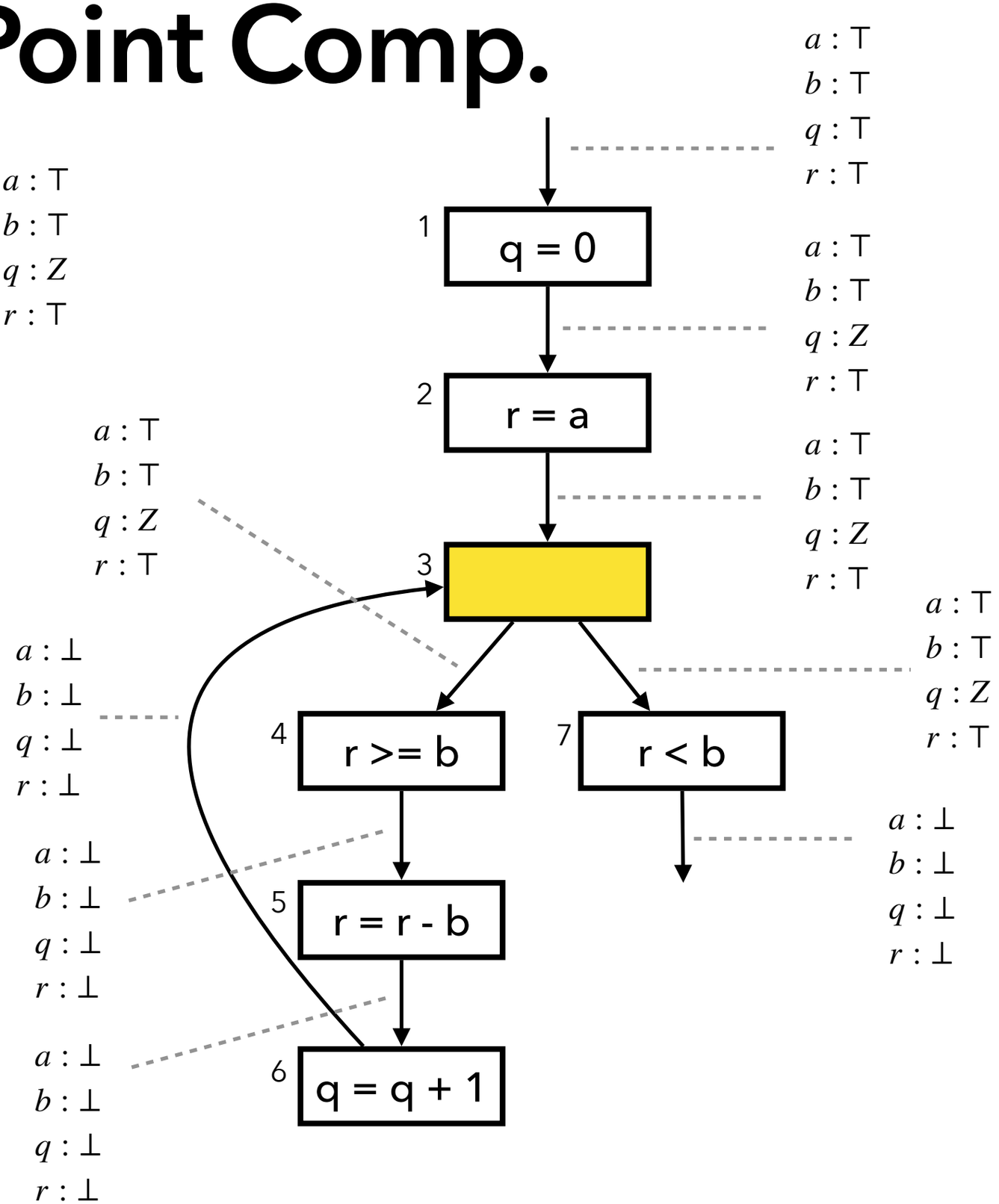
Fixed Point Comp.



