AAA616: Program Analysis

Lecture 4 — Static Analysis Examples (1)

Hakjoo Oh

2026 Fall

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" (type system)
  - "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" (type system)
  - "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" (type system)
  - $\boxed{\text{"odd integer"}}$ 1. Choose abstract value (domain)
  - "positive integer"
  - "integer between 400 and 500"
  - …

2. "Execute" the program with abstract values

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

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

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

$\top$ (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

$\top$ (don't know)

Even

```
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

Even

⊤ (don't know)

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

Odd

Strength of Static Analysis

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

Even

$\top$ (don't know)

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

Odd

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

$\top$ (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

$\top$ (don't know)

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

$\top$ (don't know)

Weakness of Static Analysis

- Instead, static analysis may produce false alarms

$\top$ (don't know)

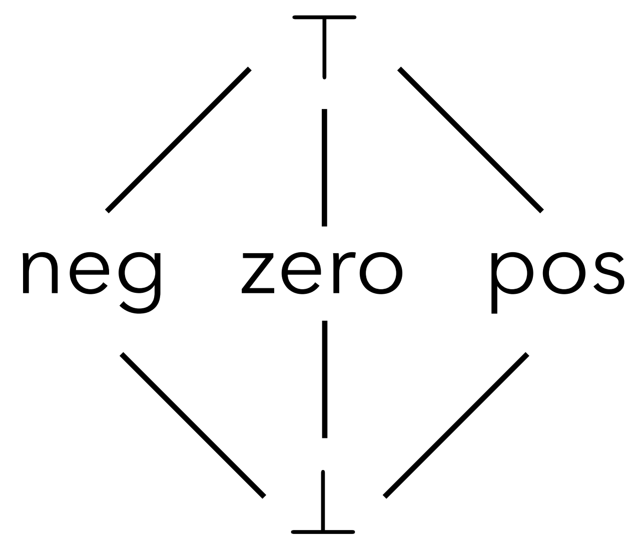
$\top$ (don't know)

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

false alarm

A Simple Sign Domain

- Abstract values



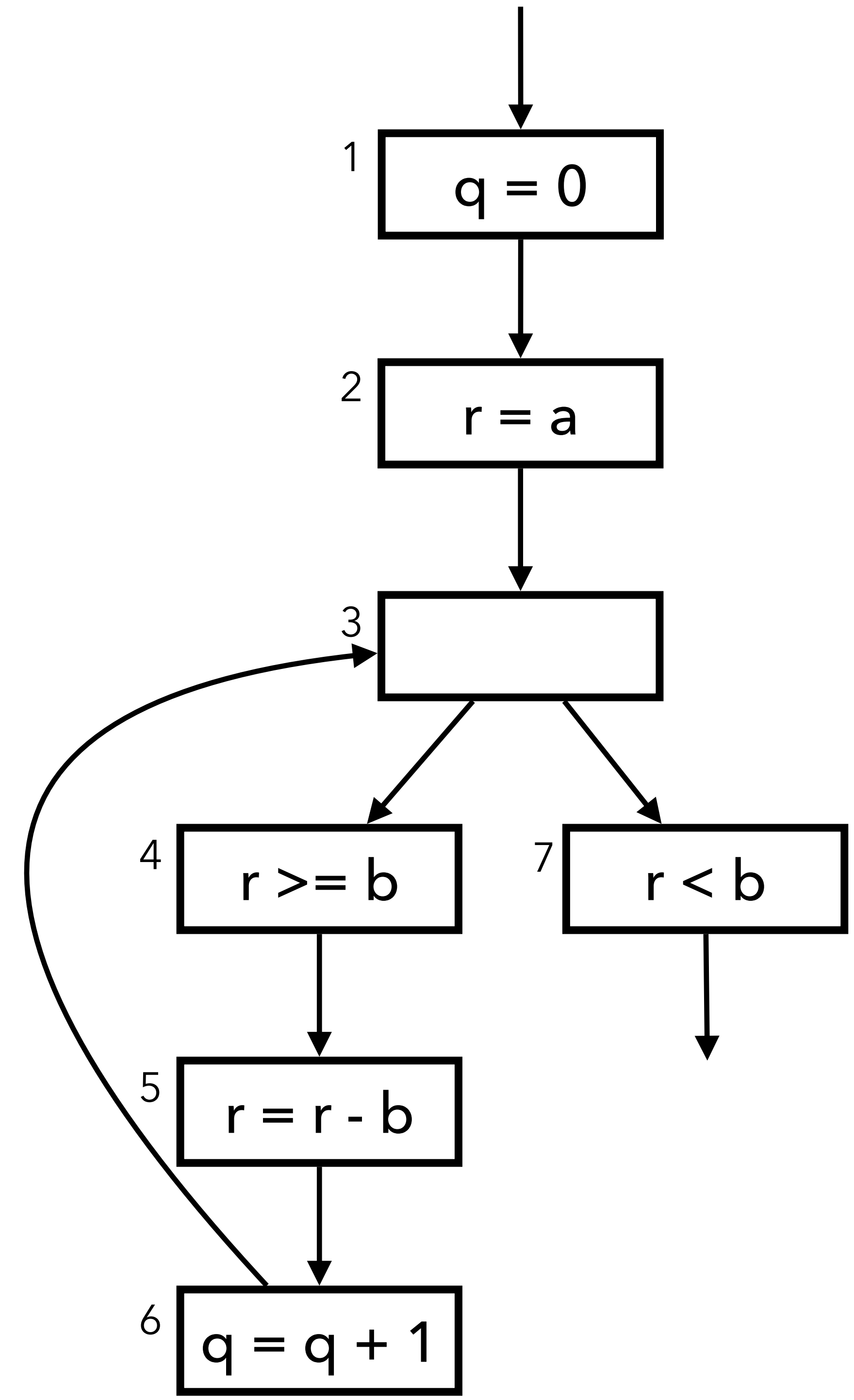
- Abstract operators

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

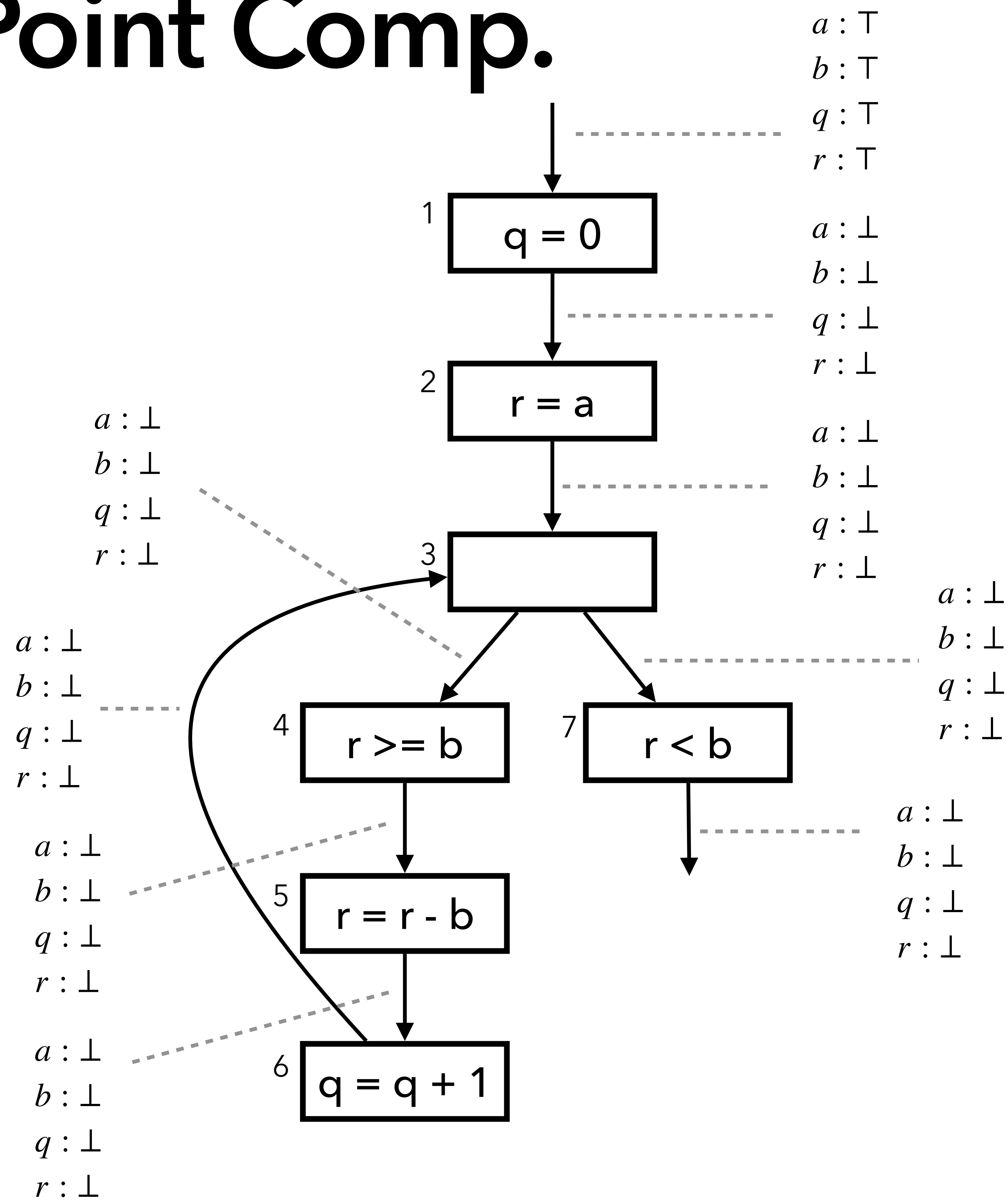
×	top	neg	zero	pos	bot
top					
neg					
zero					
pos					
bot					

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);
```

