IC637 Program Analysis

Lecture 2: Static Analysis Example 1 (Sign Domain)

Minseok Jeon

2025 Fall

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"
  - ...
- First, choose a domain (e.g., even or odd, etc.)

Principles of Static Analysis

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

- Second, execute the program with the chosen domain (i.e., abstract values)
- If the chosen domain is even (e) or odd (o), compute

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

with:

$\hat{\times}$	$\top$	e	o	$\bot$
$\top$	$\top$	e	$\top$	$\bot$
e	e	e	e	$\bot$
o	$\top$	e	o	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

$\hat{+}$	$\top$	e	o	$\bot$
$\top$	$\top$	$\top$	$\top$	$\bot$
e	$\top$	e	o	$\bot$
o	$\top$	o	e	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

Example: Even-Odd Domain

- In even-odd domain, abstract values are $\top$, e, o, and $\bot$.
- The abstract operations are defined as follows:

$\hat{\times}$	$\top$	e	o	$\bot$
$\top$	$\top$	e	$\top$	$\bot$
e	e	e	e	$\bot$
o	$\top$	e	o	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

$\hat{+}$	$\top$	e	o	$\bot$
$\top$	$\top$	$\top$	$\top$	$\bot$
e	$\top$	e	o	$\bot$
o	$\top$	o	e	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

$\hat{/}$	$\top$	e	o	$\bot$
$\top$				
e				
o				
$\bot$				

$\hat{-}$	$\top$	e	o	$\bot$
$\top$				
e				
o				
$\bot$				

$\hat{\%}$	$\top$	e	o	$\bot$
$\top$				
e				
o				
$\bot$				

Strength of Static Analysis

- Choose an abstract domain (e.g., even-odd domain)
- Execute the program with the chosen domain

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

Weakness of Static Analysis

- Static analysis may produce false positives (false alarms)

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

- If we use even-odd domain, the analysis is unable to prove the assertion.
- **Question:** Is there a domain that can prove the assertion?

Weakness of Static Analysis

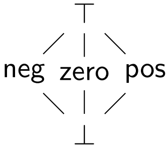
- Example showing a limitation of even-odd domain

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

- Which abstract domain should we use to prove the assertions?

Example : Sign Domain

- Abstract values :



- Abstract operations:

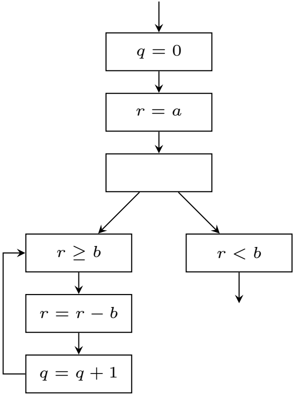
$\hat{+}$	$\top$	pos	zero	neg	$\bot$
$\top$					
pos					
zero					
neg					
$\bot$					

$\hat{-}$	$\top$	pos	zero	neg	$\bot$
$\top$					
pos					
zero					
neg					
$\bot$					

