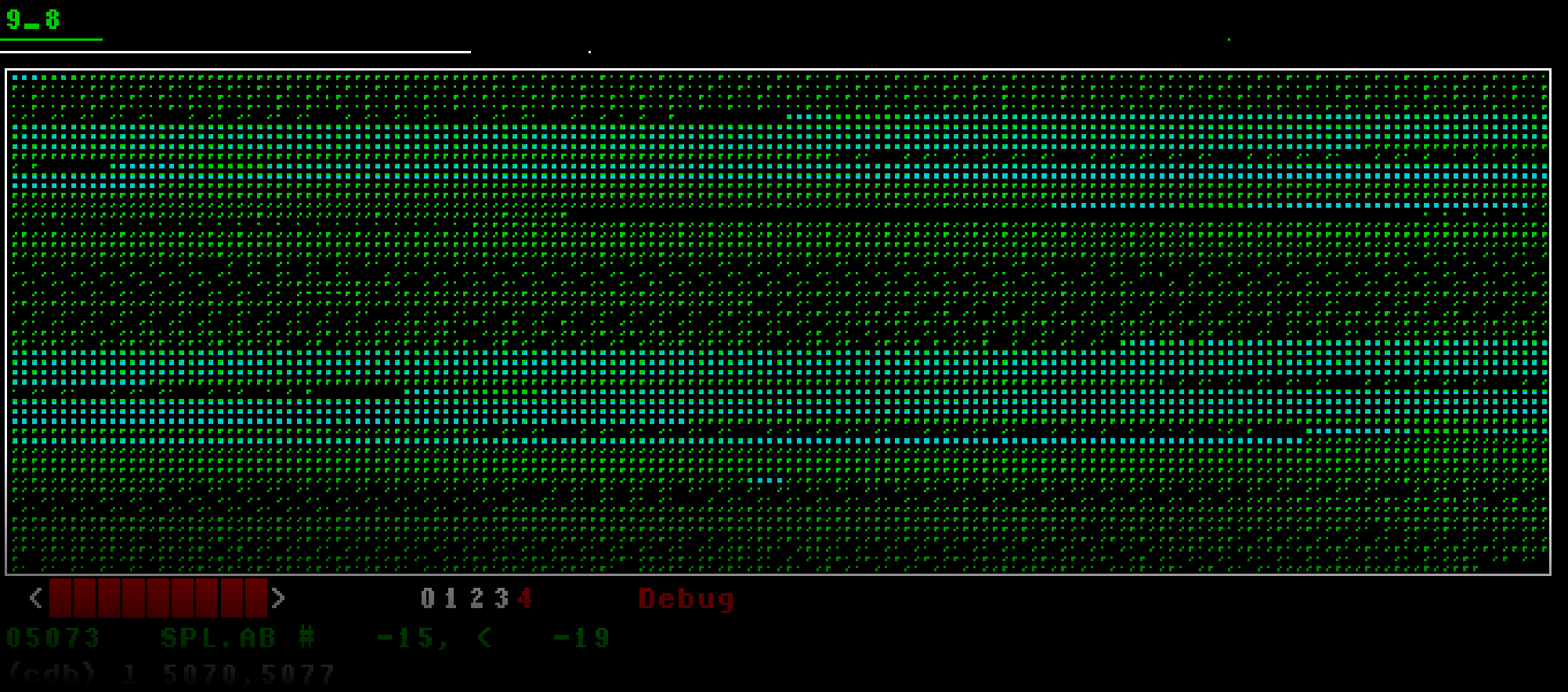
Heap Invariants for LLVM IR

Cornelius Aschermann

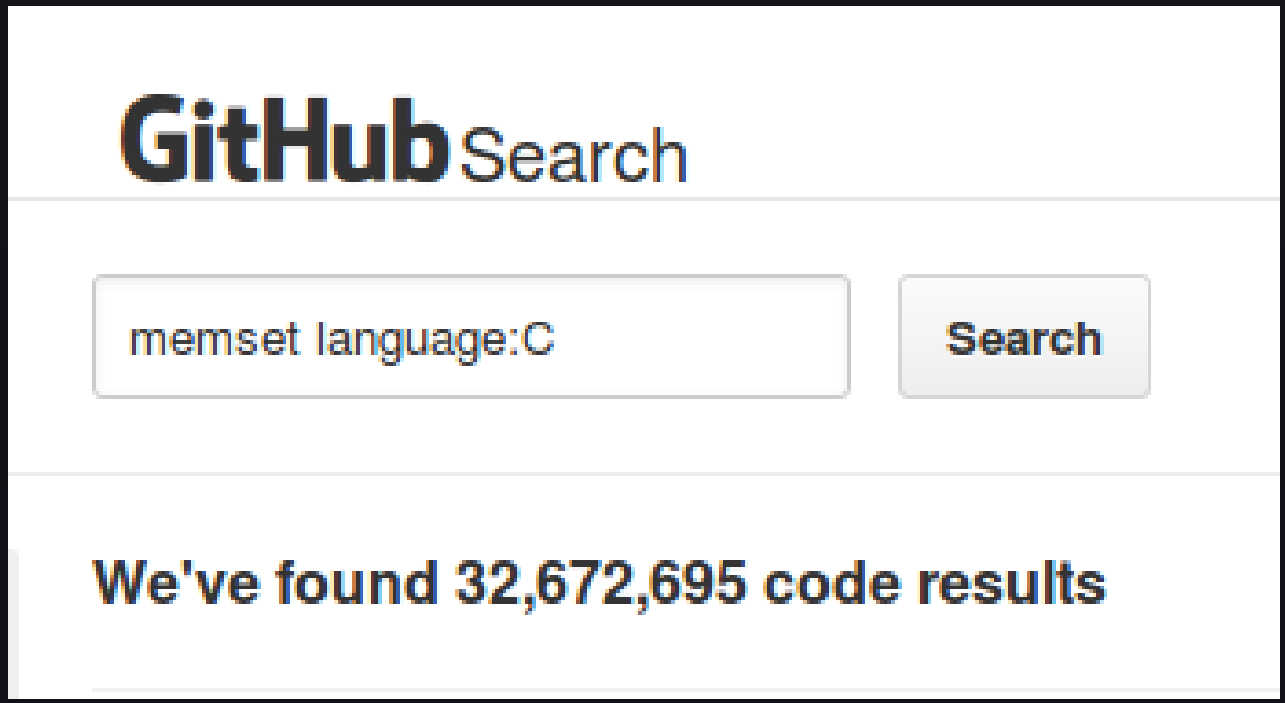
LuFG I2 RWTH Aachen

March 26, 2014

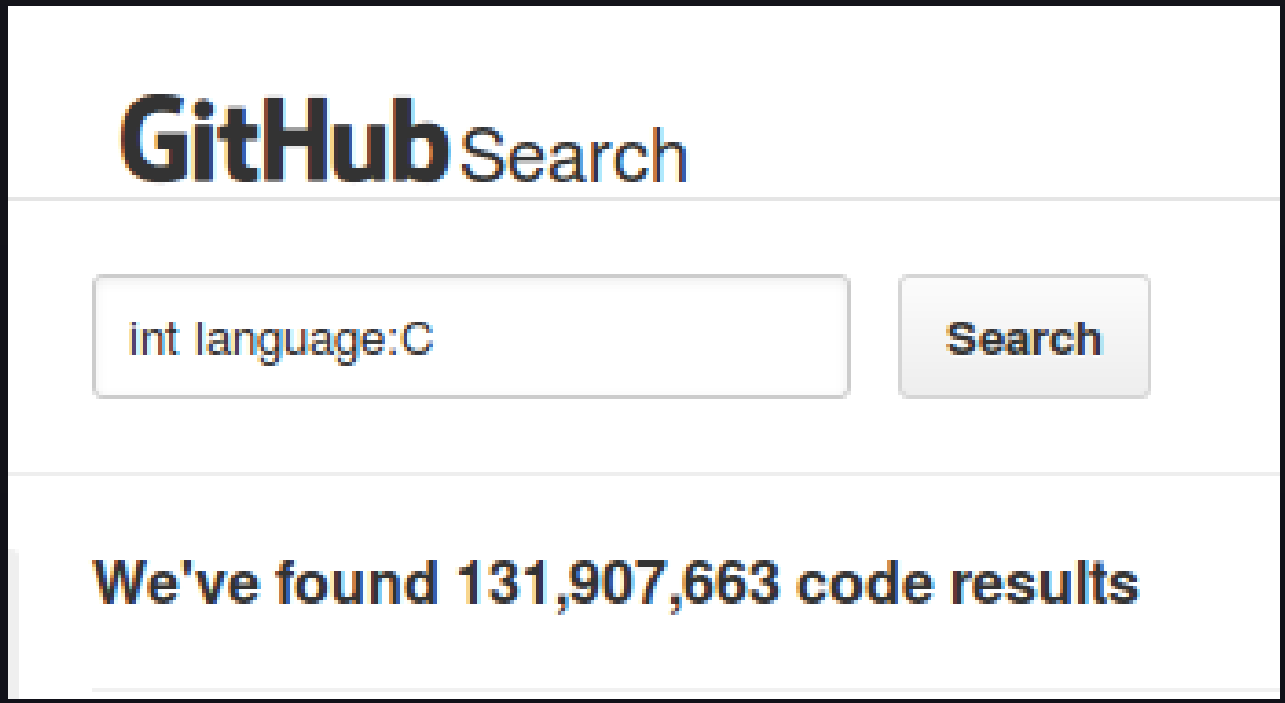


memset

relevance?



memset $3.2 * 10^7$



memset $3.2 * 10^7$

int $13 * 10^7$

Related Work

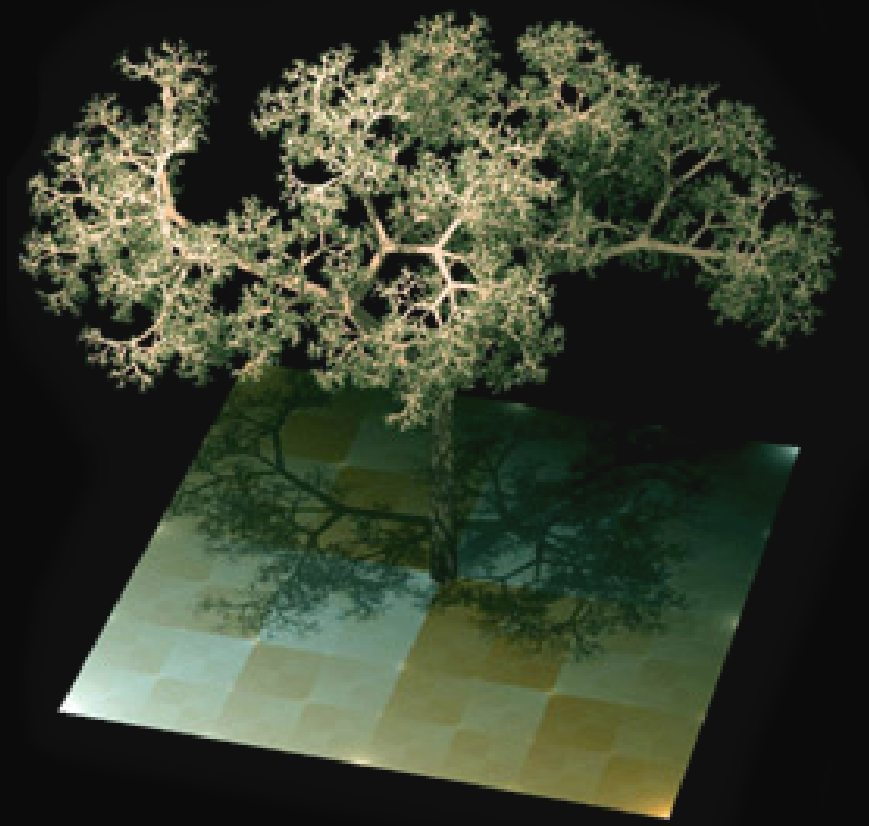
```
struct list_node {
    int data;
    struct list_node *tail;
};
typedef struct list_node list;

list *reverse(list *l) {
    list *r = l, *p = NULL;
    while (r != NULL) {
        list *q = r;
        r = r->tail;
        q->tail = p;
        p = q;
    }
    return p;
}
```