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

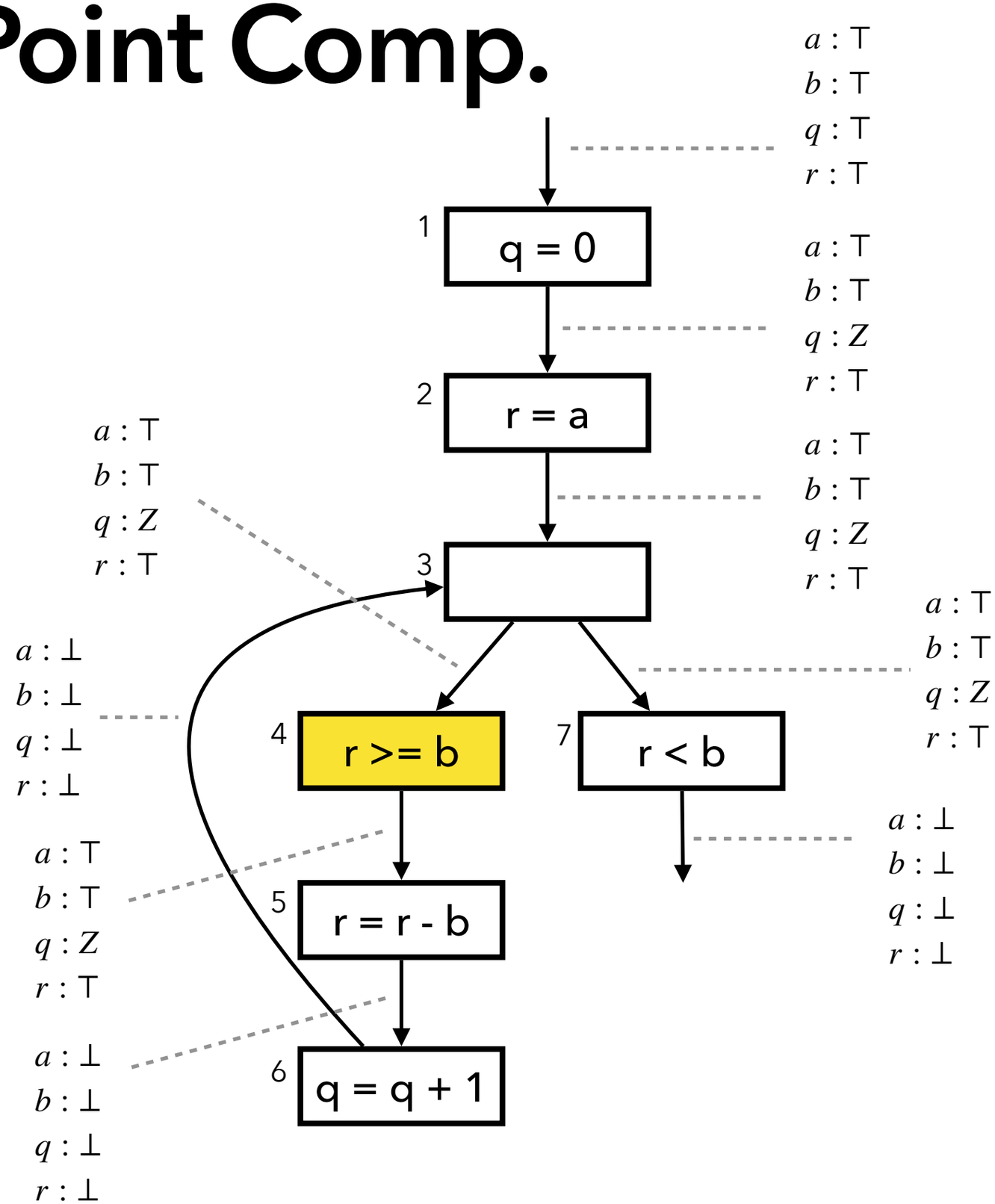
Fixed Point Comp.

$$\begin{matrix} a:\top \\ b:\top \\ q:Z \\ r:\top \end{matrix} \sqcup \begin{matrix} a:\bot \\ b:\bot \\ q:\bot \\ r:\bot \end{matrix} = \begin{matrix} a:\top \\ b:\top \\ q:Z \\ r:\top \end{matrix}$$