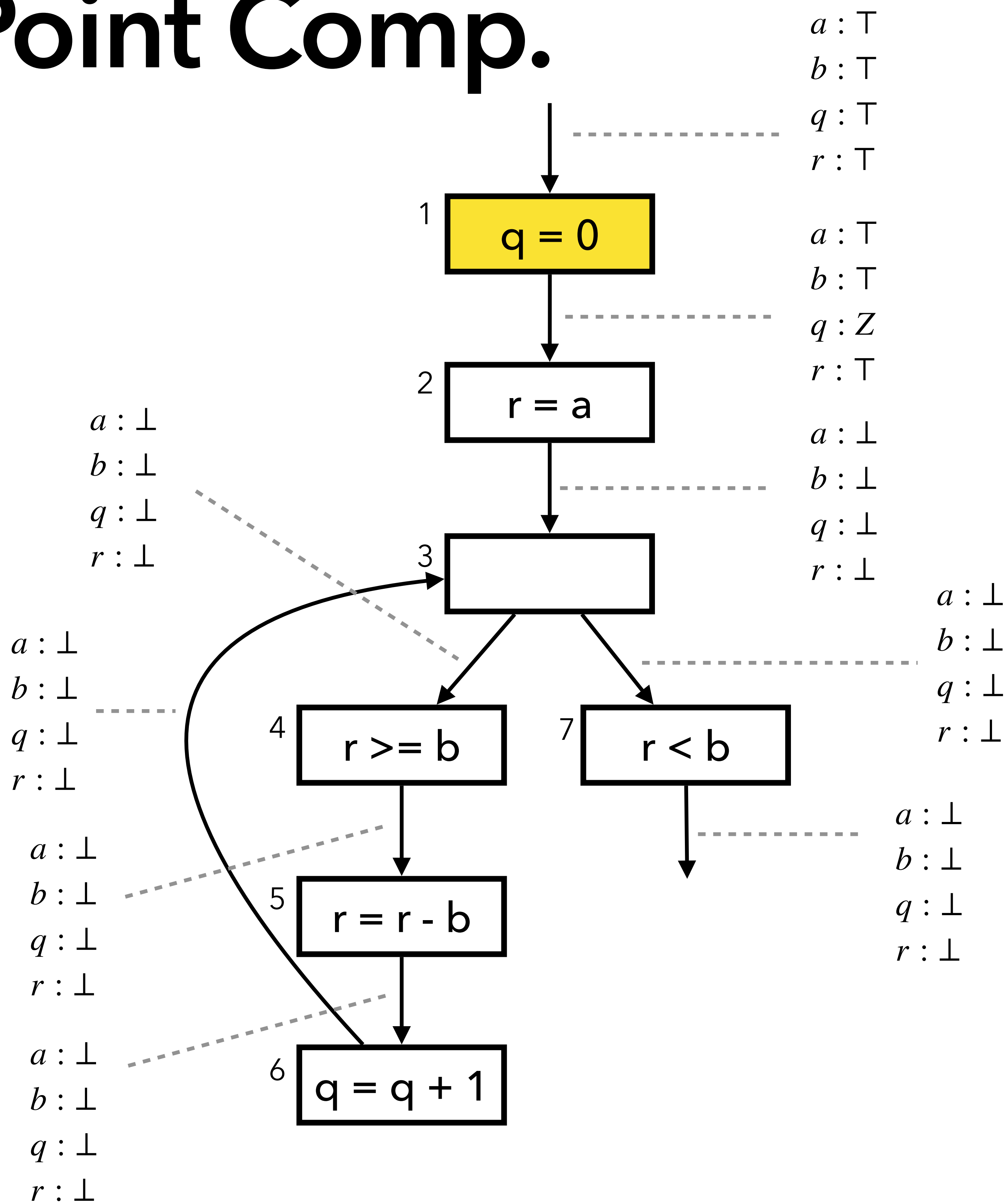


Fixed Point Comp.