Background

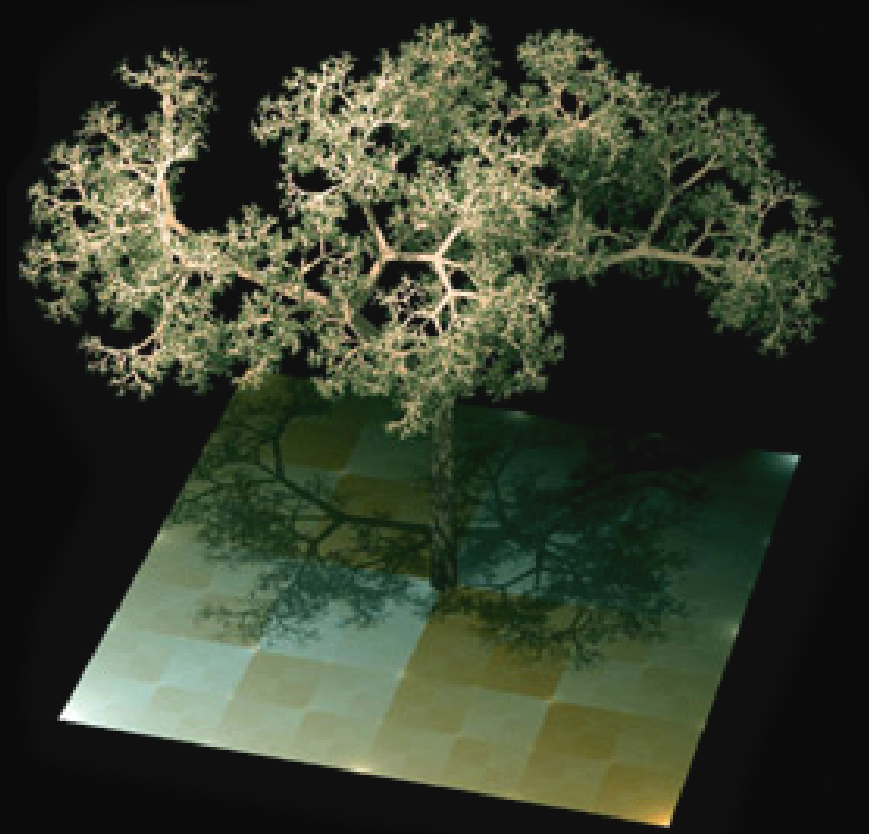

AProVE

Term Rewriting

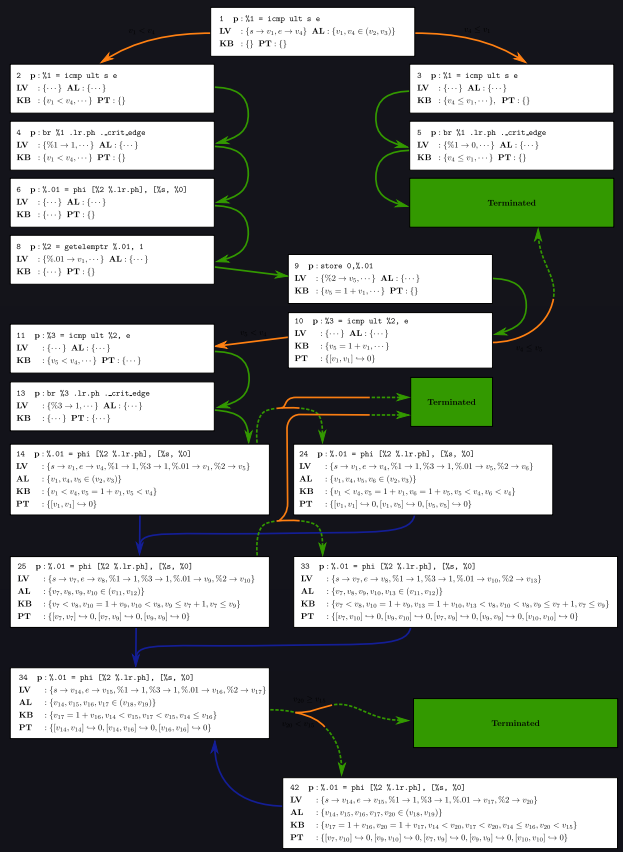


$$f(a,b) \to f(b,c) \quad :|: c = a \bmod b \wedge b \neq 0$$

$$f(a,b) \to g(a) \quad :|: b = 0$$

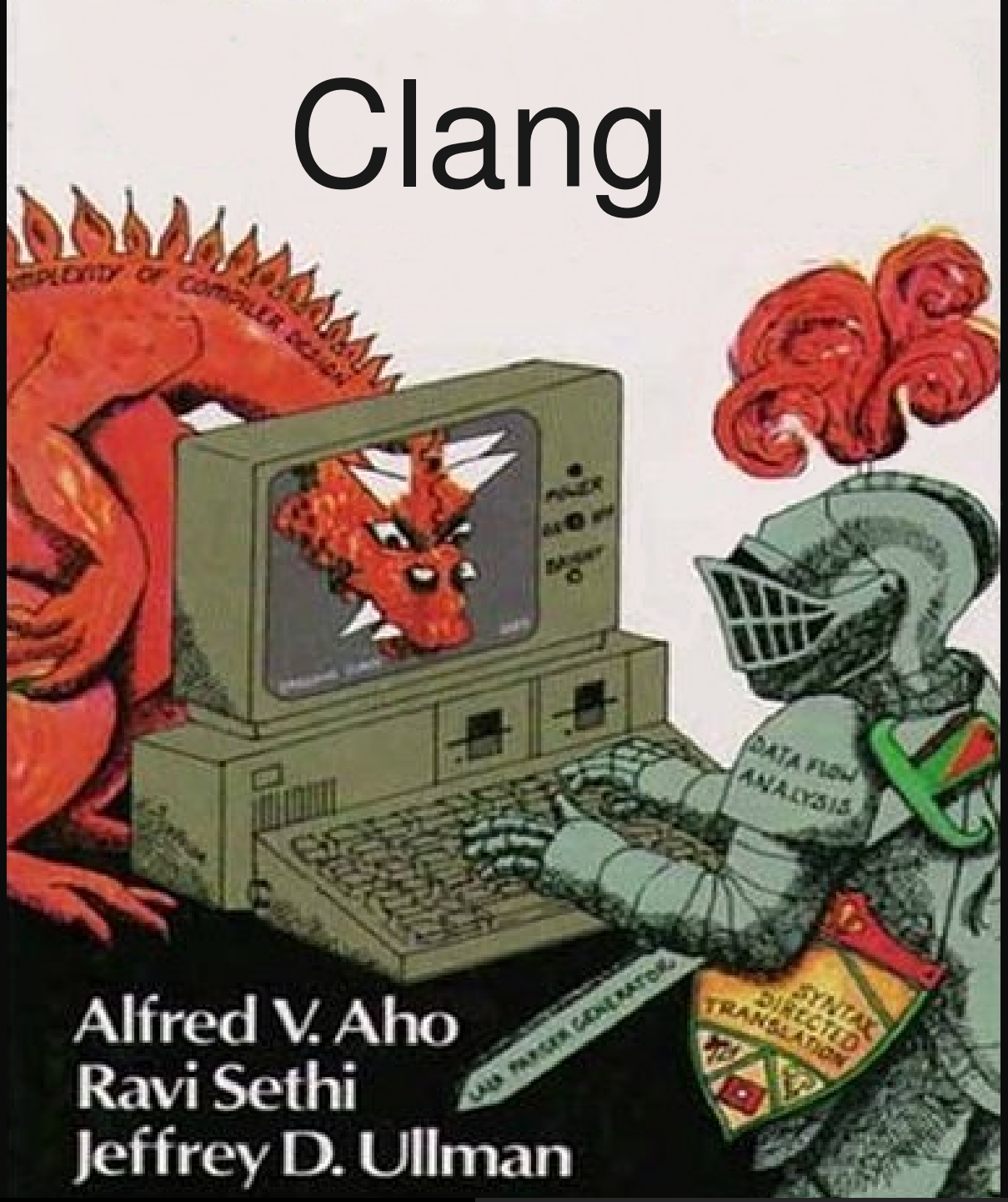


Symbolic Execution

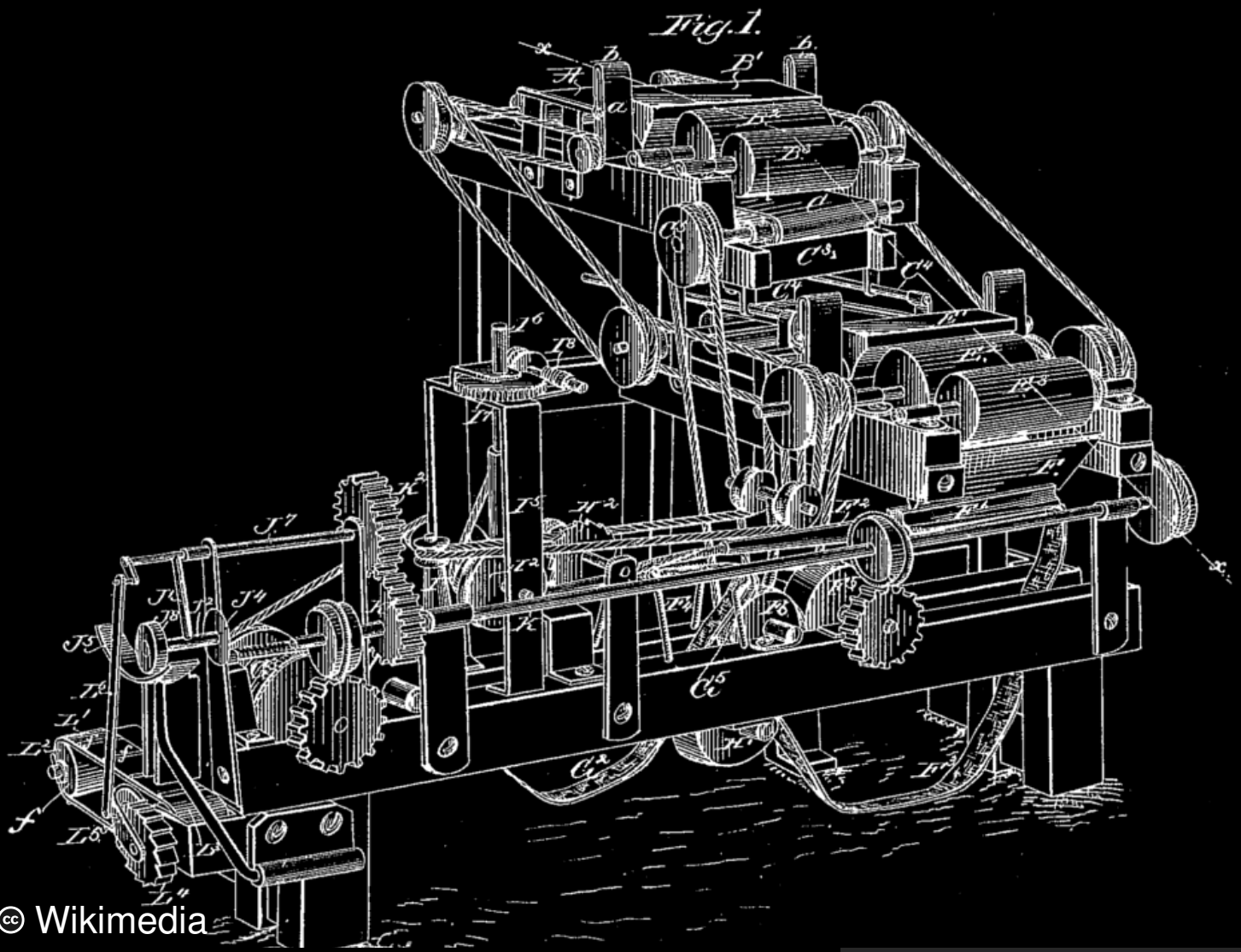


LLVM

Clang



LLVM IR



Assembler

Symbolically Executing LLVM IR

Arithmetic

$x = 5, y \geq 1$

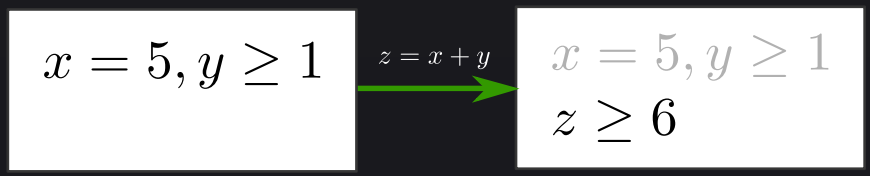
Arithmetic

```
z = add x, y
```

$x = 5, y \geq 1$

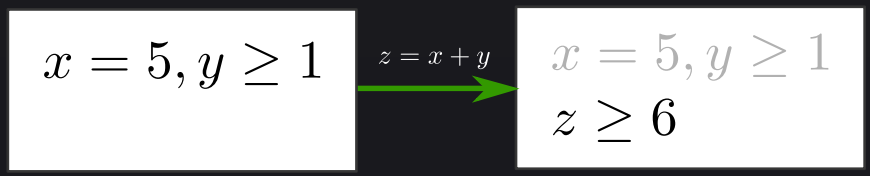
Arithmetic

```
z = add x, y
```



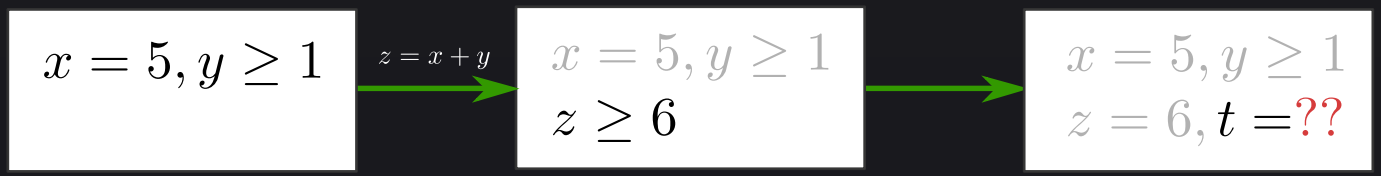
Arithmetic

```
t = icmp ule z, 6
```

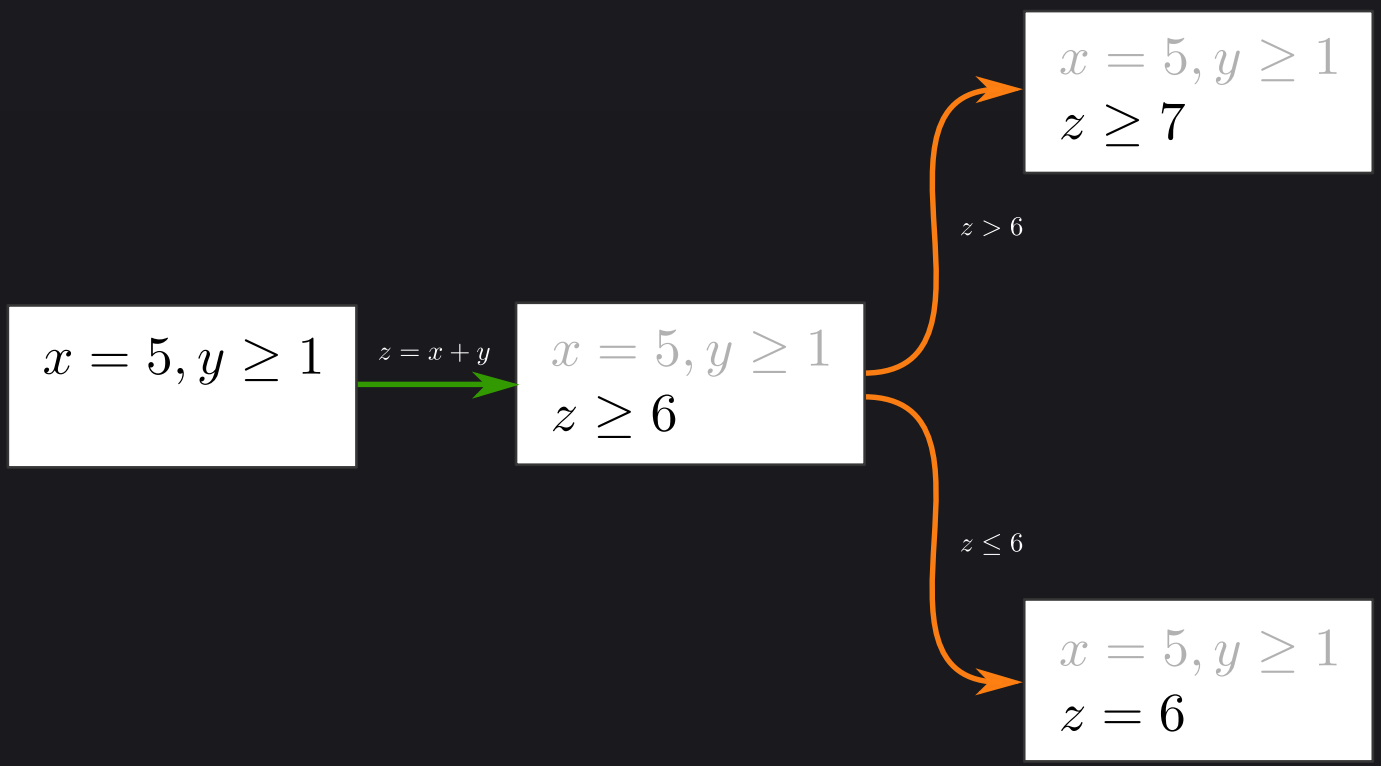