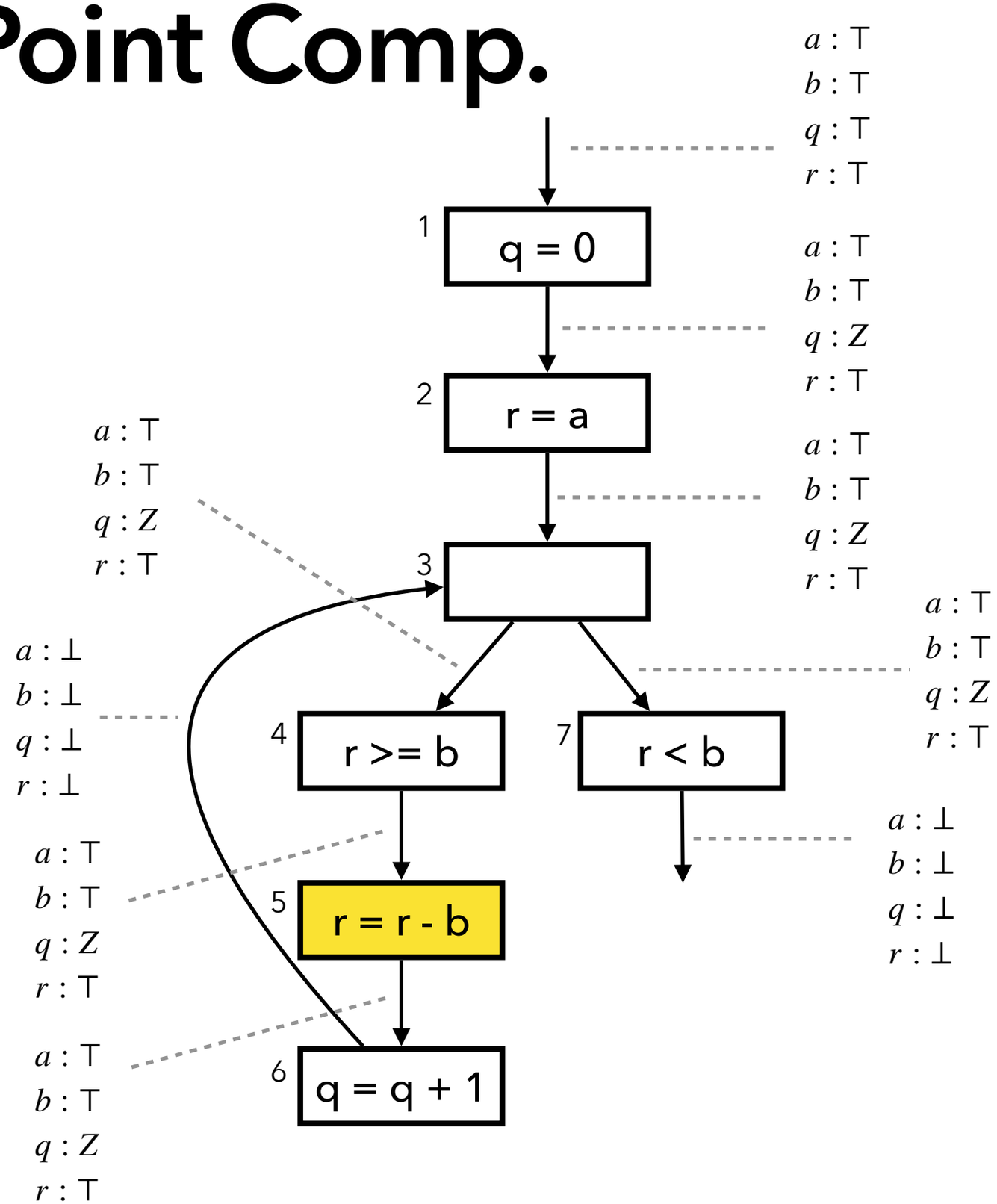
W = { ~~1~~, ~~2~~, ~~3~~, 4, 5, 6, 7 }

Fixed Point Comp.



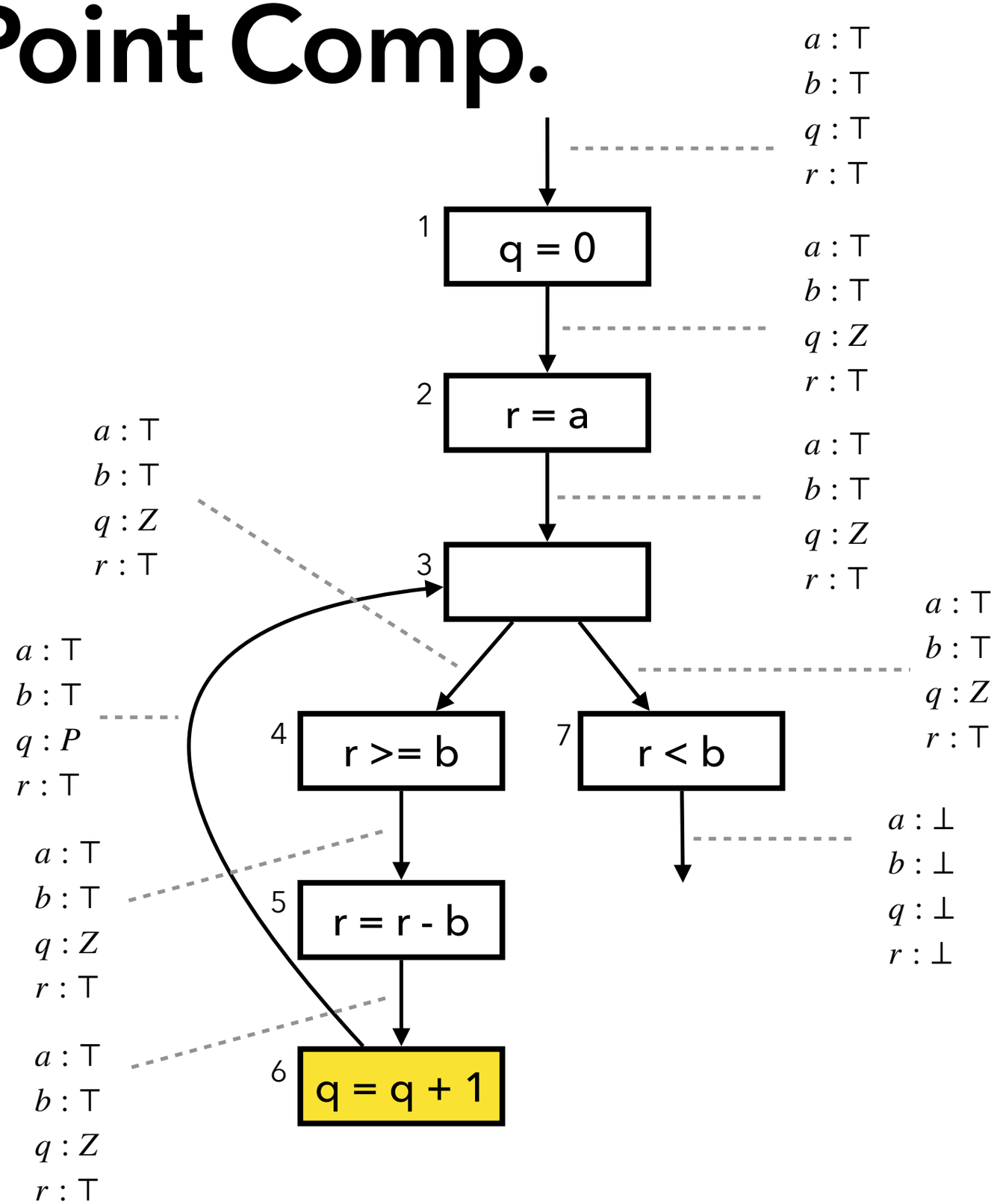
W = { ~~1~~, ~~2~~, ~~3~~, ~~4~~, 5, 6, 7 }

Fixed Point Comp.