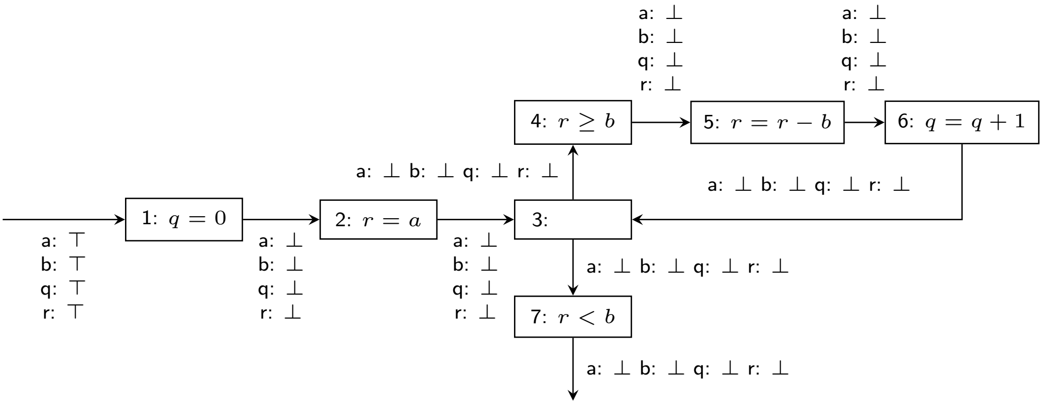
Example : Sign Domain

- 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);
```

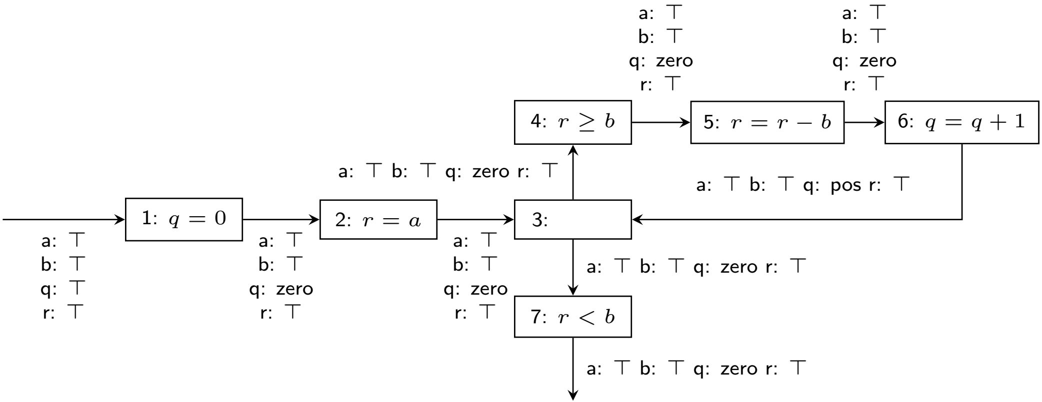


Fixed Point Computation (Iteration)



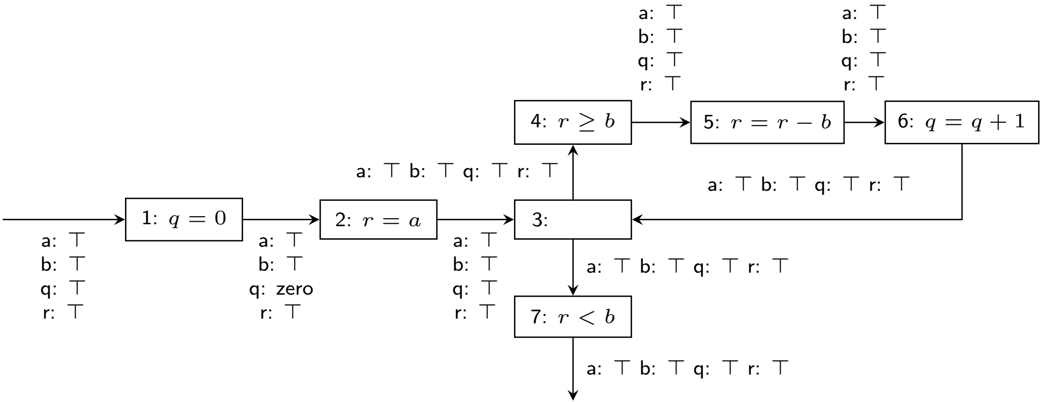
#iteration $= 0$

Fixed Point Computation (Iteration)



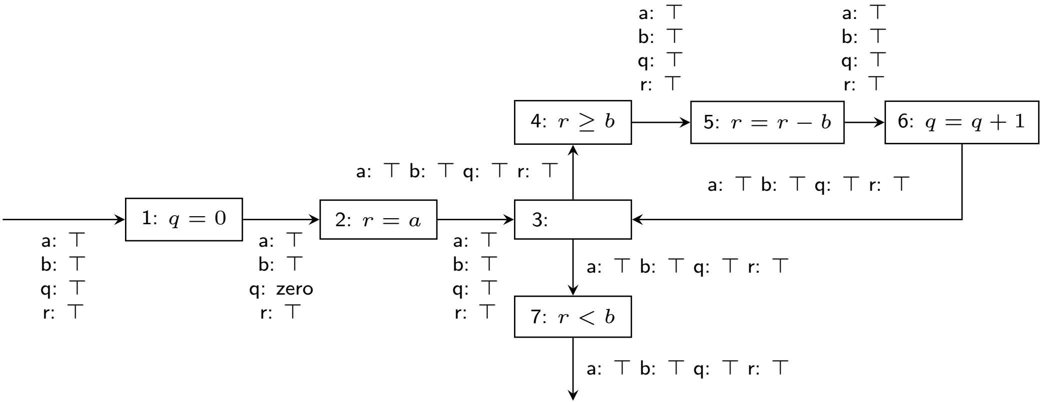
$\#\text{iteration} = 1$

Fixed Point Computation (Iteration)



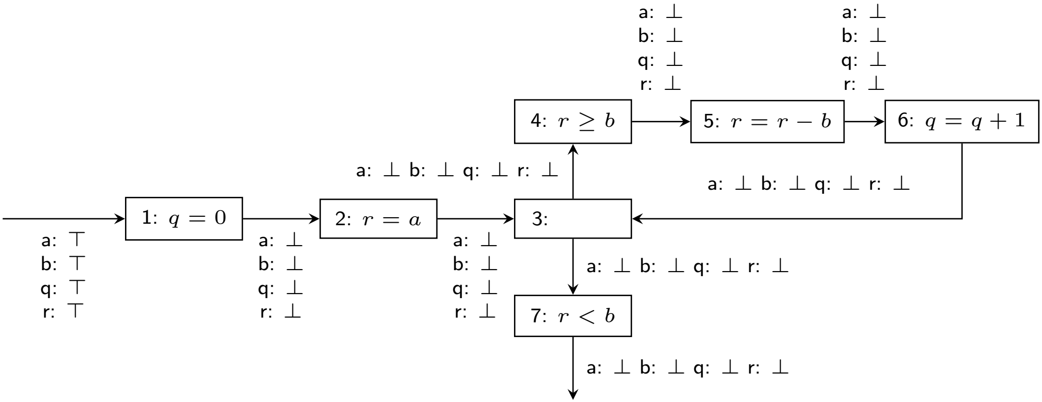
#iteration = 2

Fixed Point Computation (Iteration)



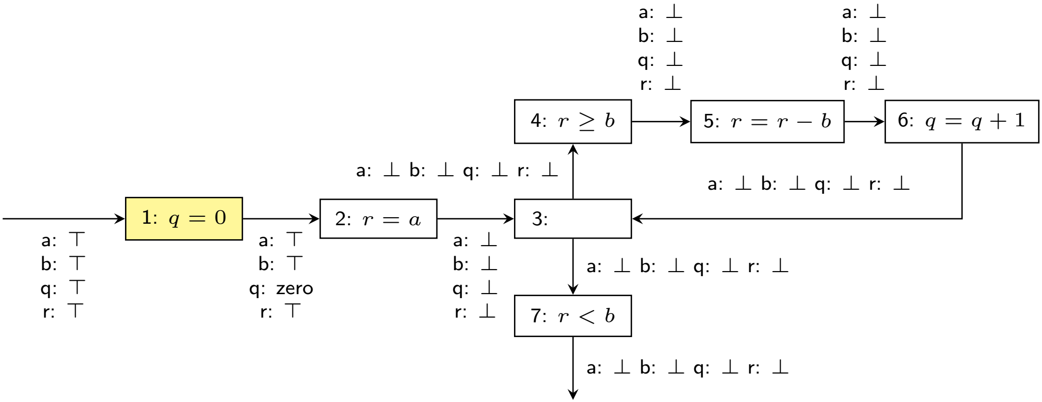
$\#\text{iteration} = 3$