Arithmetic

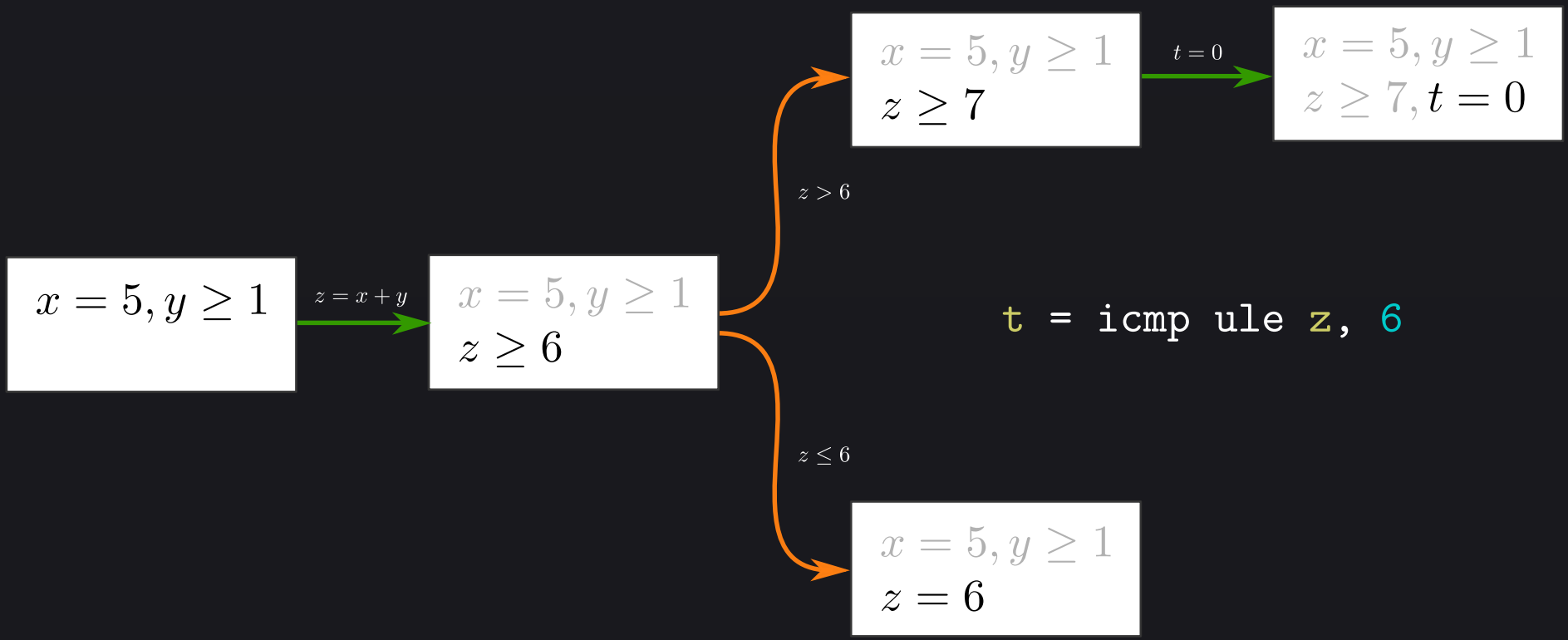
```
t = icmp ule z, 6
```



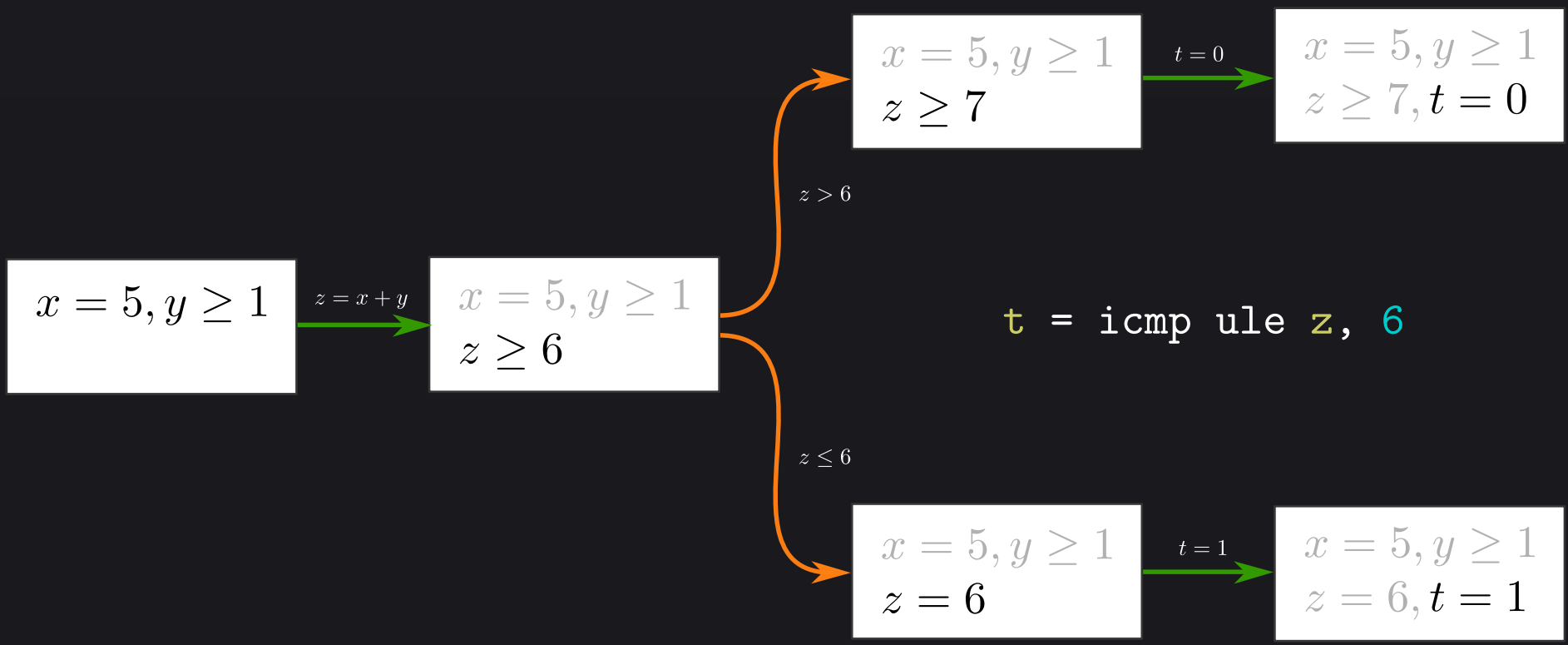
Arithmetic


$x = 5, y \geq 1$
$z = x + y$
$x = 5, y \geq 1$
$z \geq 6$
$z > 6$
$x = 5, y \geq 1$
$z \geq 7$
$z \leq 6$
$x = 5, y \geq 1$
$z = 6$

Arithmetic


$x = 5, y \geq 1$
$z = x + y$
$x = 5, y \geq 1$
$z \geq 6$
$z > 6$
$x = 5, y \geq 1$
$z \geq 7$
$t = 0$
$x = 5, y \geq 1$
$z \geq 7, t = 0$
$z \leq 6$
$x = 5, y \geq 1$
$z = 6$
t = icmp ule z, 6

Arithmetic


$x = 5, y \geq 1$
$z = x + y$
$x = 5, y \geq 1$
$z \geq 6$
$z > 6$
$x = 5, y \geq 1$
$z \geq 7$
$t = 0$
$x = 5, y \geq 1$
$z \geq 7, t = 0$
$z \leq 6$
$x = 5, y \geq 1$
$z = 6$
$t = 1$
$x = 5, y \geq 1$
$z = 6, t = 1$
t = icmp ule z, 6