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

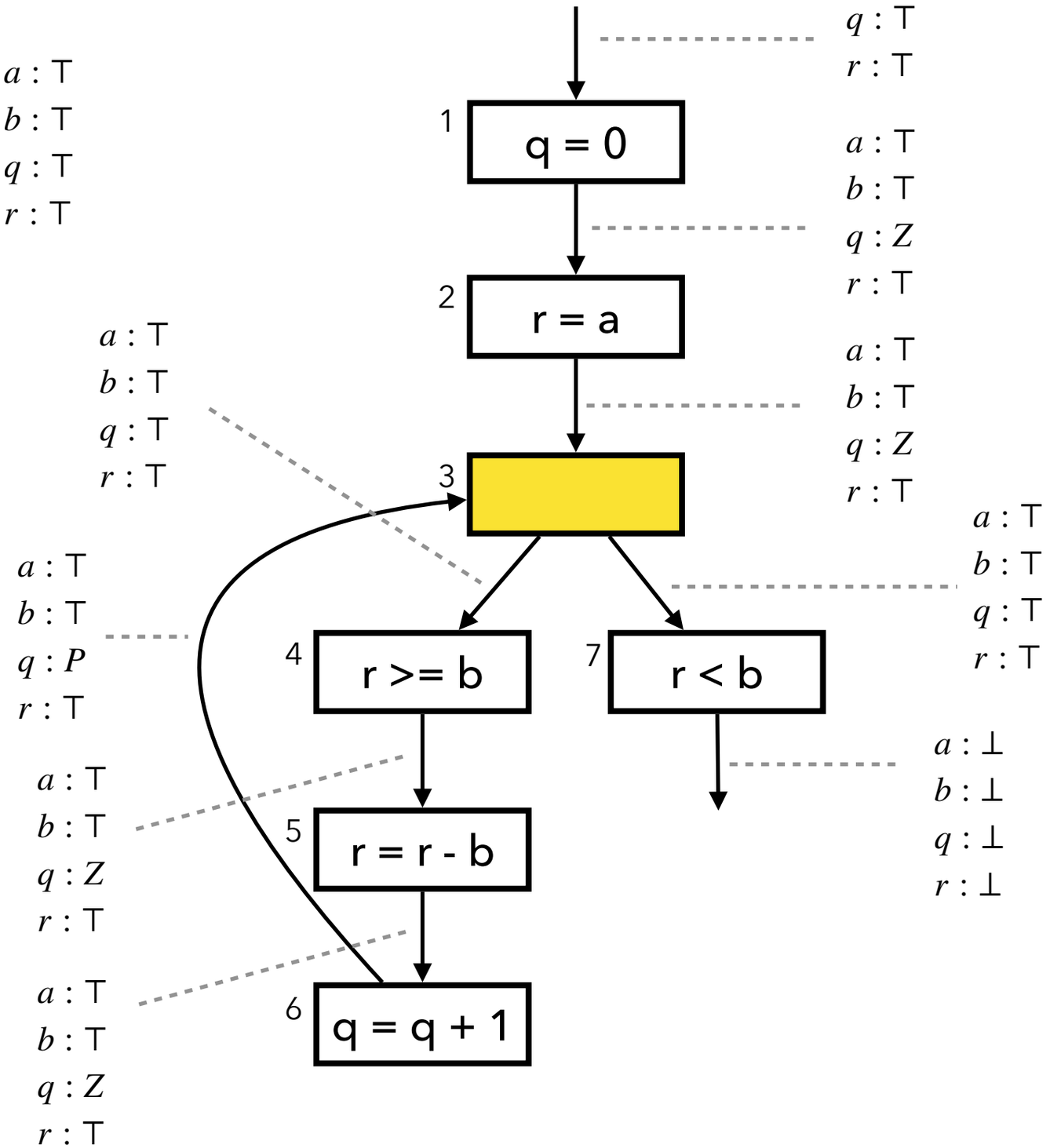
Fixed Point Comp.



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

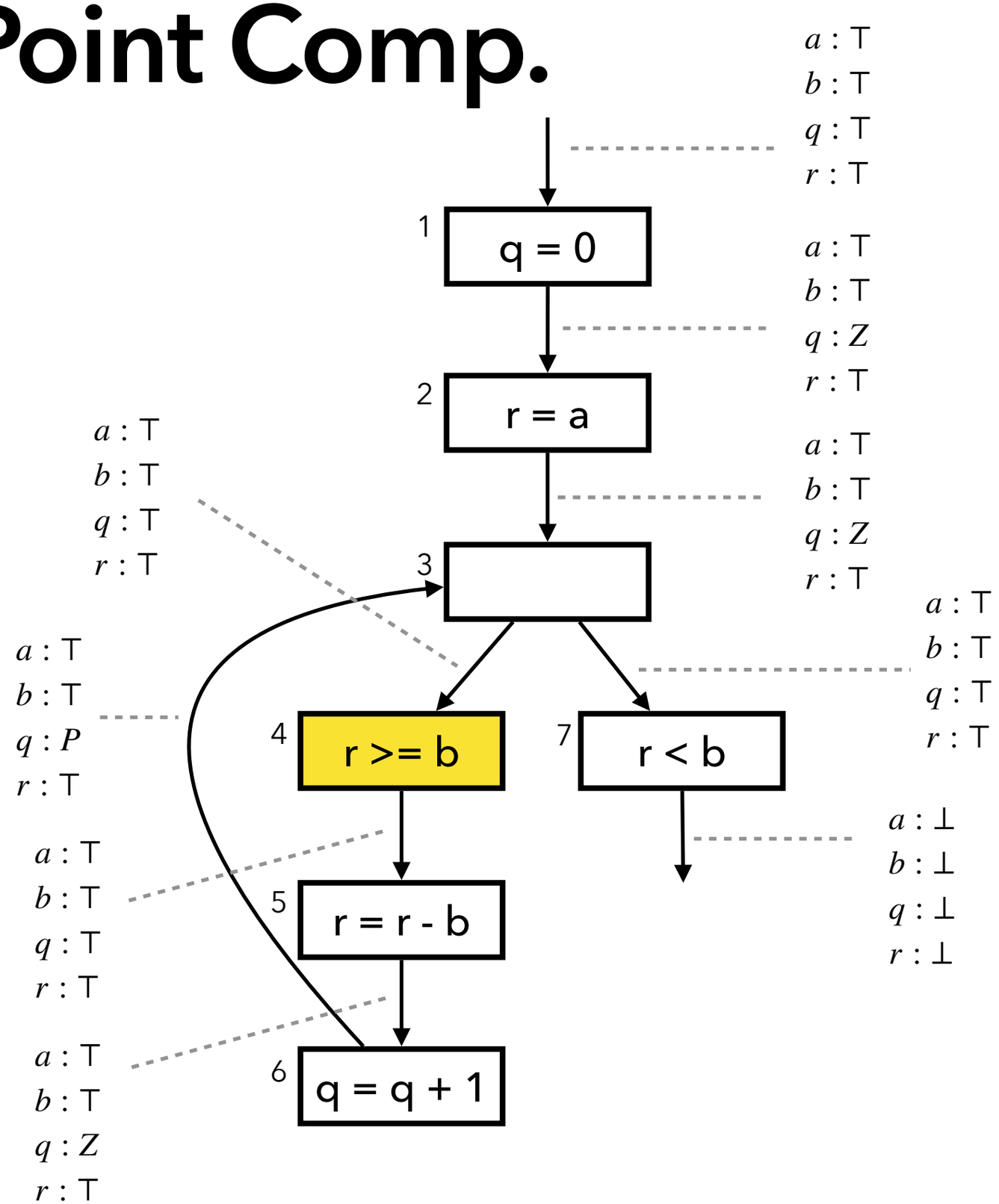
Fixed Point Comp.