Fixed Point Computation (Worklist)



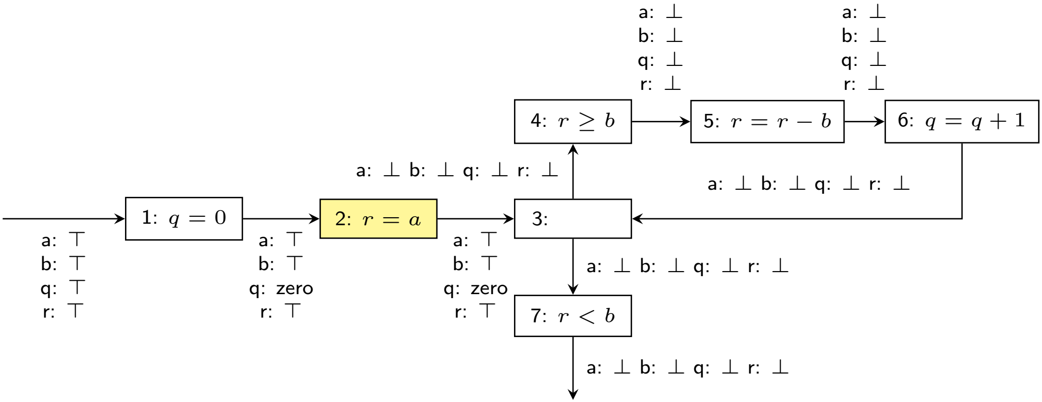
Worklist $= \{1, 2, 3, 4, 5, 6, 7\}$

Fixed Point Computation (Worklist)



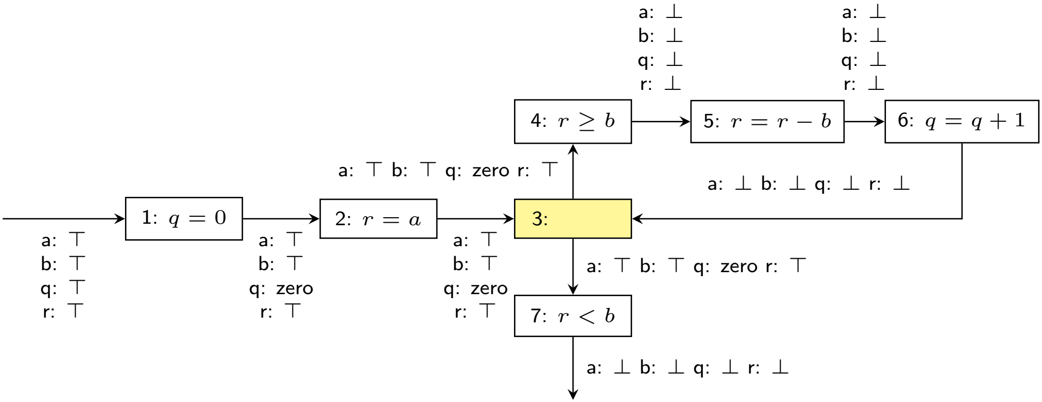
Worklist $= \{\cancel{1}, 2, 3, 4, 5, 6, 7\}$

Fixed Point Computation (Worklist)



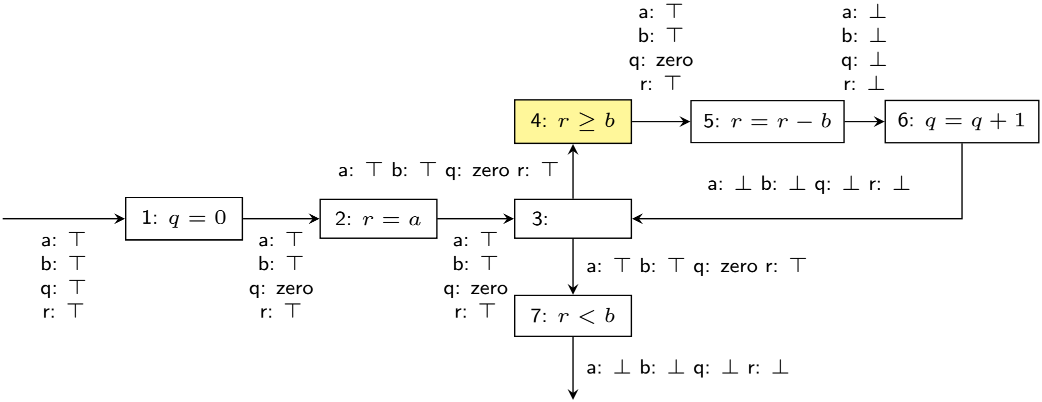
Worklist $= \{\cancel{1}, \cancel{2}, 3, 4, 5, 6, 7\}$

Fixed Point Computation (Worklist)



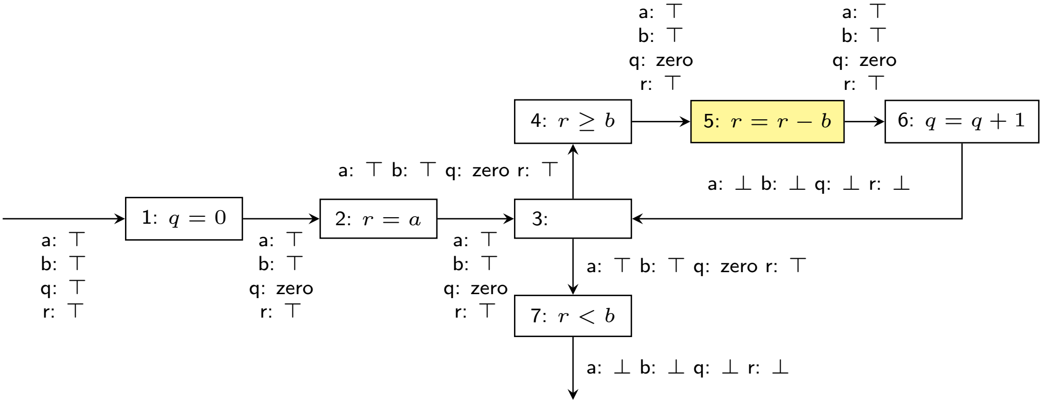
Worklist $= \{\cancel{1}, \cancel{2}, \cancel{3}, 4, 5, 6, 7\}$

Fixed Point Computation (Worklist)



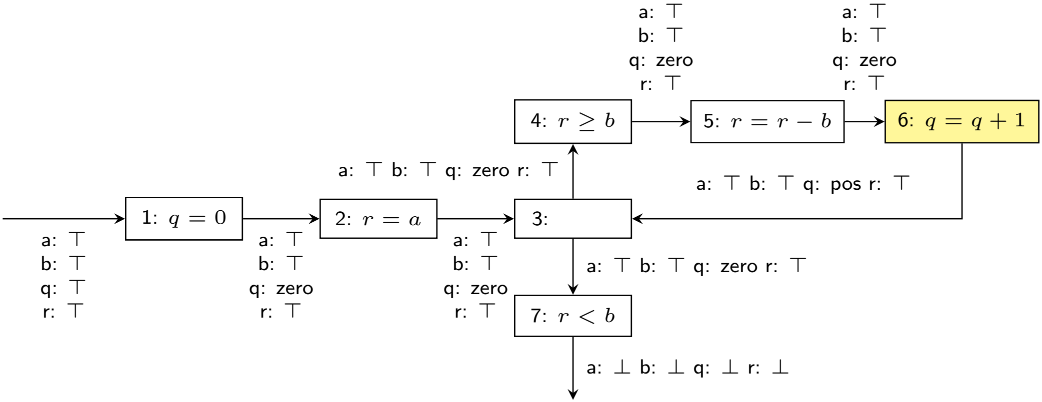