Memory

$A : x \in (v_1, v_2)$

$K : x + 3 < v_2, y = 0$

Memory

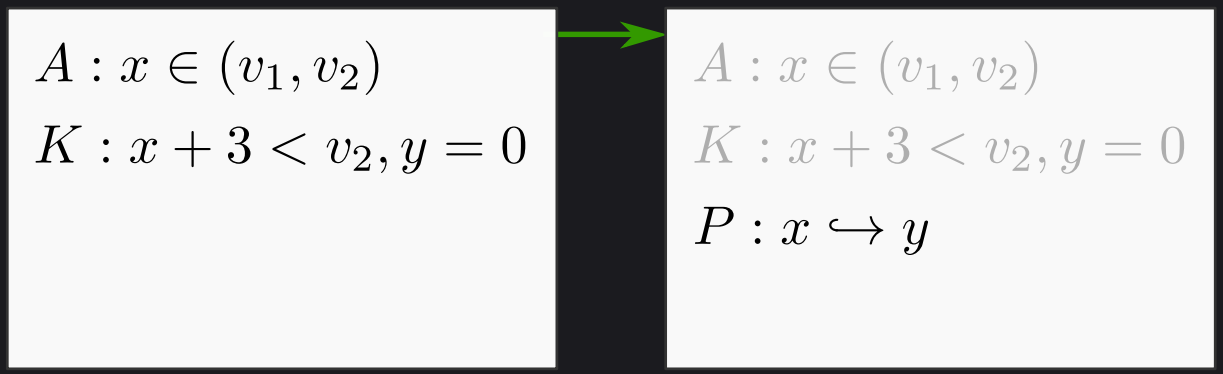
```
store y, x
```

$A : x \in (v_1, v_2)$

$K : x + 3 < v_2, y = 0$

Memory

`store y, x`



Memory What If?

```
store y, x
```

$A : x \in (v_1, v_2)$

$K : x + 3 < v_2, y = 0$

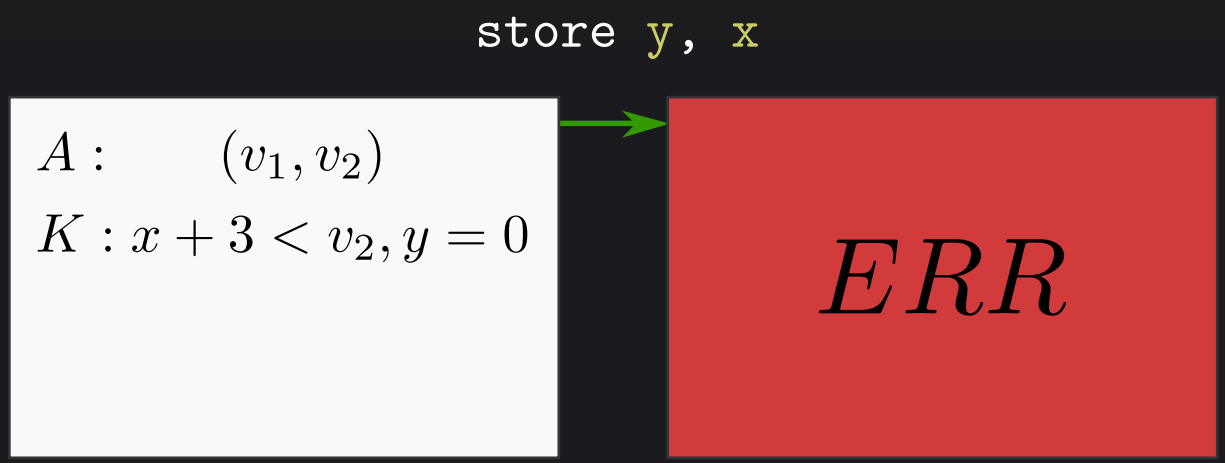
Memory What If?

```
store y, x
```

$A: \quad (v_1, v_2)$

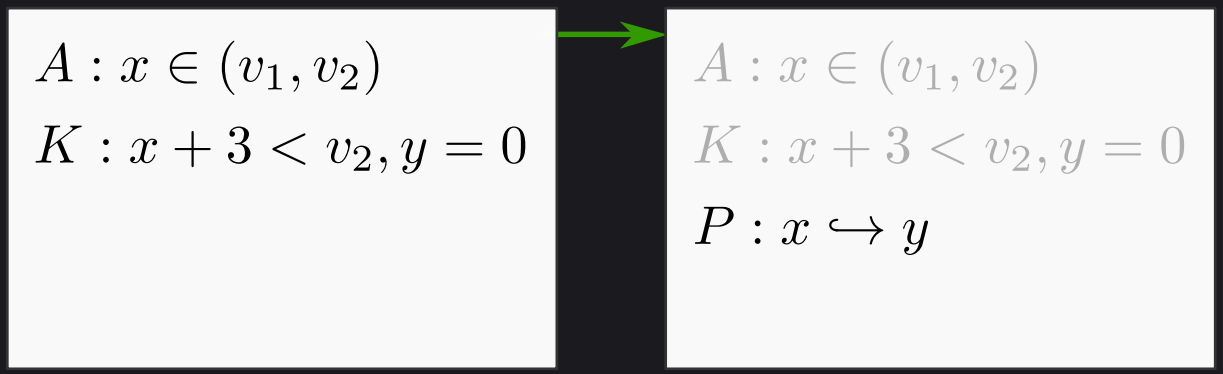
$K: x + 3 < v_2, y = 0$

Memory



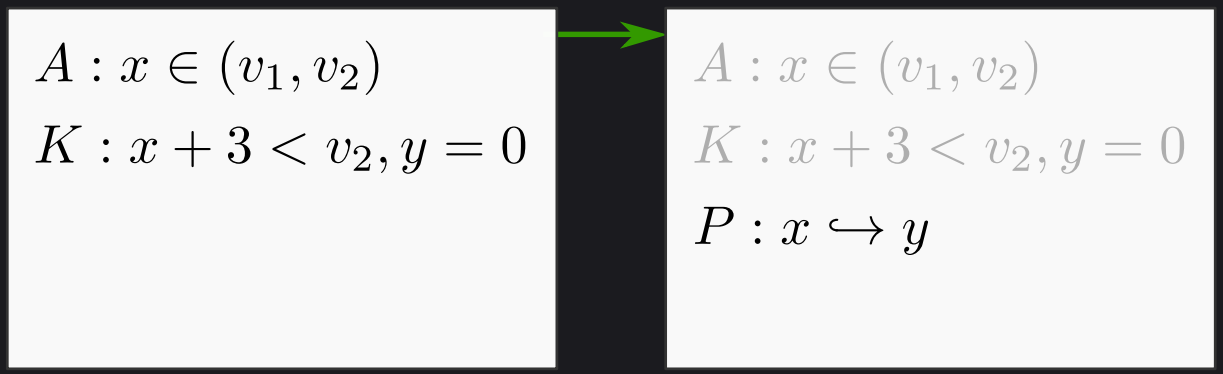
Memory

`store y, x`

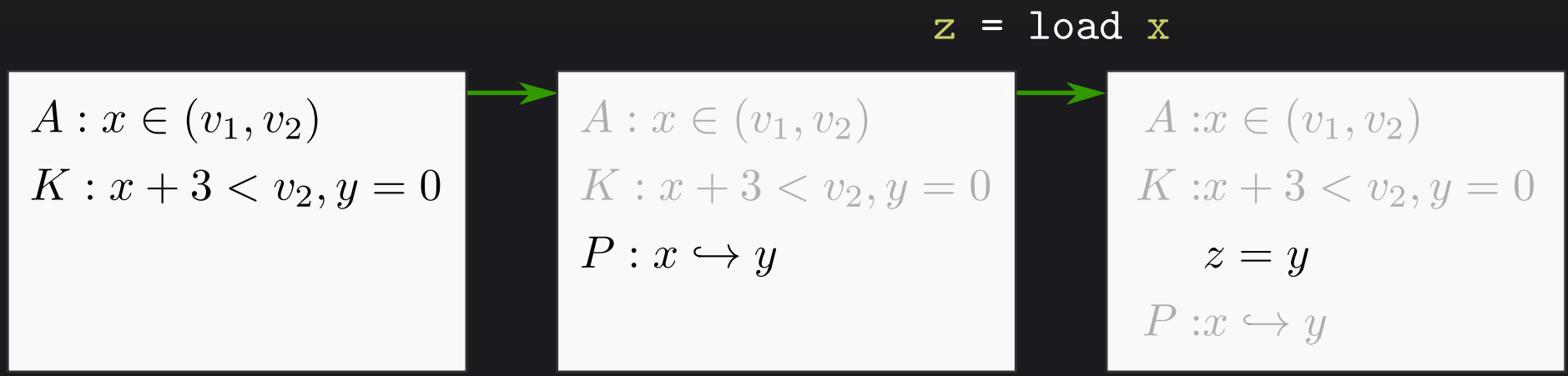


Memory

```
z = load x
```

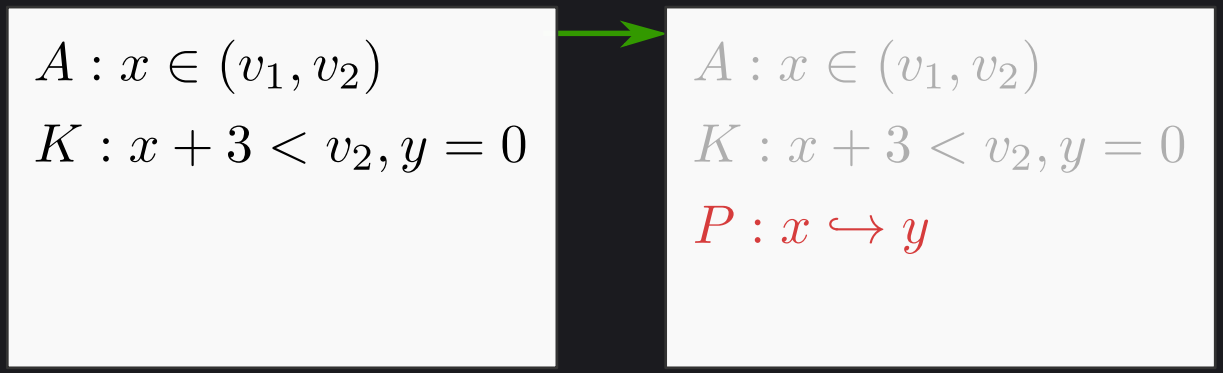