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



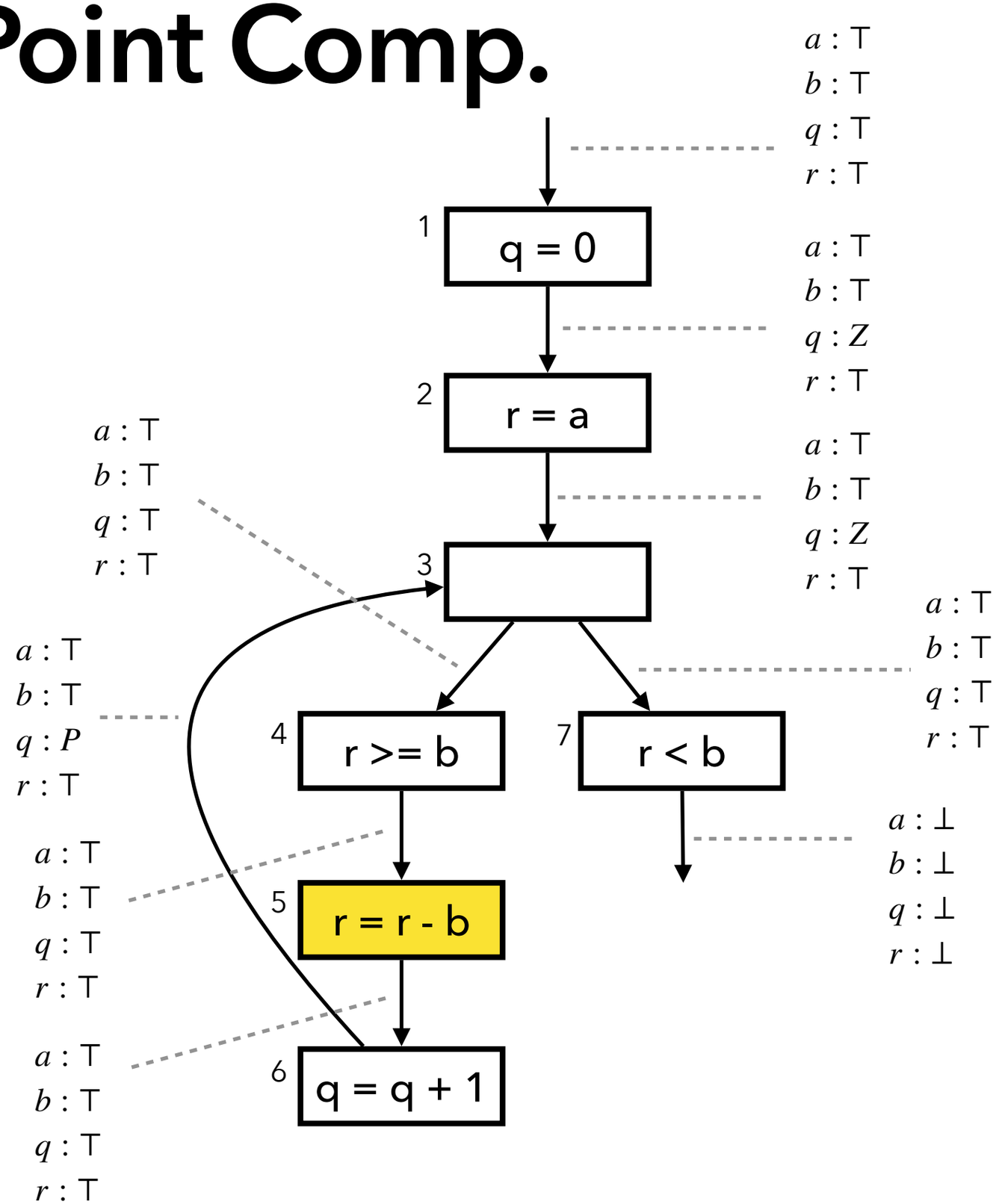
W = { ~~1~~, ~~2~~, ~~3~~, 4, ~~5~~, ~~6~~, 7 }

Fixed Point Comp.