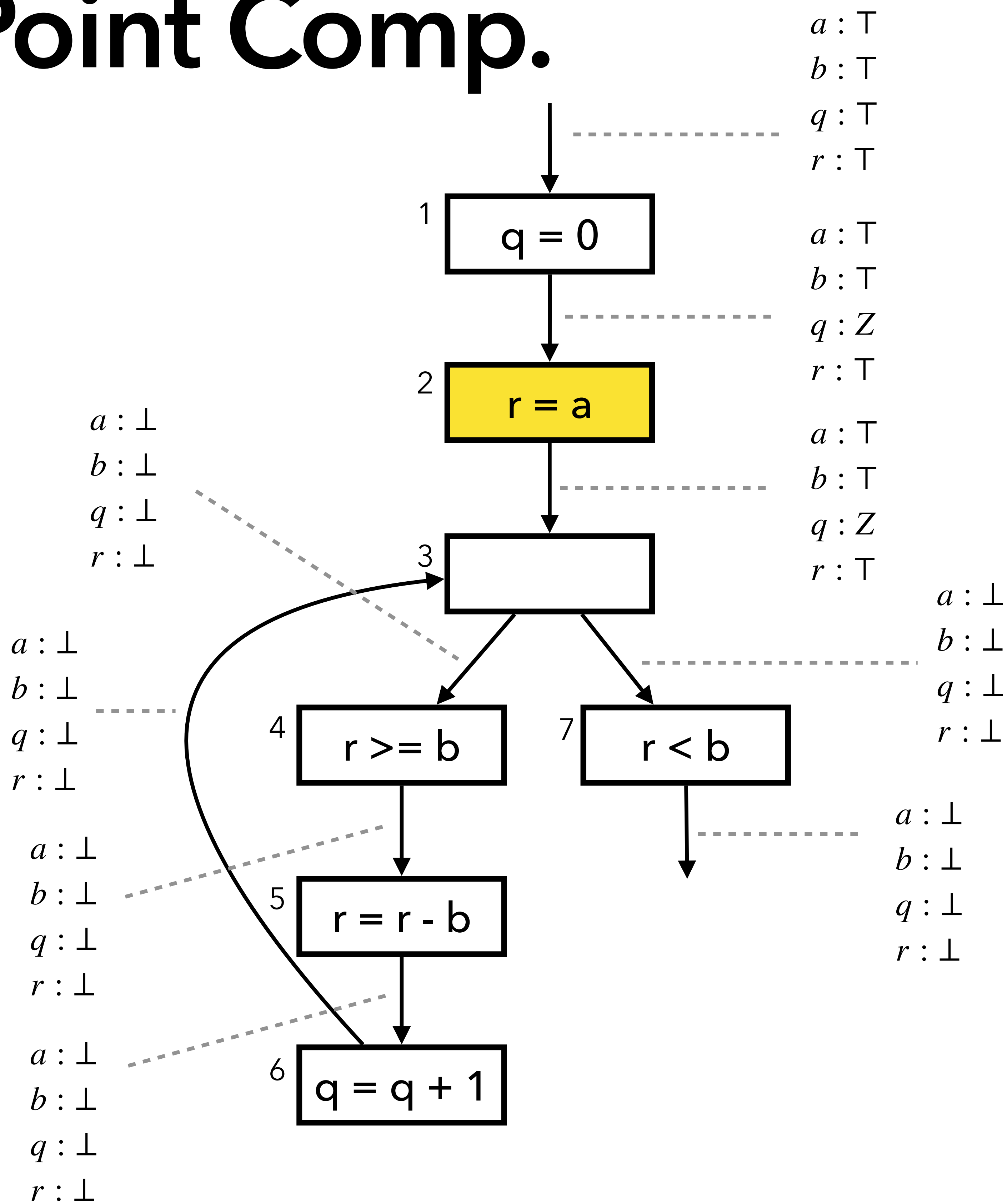
W = { 1, 2, 3, 4, 5, 6, 7 }

Fixed Point Comp.



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

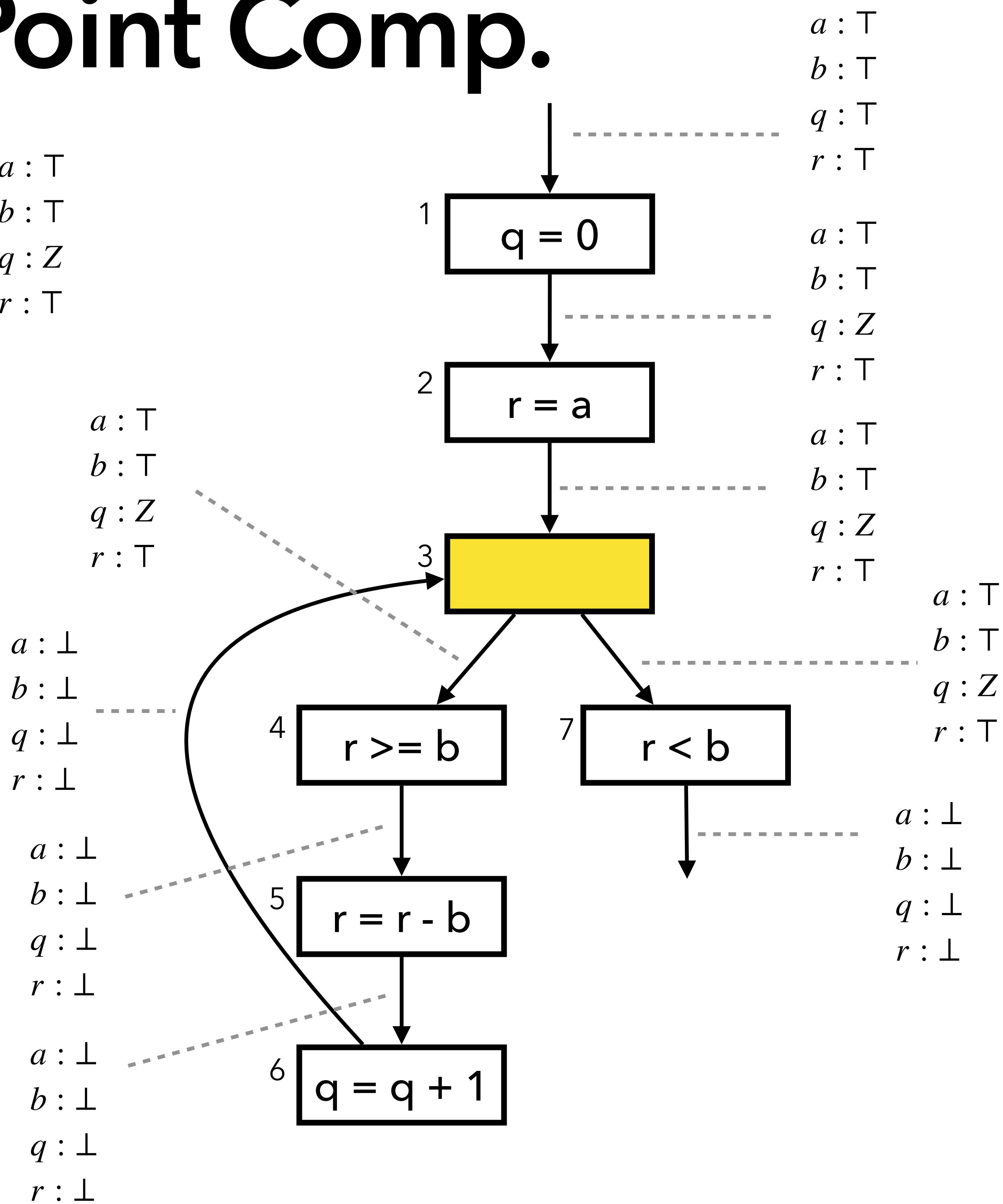
Fixed Point Comp.