Worklist $= \{\cancel{1}, \cancel{2}, \cancel{3}, \cancel{4}, 5, 6, 7\}$

Fixed Point Computation (Worklist)



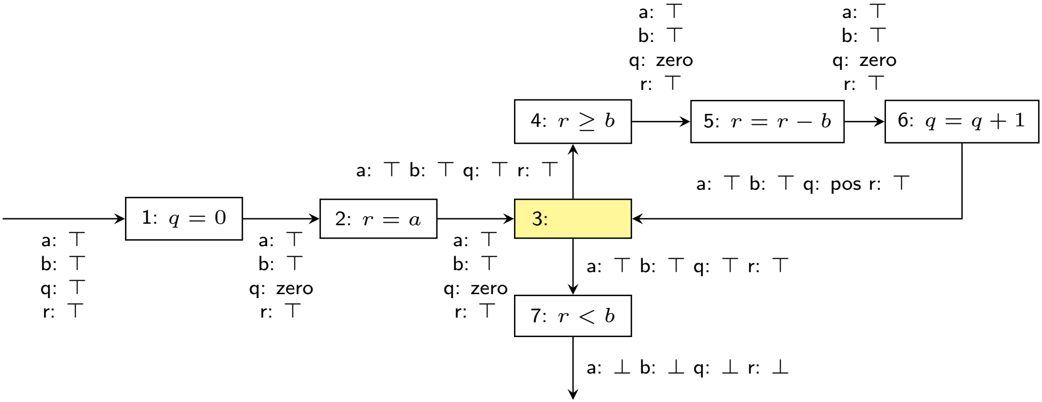
Worklist $= \{\cancel{1}, \cancel{2}, \cancel{3}, \cancel{4}, \cancel{5}, 6, 7\}$

Fixed Point Computation (Worklist)



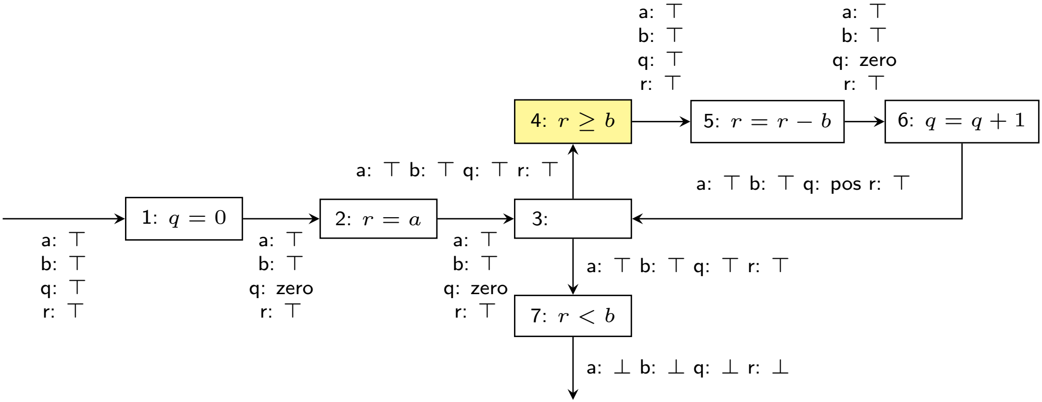
Worklist $= \{\cancel{1}, \cancel{2}, 3, \cancel{4}, \cancel{5}, \cancel{6}, 7\}$

Fixed Point Computation (Worklist)



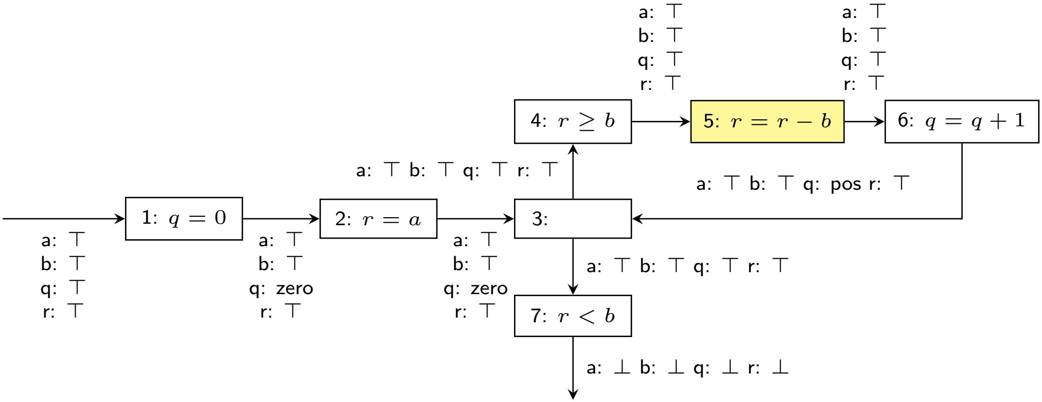
Worklist $= \{\cancel{1}, \cancel{2}, \cancel{3}, 4, \cancel{5}, \cancel{6}, 7\}$

Fixed Point Computation (Worklist)