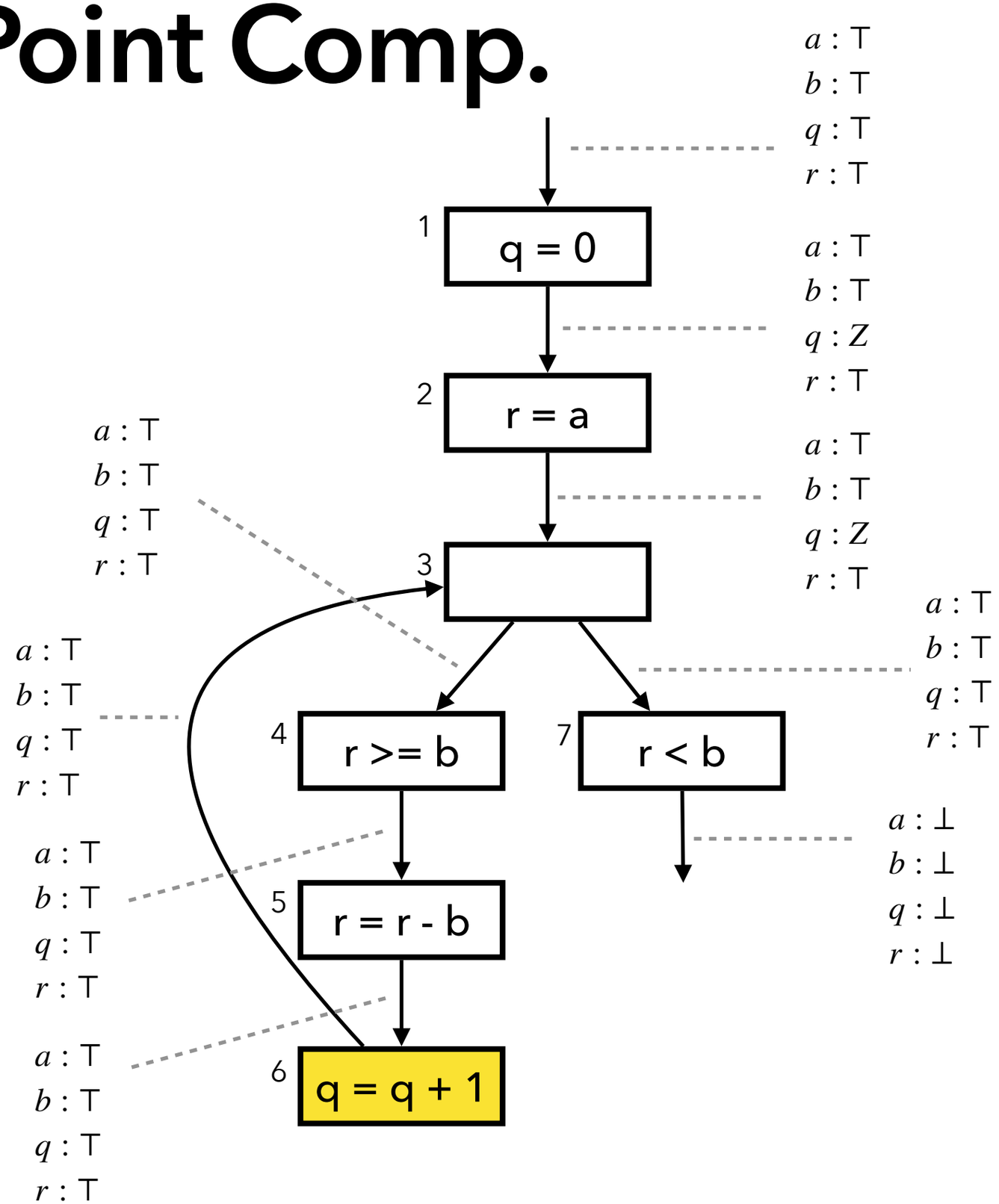
W = { ~~1~~, ~~2~~, ~~3~~, ~~4~~, 5, ~~6~~, 7 }

Fixed Point Comp.



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

Fixed Point Comp.