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

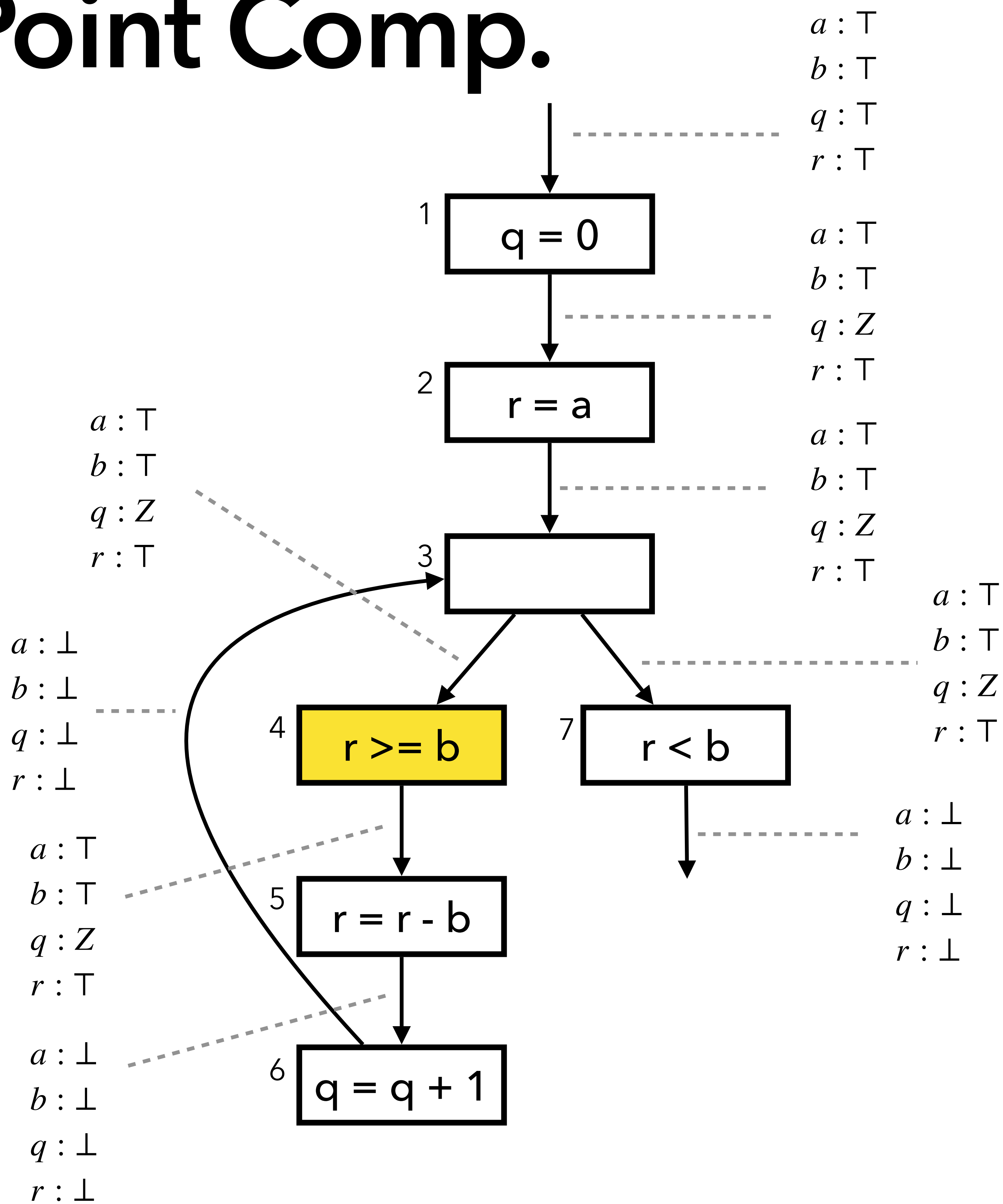
Fixed Point Comp.

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



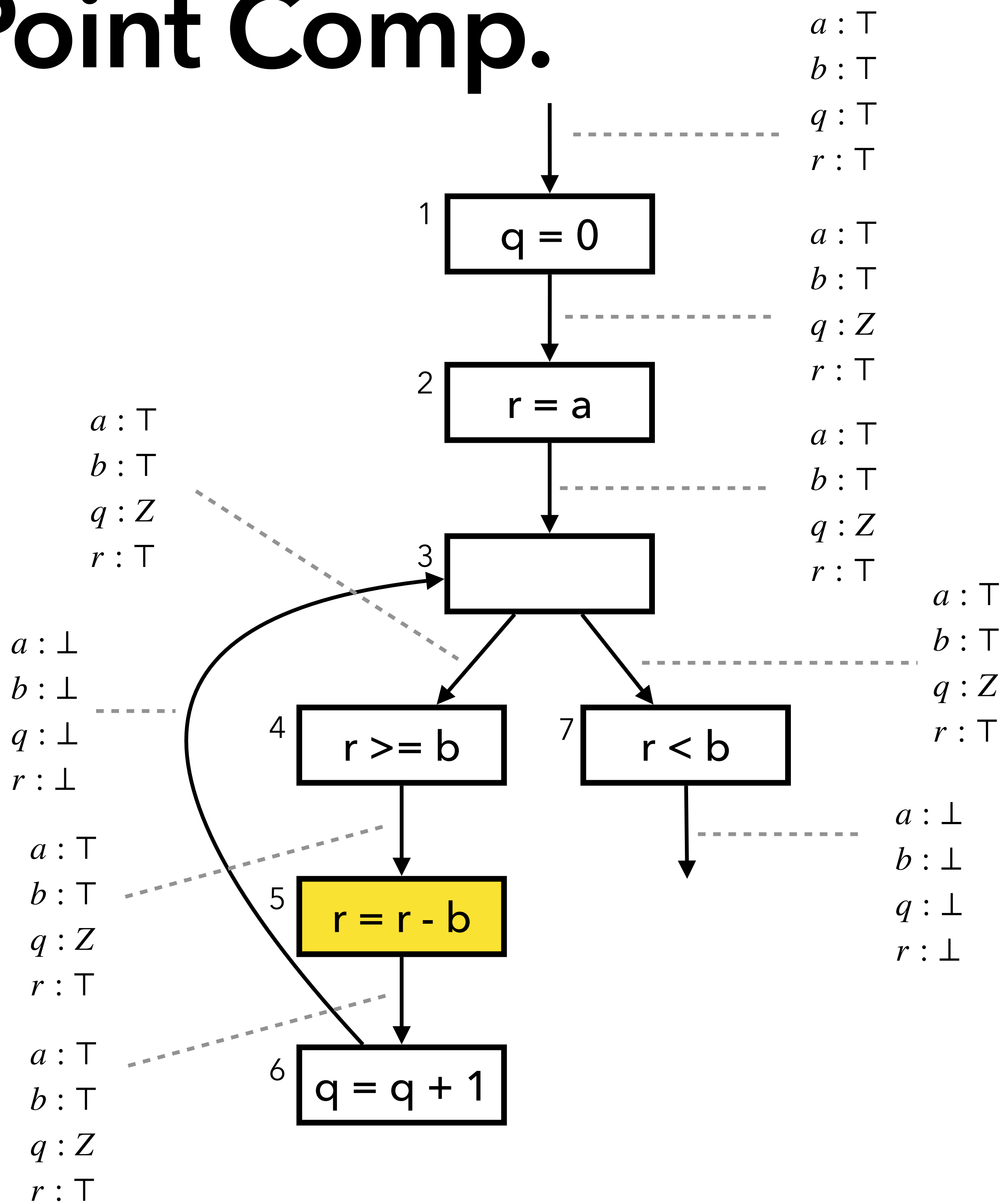
W = { ~~1~~, ~~2~~, ~~3~~, 4, 5, 6, 7 }

Fixed Point Comp.