Memory


z = load x
$A : x \in (v_1, v_2)$
$K : x + 3 < v_2, y = 0$
$A : x \in (v_1, v_2)$
$K : x + 3 < v_2, y = 0$
$P : x \hookrightarrow y$
$A : x \in (v_1, v_2)$
$K : x + 3 < v_2, y = 0$
$z = y$
$P : x \hookrightarrow y$

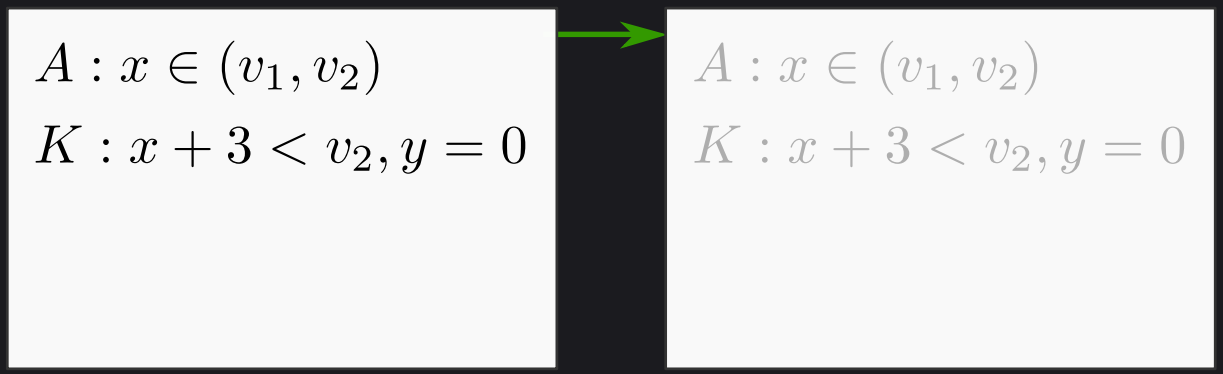
Memory What If?

```
z = load x
```

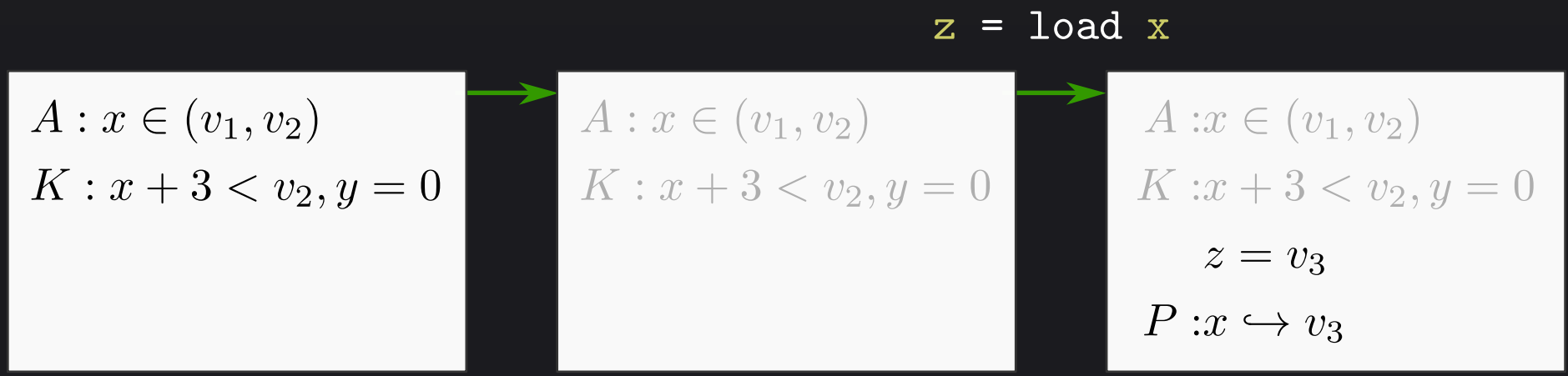


Memory What If?

```
z = load x
```



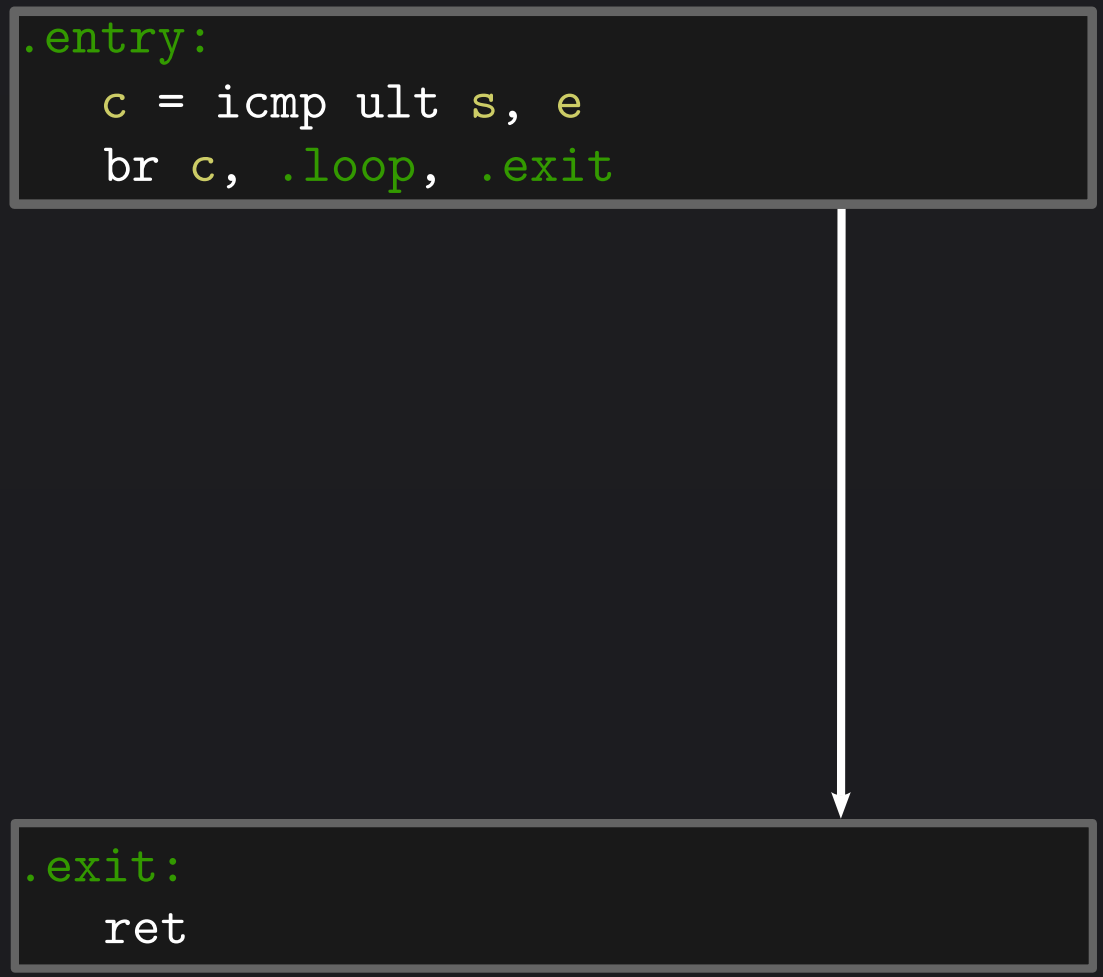
Memory


z = load x
$A : x \in (v_1, v_2)$
$K : x + 3 < v_2, y = 0$
$A : x \in (v_1, v_2)$
$K : x + 3 < v_2, y = 0$
$A : x \in (v_1, v_2)$
$K : x + 3 < v_2, y = 0$
$z = v_3$
$P : x \hookrightarrow v_3$