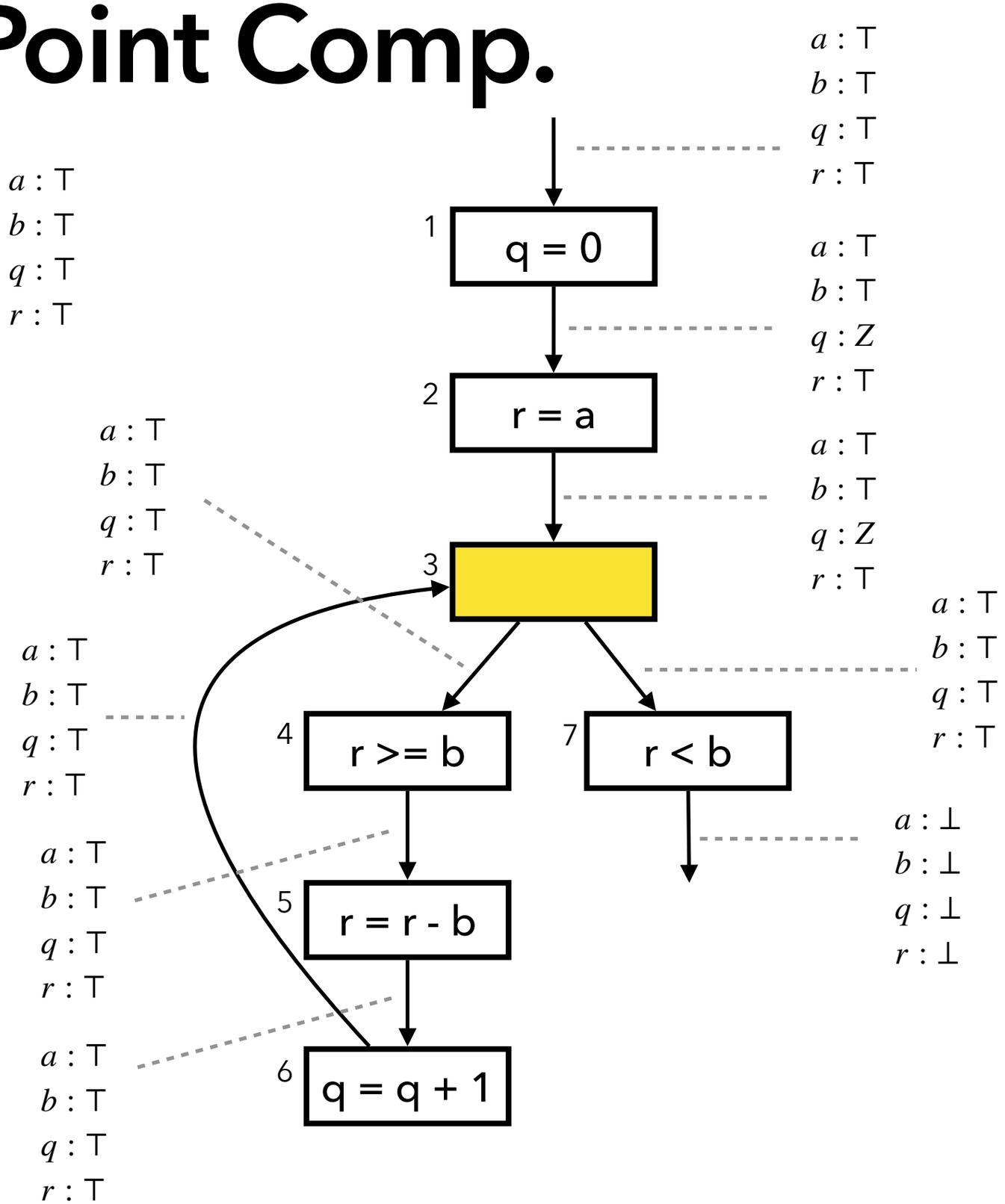


W = { ~~1~~, ~~2~~, 3, ~~4~~, ~~5~~, ~~6~~, 7 }

Fixed Point Comp.

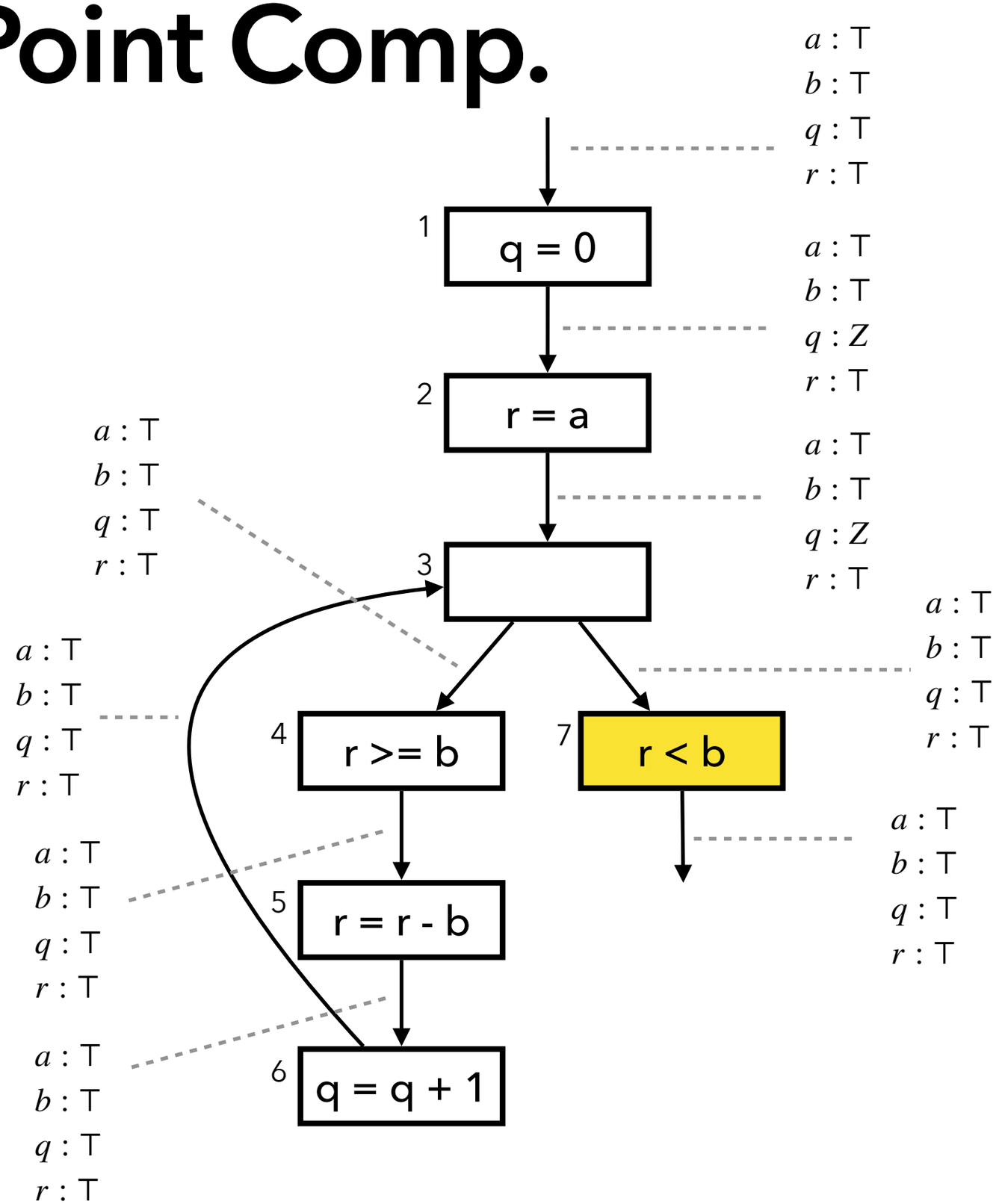
$$\begin{matrix} a:\top \\ b:\top \\ q:Z \\ r:\top \end{matrix} \sqcup \begin{matrix} a:\top \\ b:\top \\ q:\top \\ r:\top \end{matrix} = \begin{matrix} a:\top \\ b:\top \\ q:\top \\ r:\top \end{matrix}$$