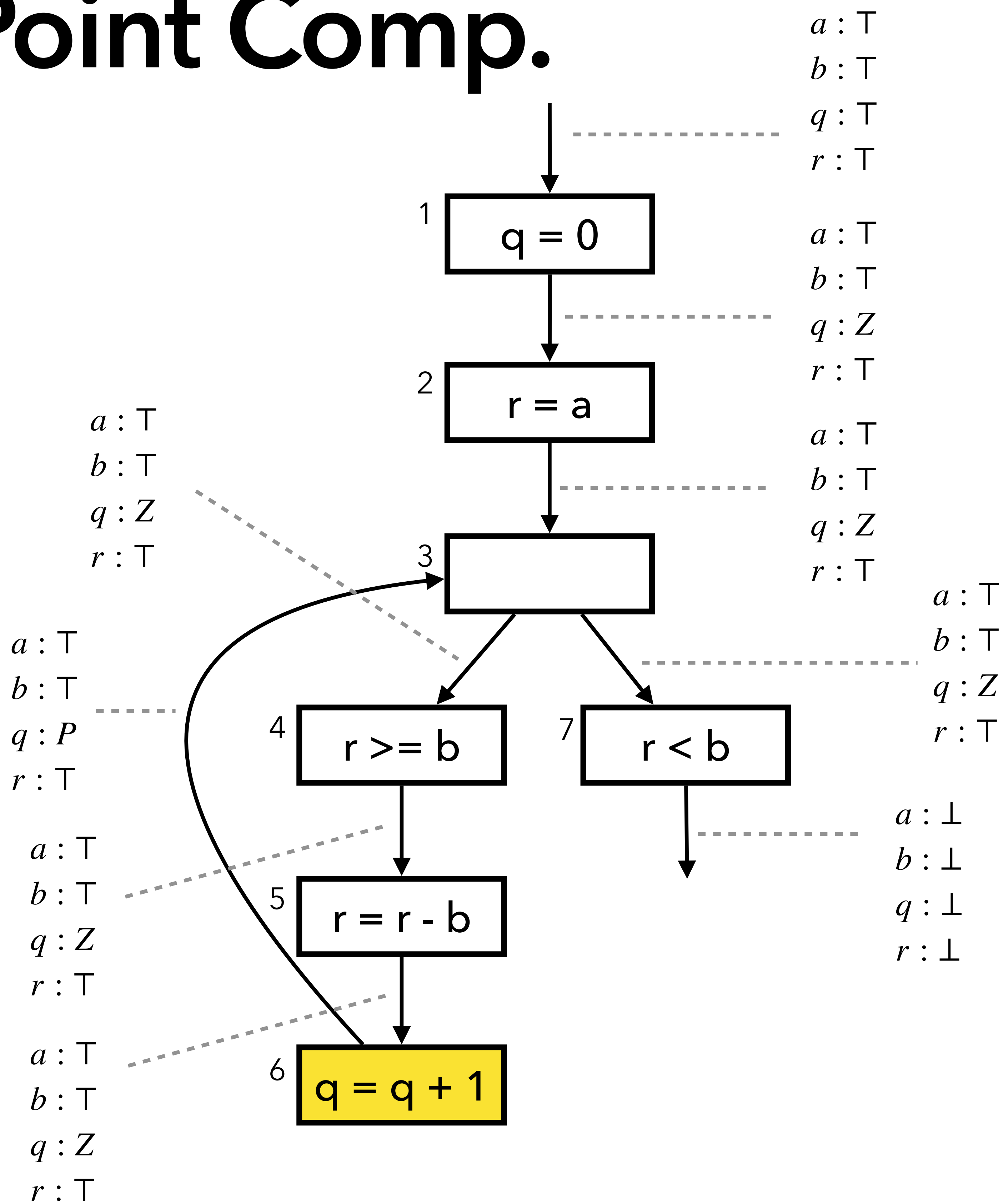
W = { ~~1~~, ~~2~~, ~~3~~, ~~4~~, 5, 6, 7 }

Fixed Point Comp.



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

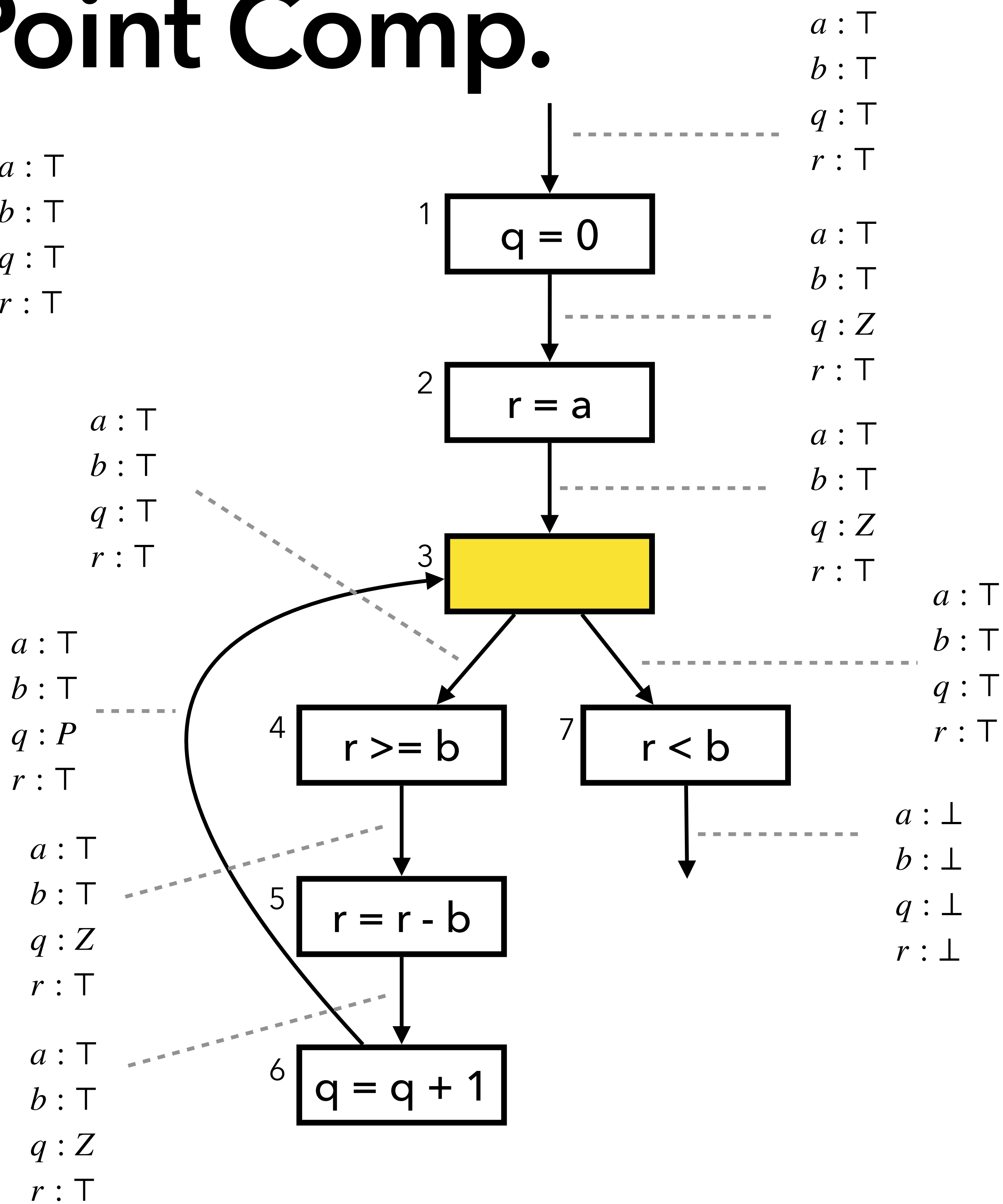
Fixed Point Comp.