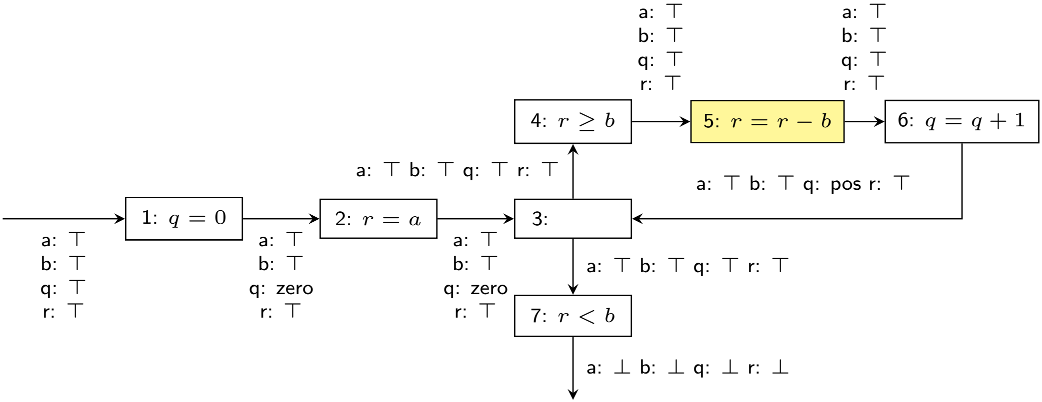
Worklist $= \{\cancel{1}, \cancel{2}, \cancel{3}, \cancel{4}, 5, \cancel{6}, 7\}$

Fixed Point Computation (Worklist)



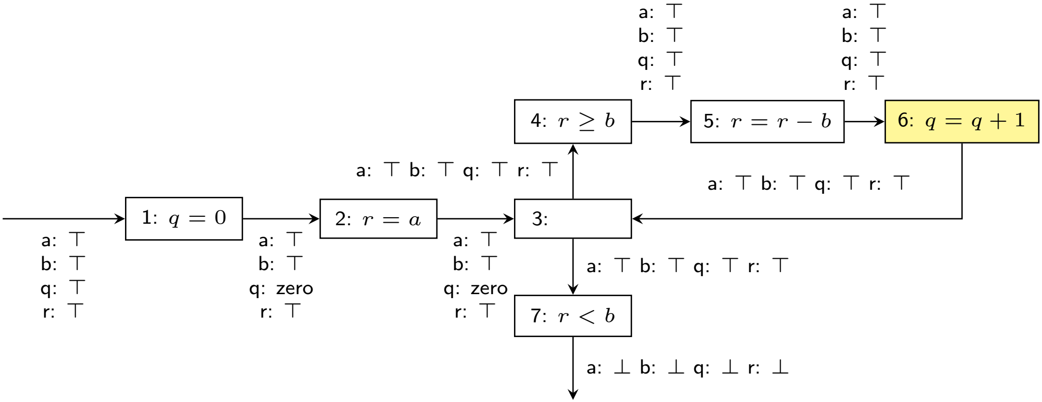
Worklist $= \{\cancel{1}, \cancel{2}, 3, \cancel{4}, \cancel{5}, \cancel{6}, 7\}$

Fixed Point Computation (Worklist)



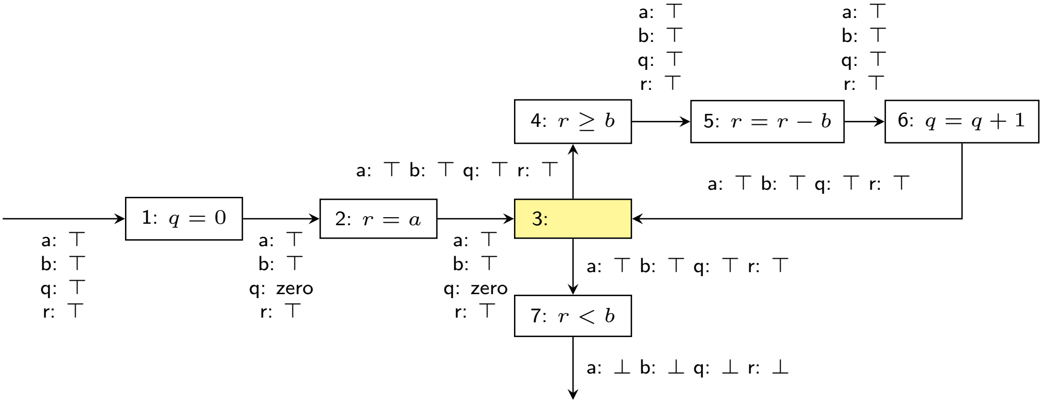
Worklist $= \{\cancel{1}, \cancel{2}, 3, \cancel{4}, \cancel{5}, \cancel{6}, 7\}$

Fixed Point Computation (Worklist)



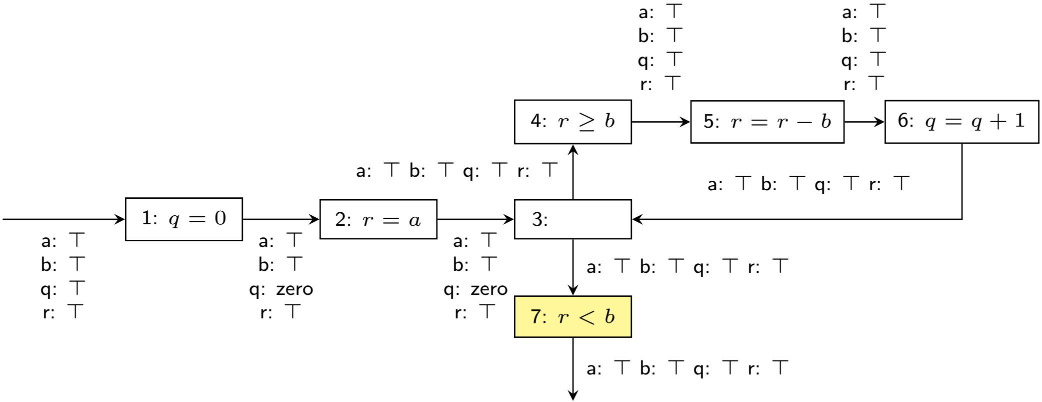
Worklist $= \{\cancel{1}, \cancel{2}, 3, \cancel{4}, \cancel{5}, \cancel{6}, 7\}$