(fixed point)



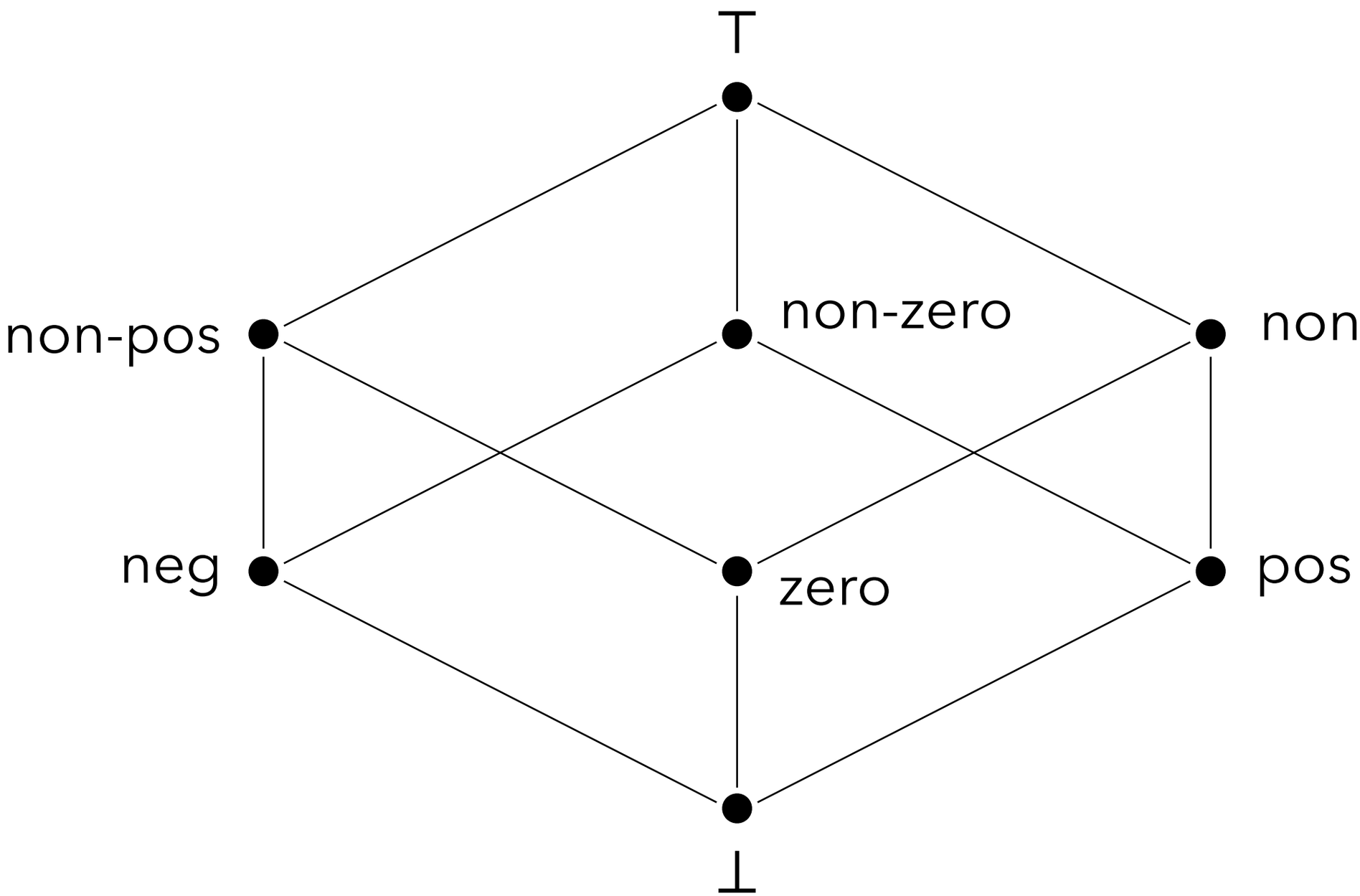
W = { ~~1~~, ~~2~~, ~~3~~, ~~4~~, ~~5~~, ~~6~~, 7 }

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);
```