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

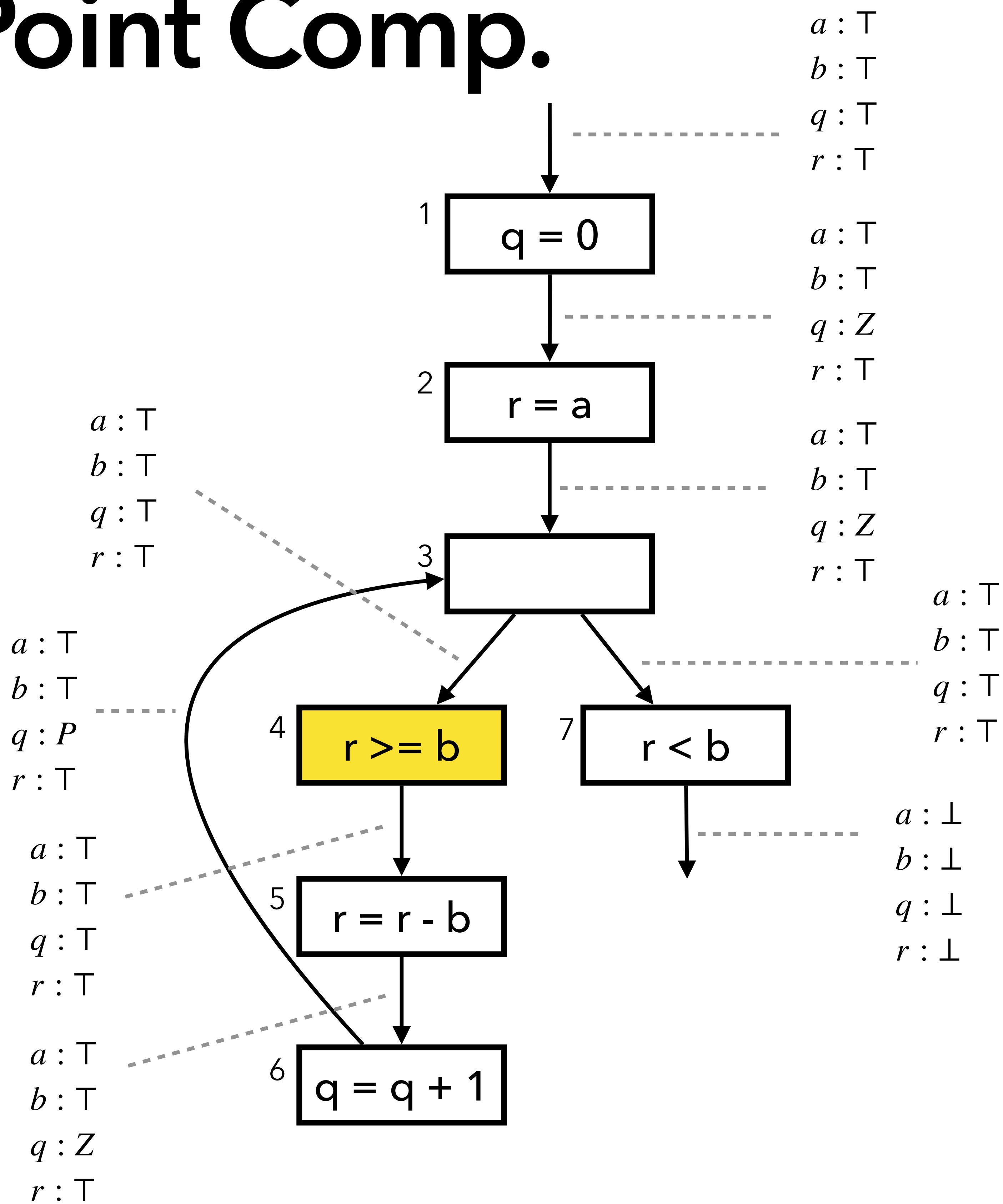
Fixed Point Comp.

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



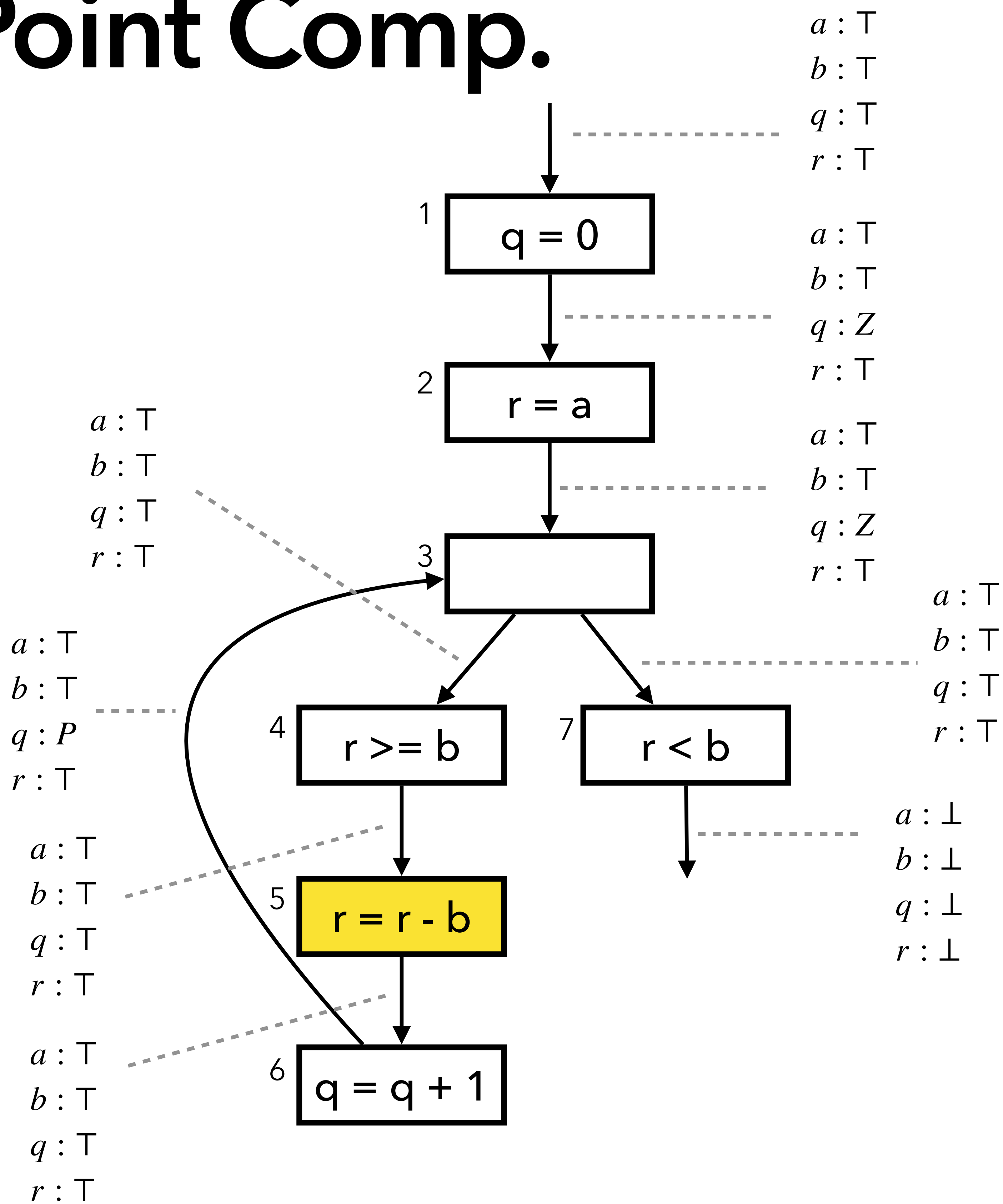
W = { ~~1~~, ~~2~~, ~~3~~, 4, ~~5~~, ~~6~~, 7 }

Fixed Point Comp.