Fixed Point Computation (Worklist)



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

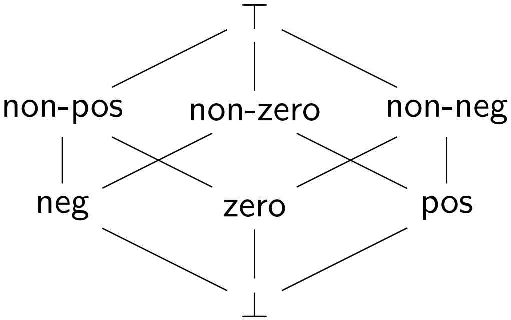
Fixed Point Computation (Worklist)



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

Extended Sign Domain

- Abstract values :



Extended Sign Domain

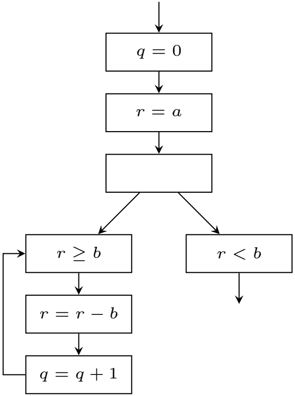
$\hat{+}$	$\top$	pos	zero	neg	non-pos	non-zero	non-neg	$\bot$
$\top$								
pos								
zero								
neg								
non-pos								
non-zero								
non-neg								
$\bot$								

$\hat{*}$	$\top$	pos	zero	neg	non-pos	non-zero	non-neg	$\bot$