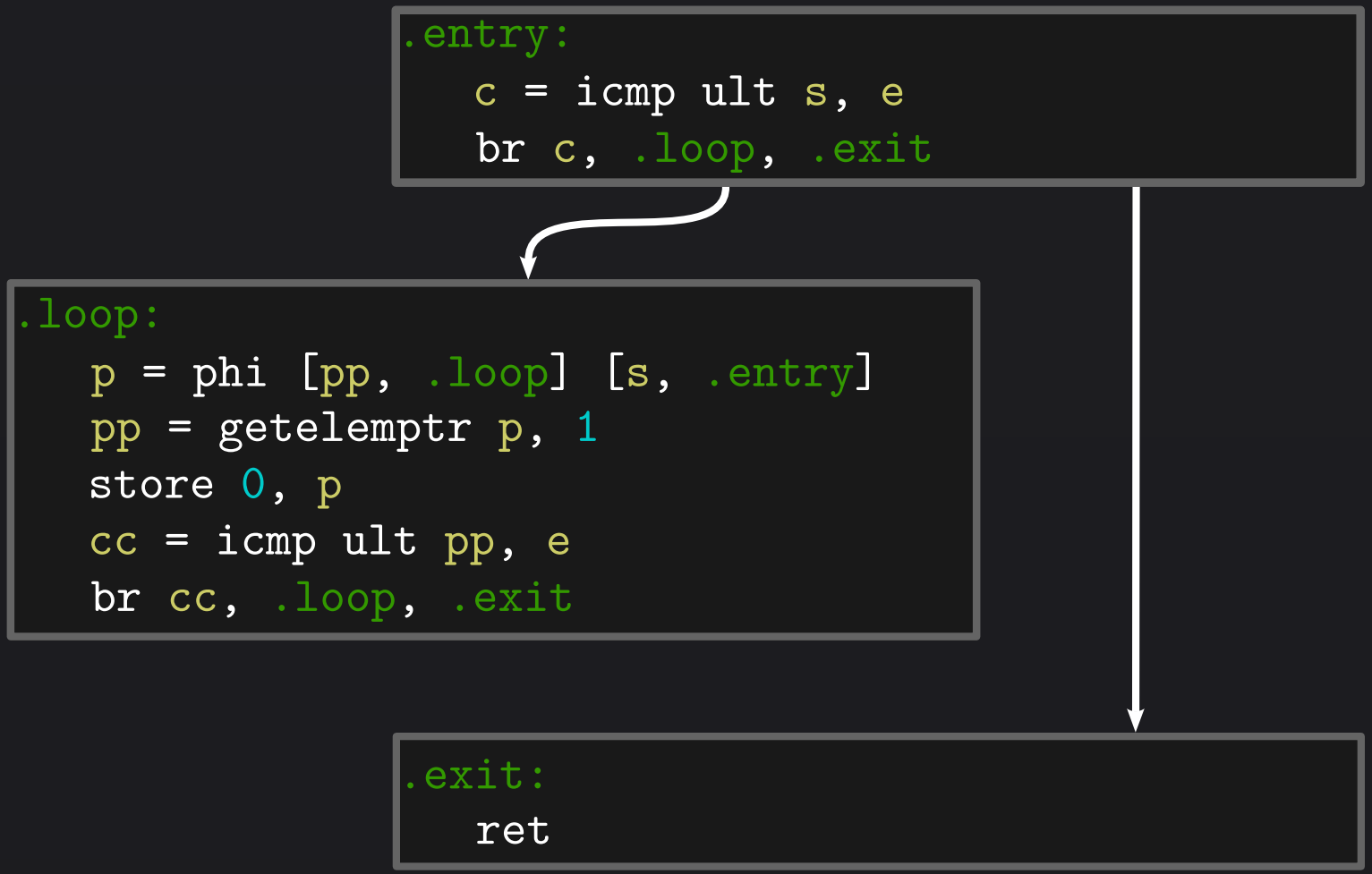
Basic Blocks

```
.entry:
  c = icmp ult s, e
  br c, .loop, .exit
```

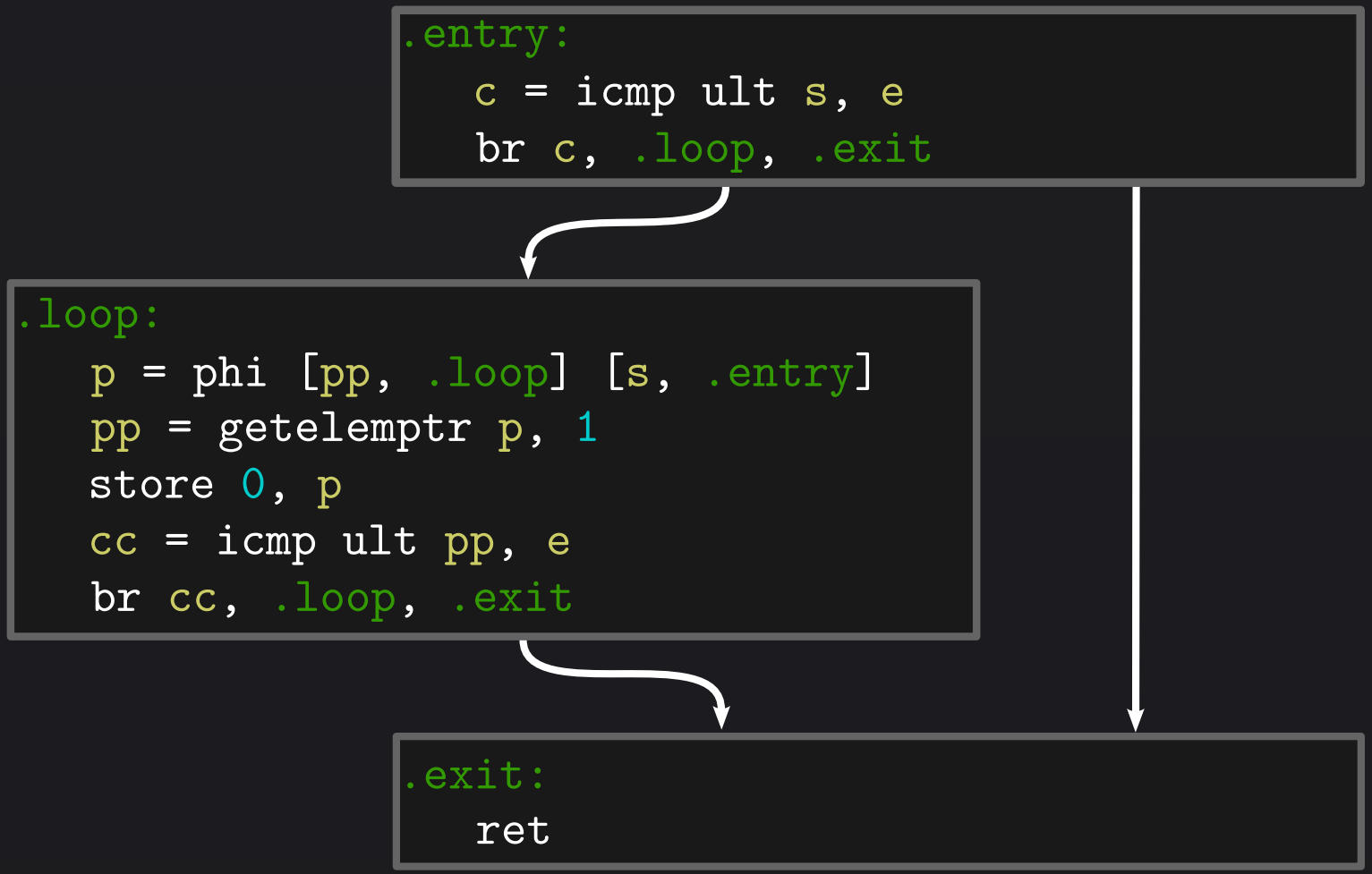
Basic Blocks



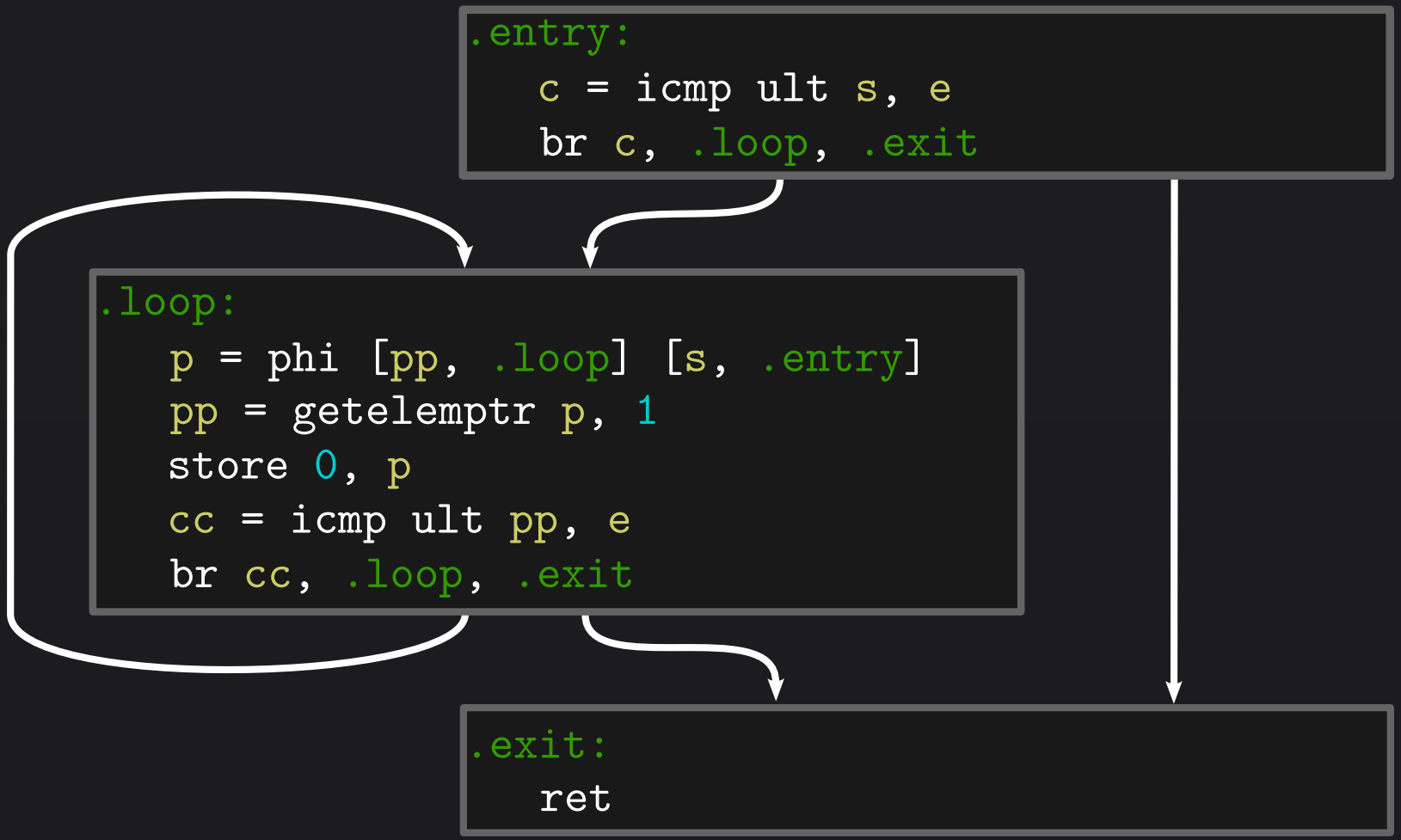
Basic Blocks



Basic Blocks



Basic Blocks



Code

```
   .entry:
1   c = icmp ult s, e
2   br c, .loop, .exit
   .loop:
3   p = phi [pp, .loop] [s, .entry]
4   pp = getelemptr p, 1
5   store 0, p
6   cc = icmp ult pp, e
7   br cc, .loop, .exit
   .exit:
8   ret
```