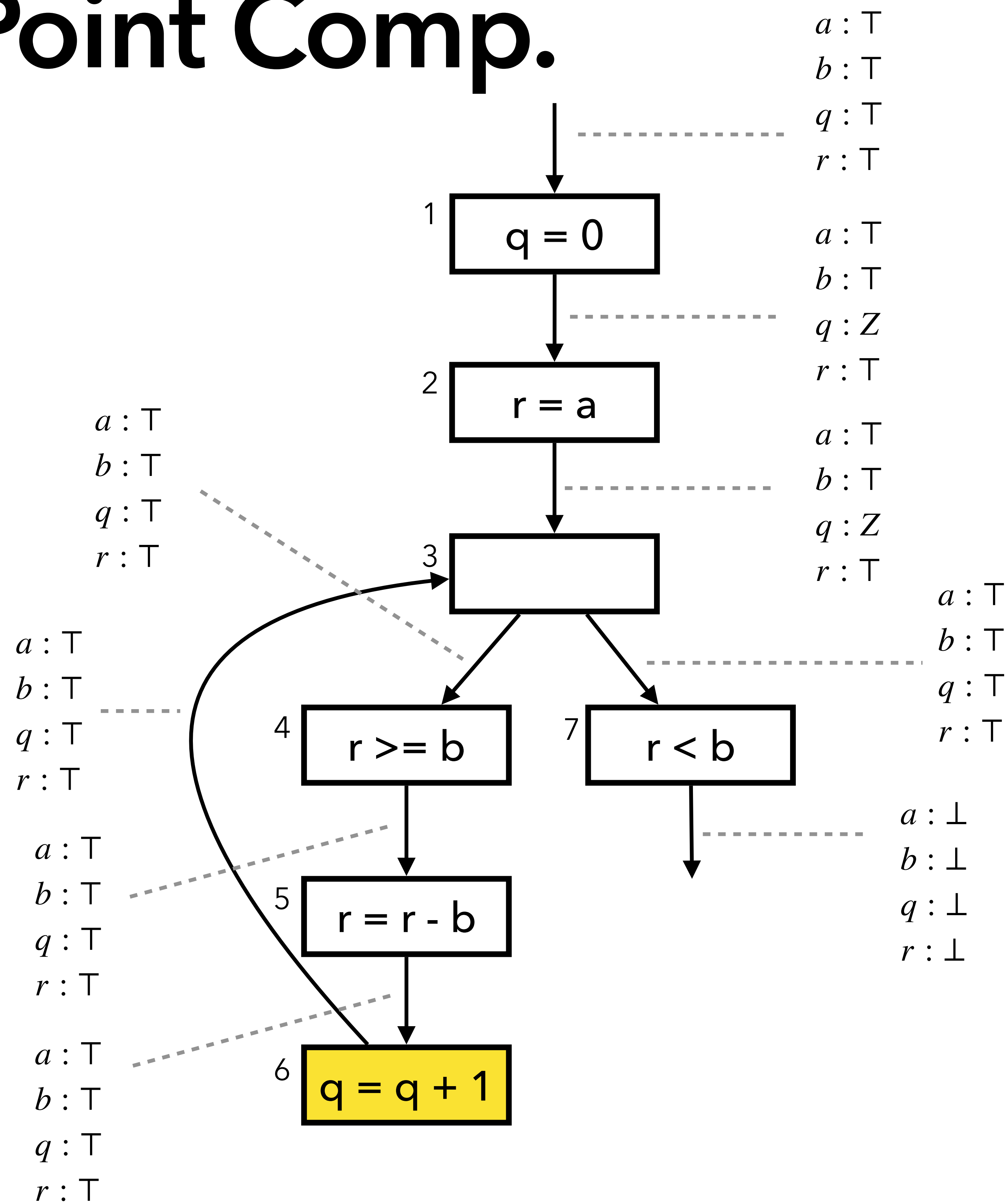
W = { ~~1~~, ~~2~~, ~~3~~, ~~4~~, 5, ~~6~~, 7 }

Fixed Point Comp.



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

Fixed Point Comp.

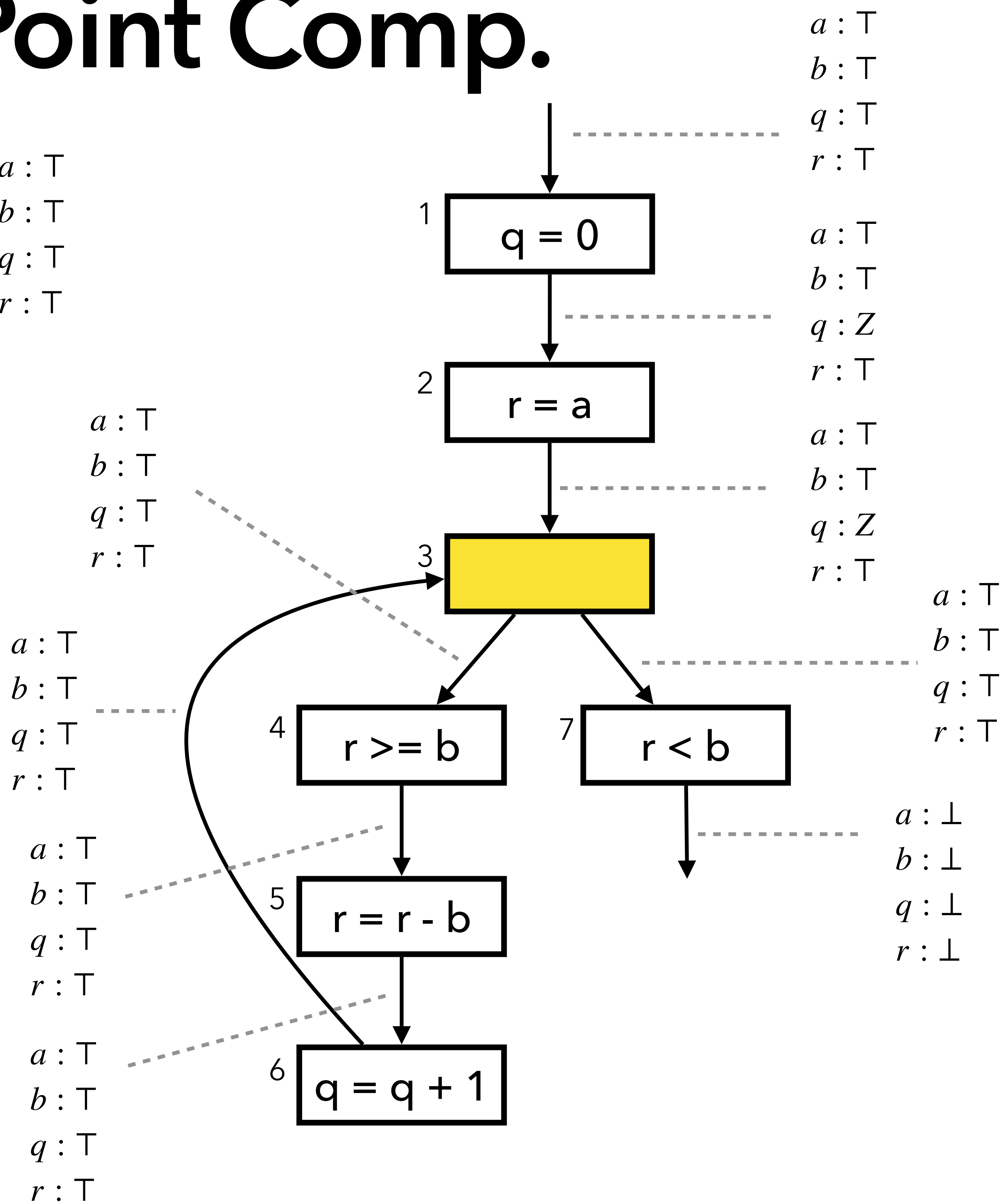


W = { ~~1~~, ~~2~~, 3, 4, ~~5~~, ~~6~~, 7 }

Fixed Point Comp.