$\top$								
pos								
zero								
neg								
non-pos								
non-zero								
non-neg								
$\bot$								

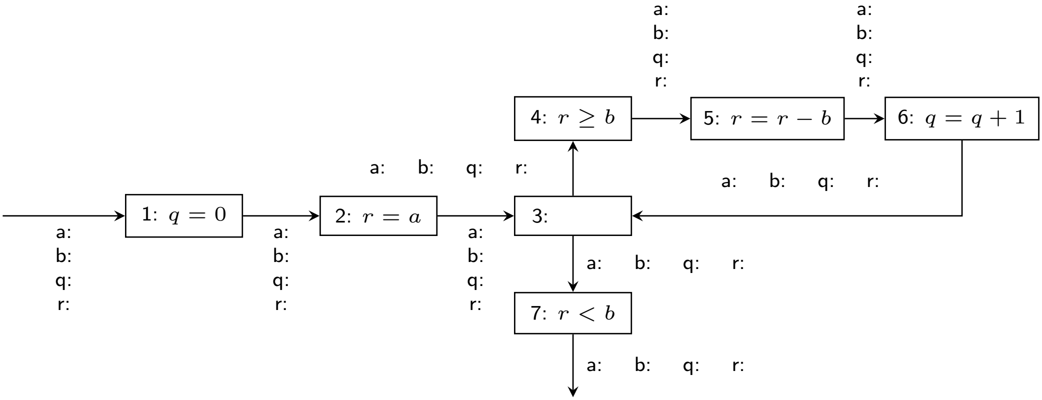
Extended Sign Domain

- 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);
```



Extended Sign Domain



Question

- Provide an example that shows a limitation of extended sign domain.

Summary

- **Static Analysis Principles**
  - Choose an abstract domain (e.g., even-odd, sign, extended sign)
  - Execute program with abstract values instead of concrete values
  - Trade precision for automation and scalability
- **Abstract Domains Covered**
  - **Even-Odd Domain**: $\{\top, e, o, \bot\}$ with abstract operations $\hat{+}, \hat{\times}, \hat{-}, \hat{/}, \hat{\%}$
  - **Sign Domain**: $\{\top, \text{pos}, \text{zero}, \text{neg}, \bot\}$
  - **Extended Sign Domain**: Adds $\{\text{non-pos}, \text{non-zero}, \text{non-neg}\}$ for more precision
- **Fixed-Point Computation**
  - **Iterative approach**: Compute until convergence
  - **Worklist algorithm**: Process nodes efficiently, add to worklist when values change
  - Guarantees termination due to finite lattice height