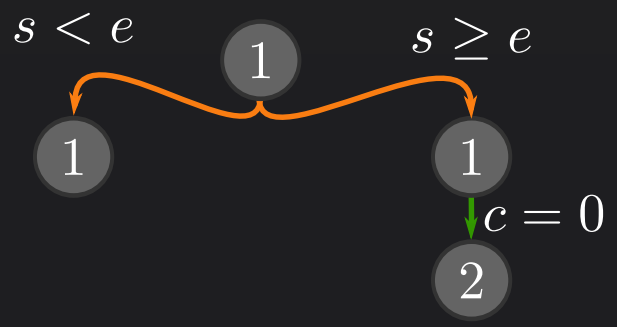
1

Code

```
   .entry:
1   c = icmp ult s, e
2   br c, .loop, .exit
   .loop:
3   p = phi [pp, .loop] [s, .entry]
4   pp = getelemptr p, 1
5   store 0, p
6   cc = icmp ult pp, e
7   br cc, .loop, .exit
   .exit:
8   ret
```

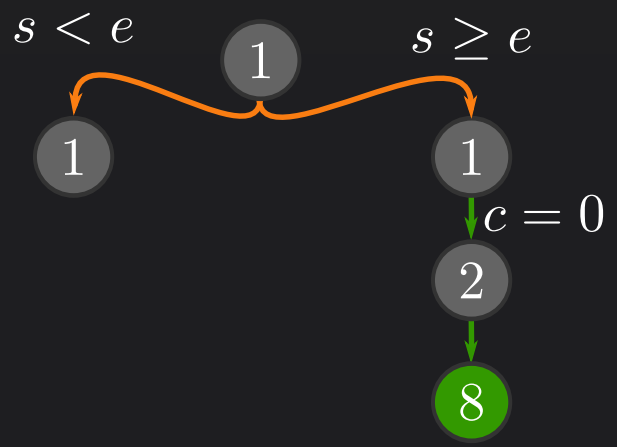
Code

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



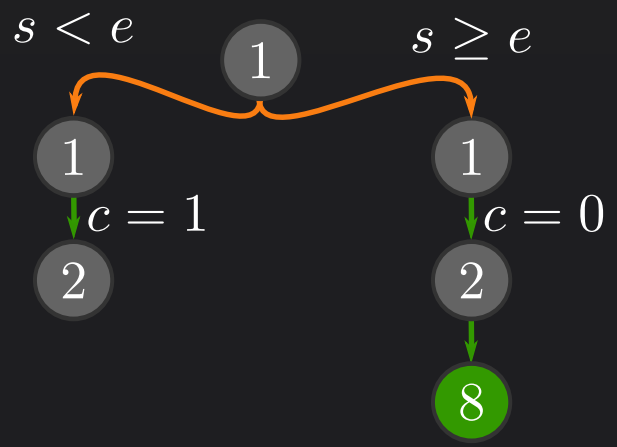
Code

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



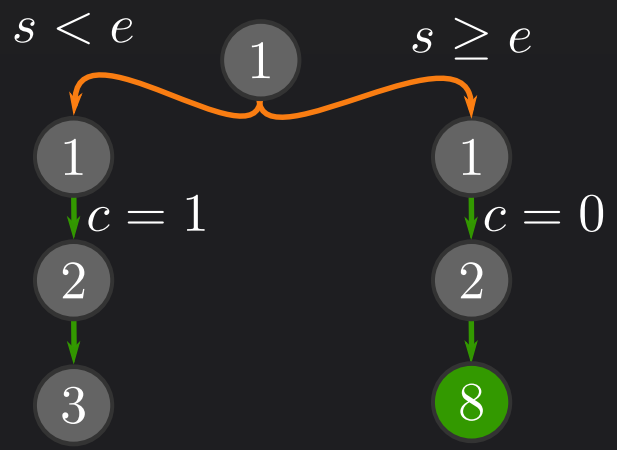
Code

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



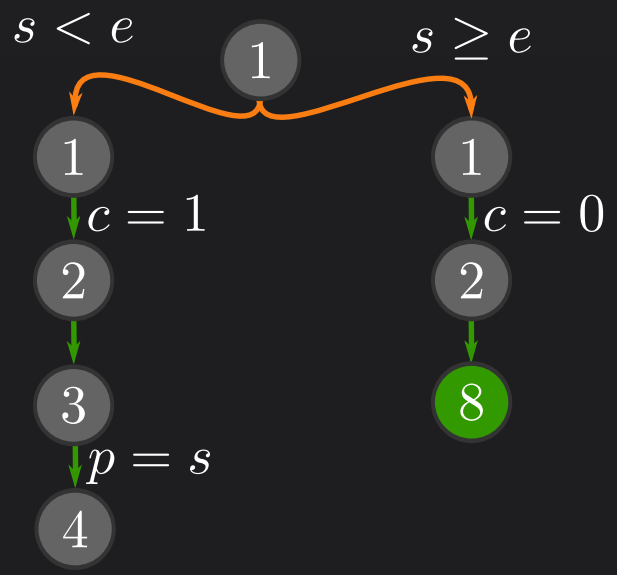
Code