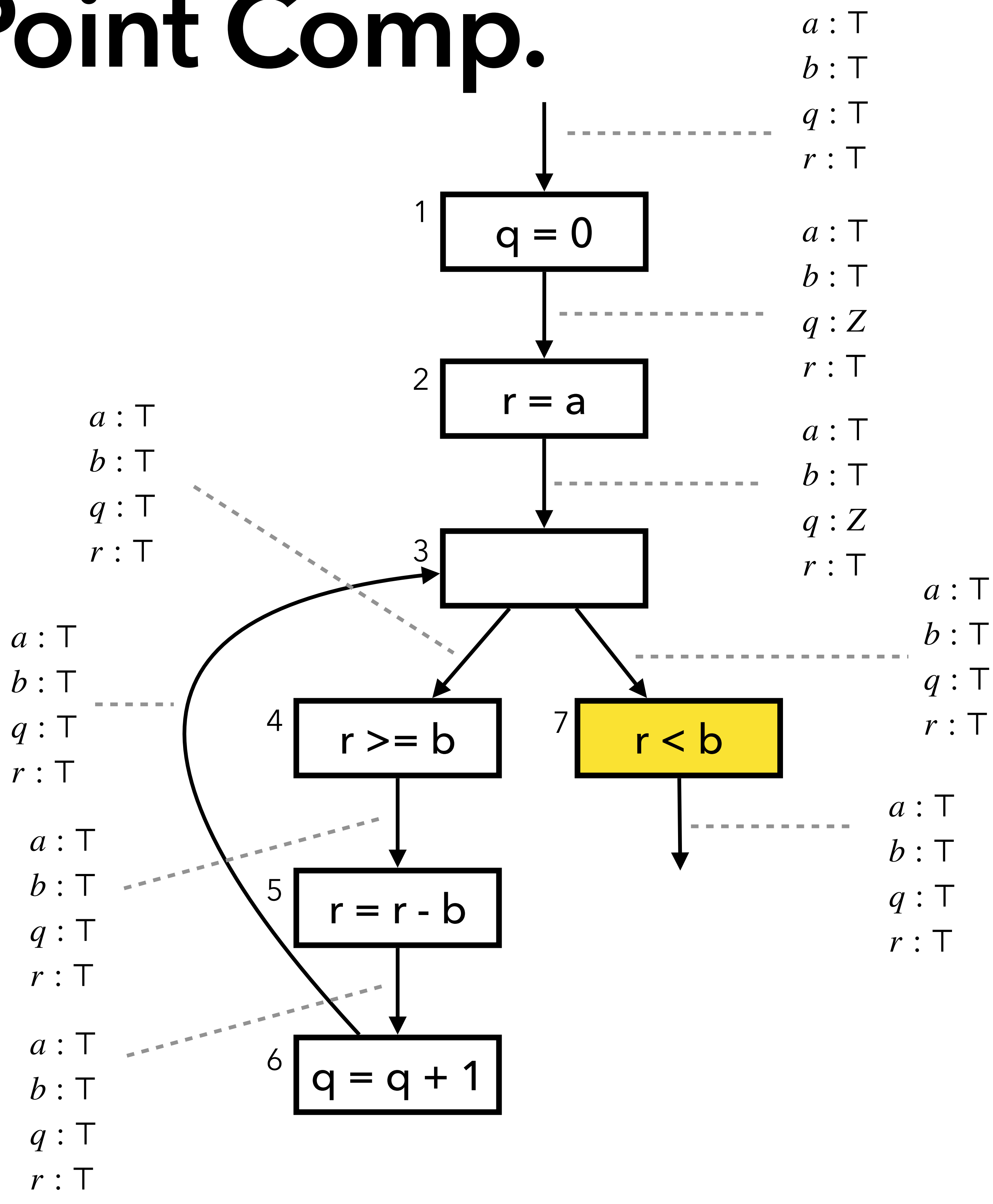
$$\begin{array}{l} a : \top \\ b : \top \\ q : Z \\ r : \top \end{array} \sqcup \begin{array}{l} a : \top \\ b : \top \\ q : \top \\ r : \top \end{array} = \begin{array}{l} a : \top \\ b : \top \\ q : \top \\ r : \top \end{array}$$

(fixed point)



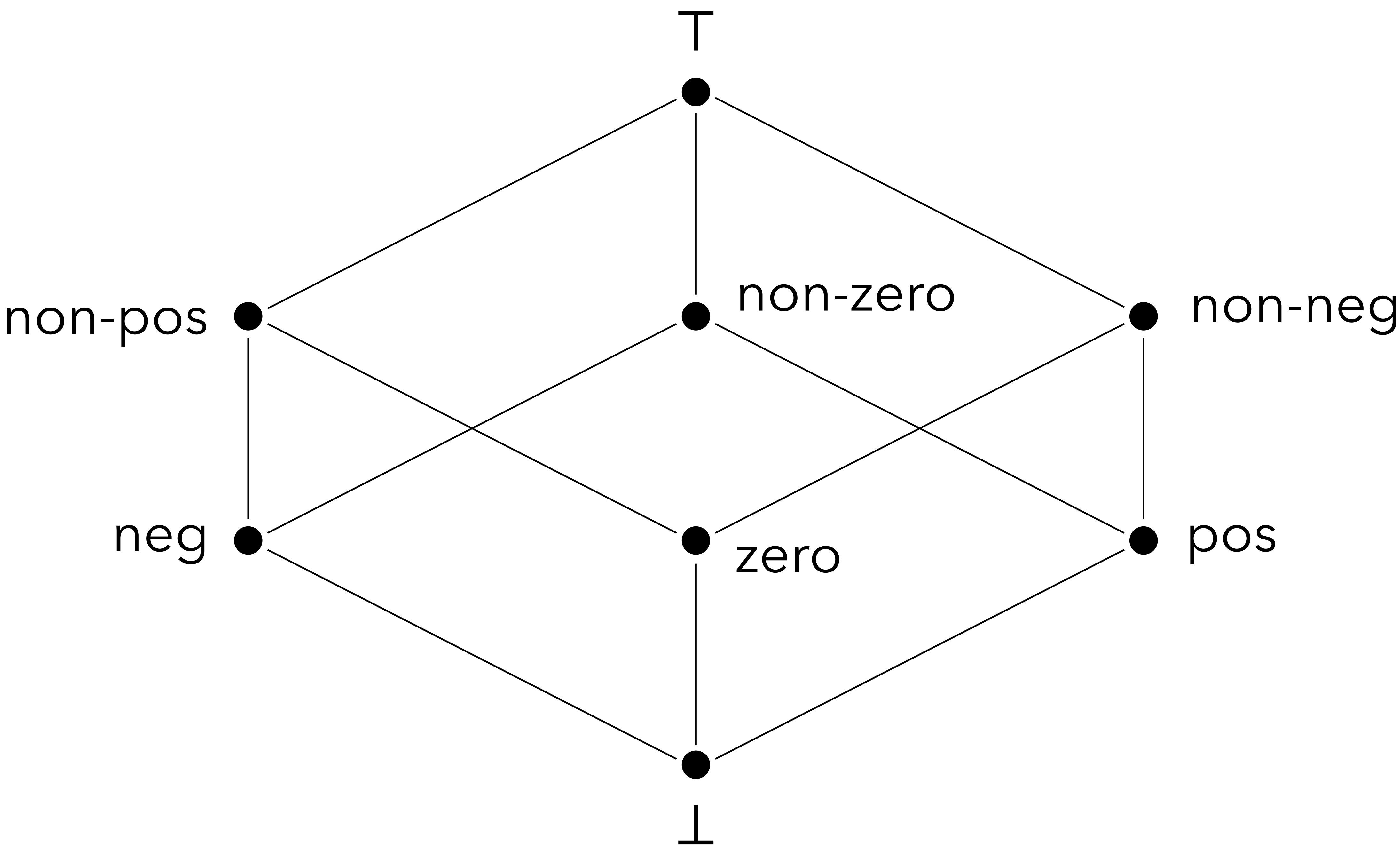
$W = \{ \cancel{1}, \cancel{2}, \cancel{3}, 4, \cancel{5}, \cancel{6}, 7 \}$

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);
```