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



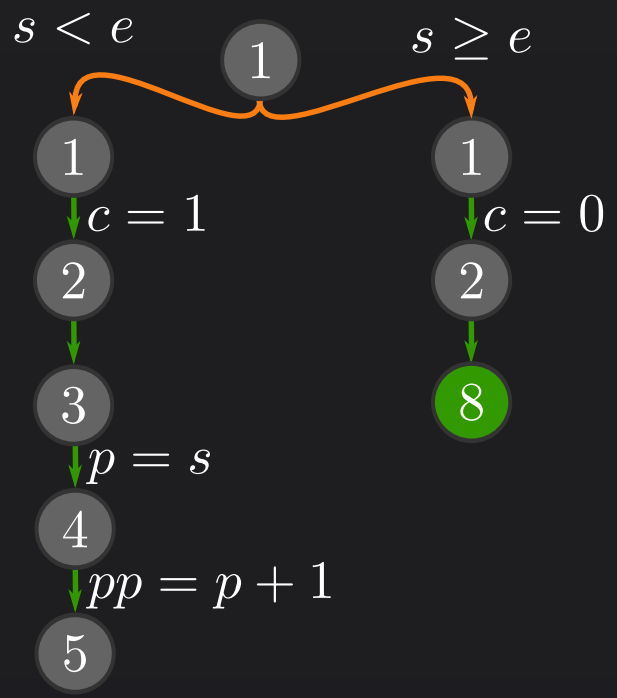
Code

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



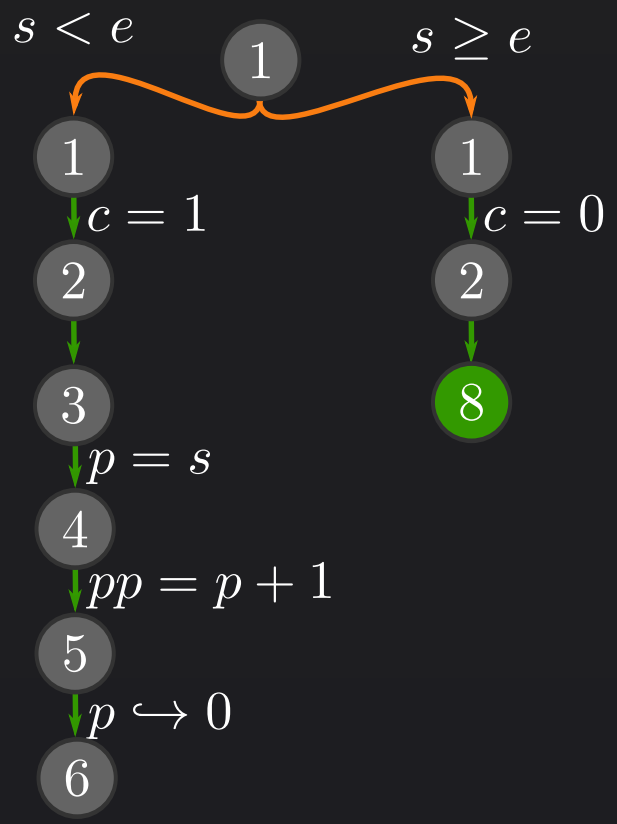
Code

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



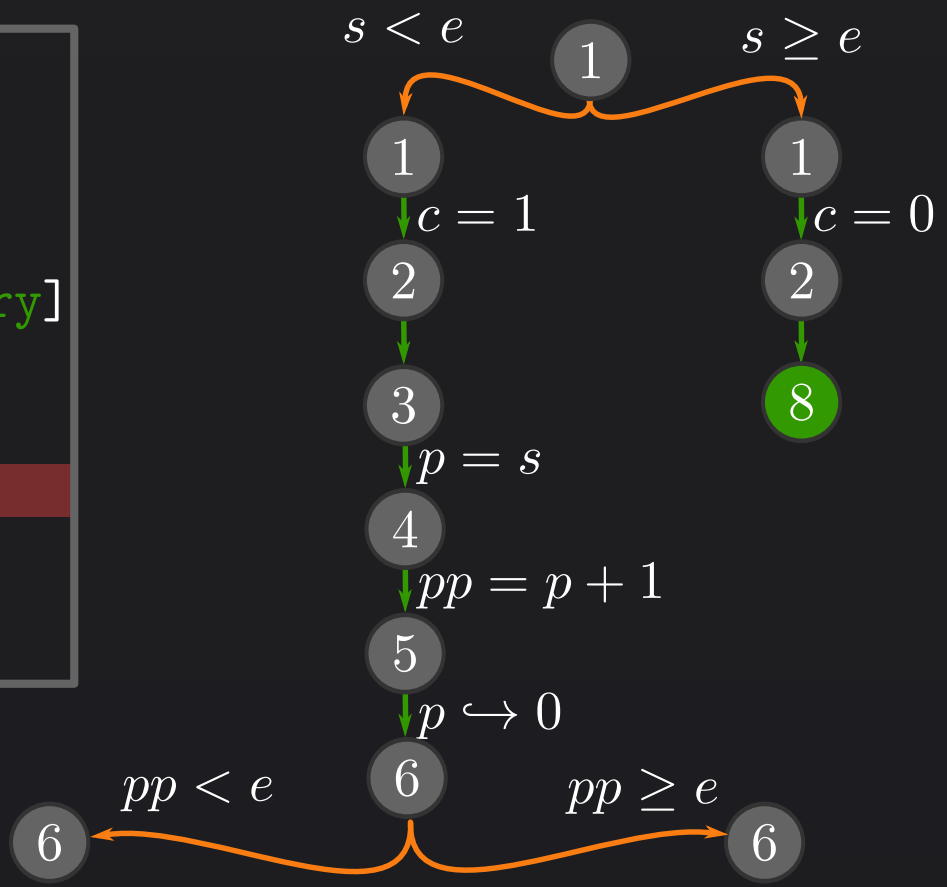
Code

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



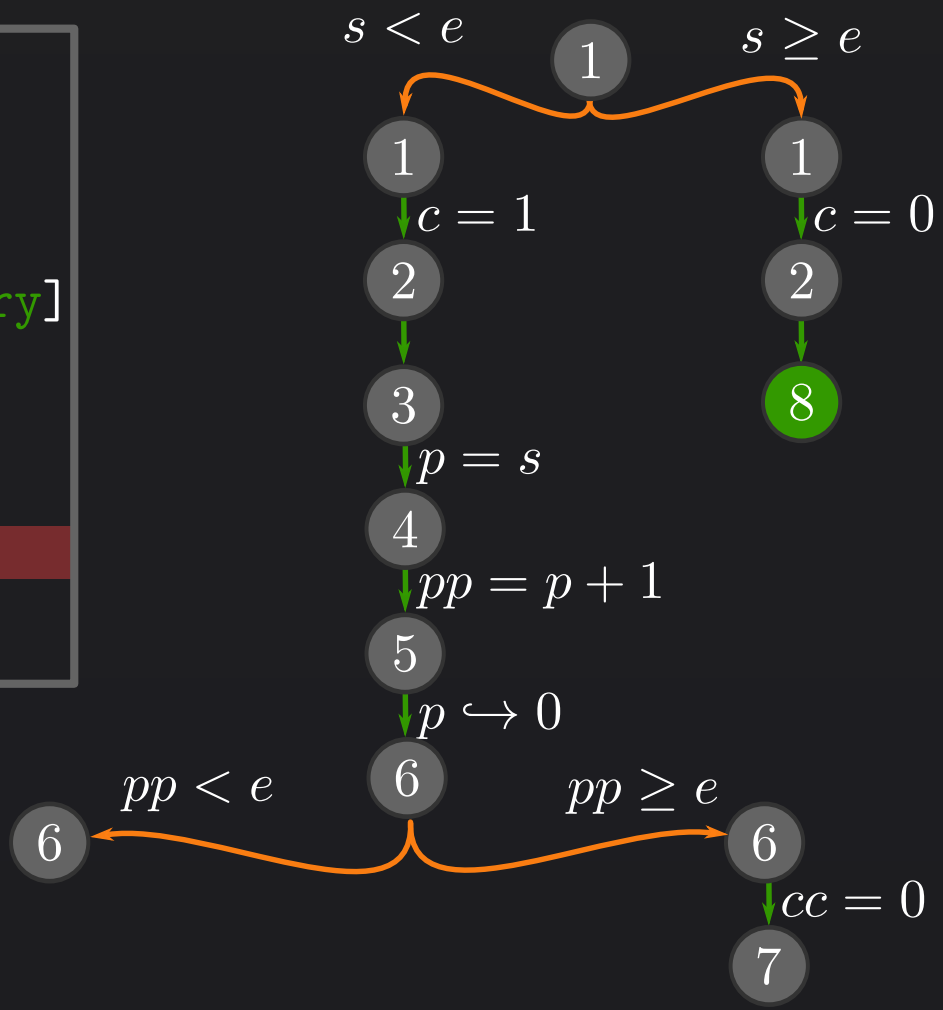
Code

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



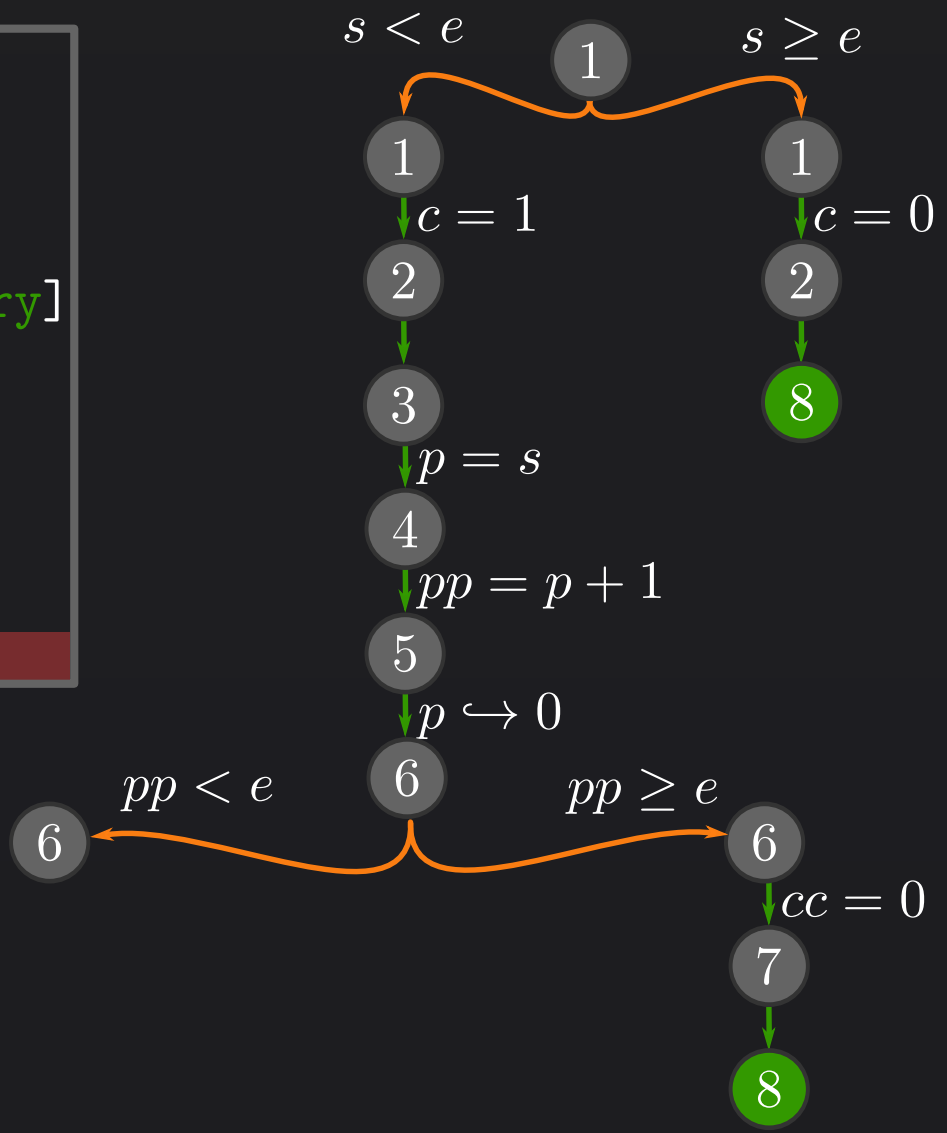
Code

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



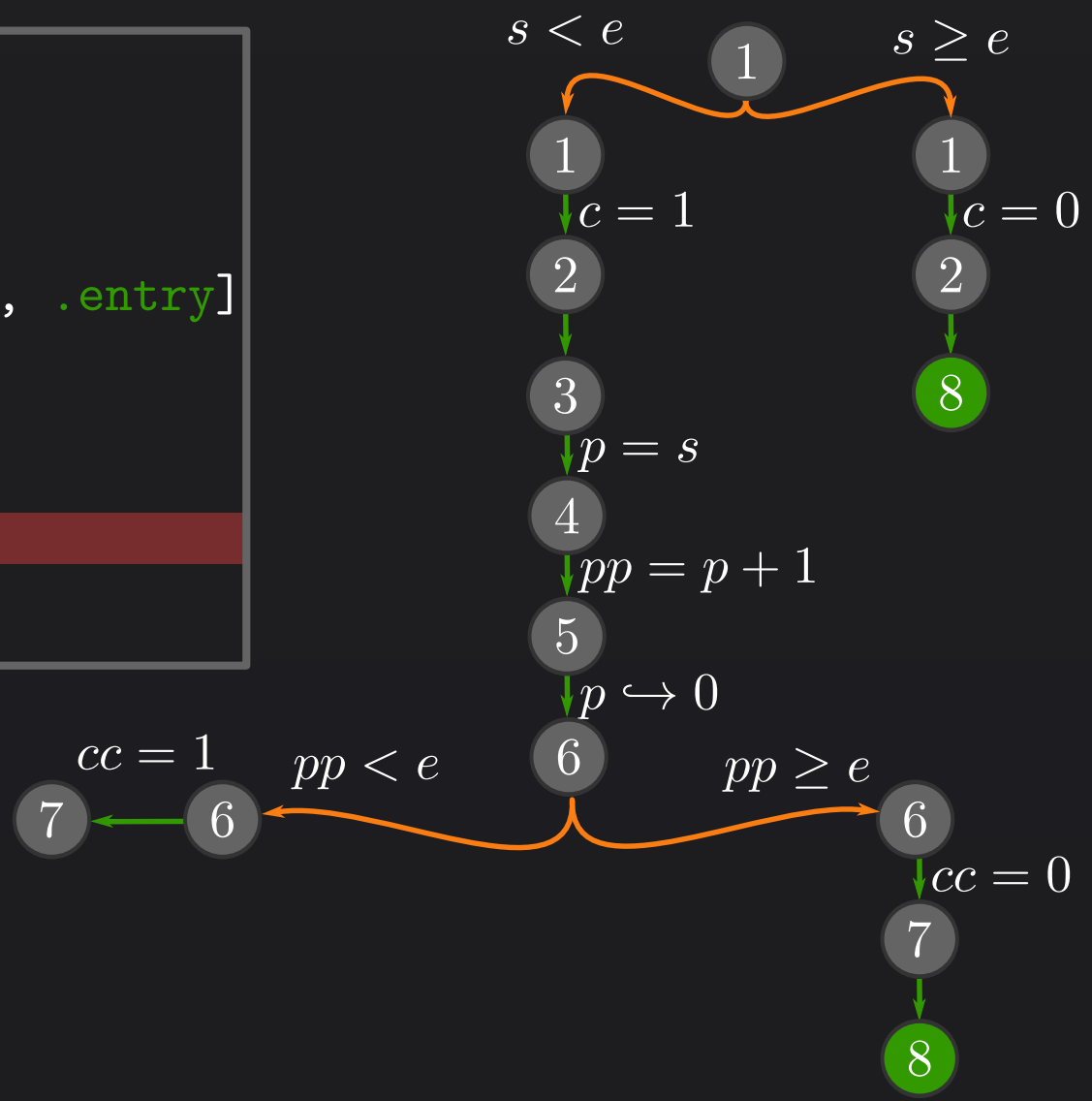
Code

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



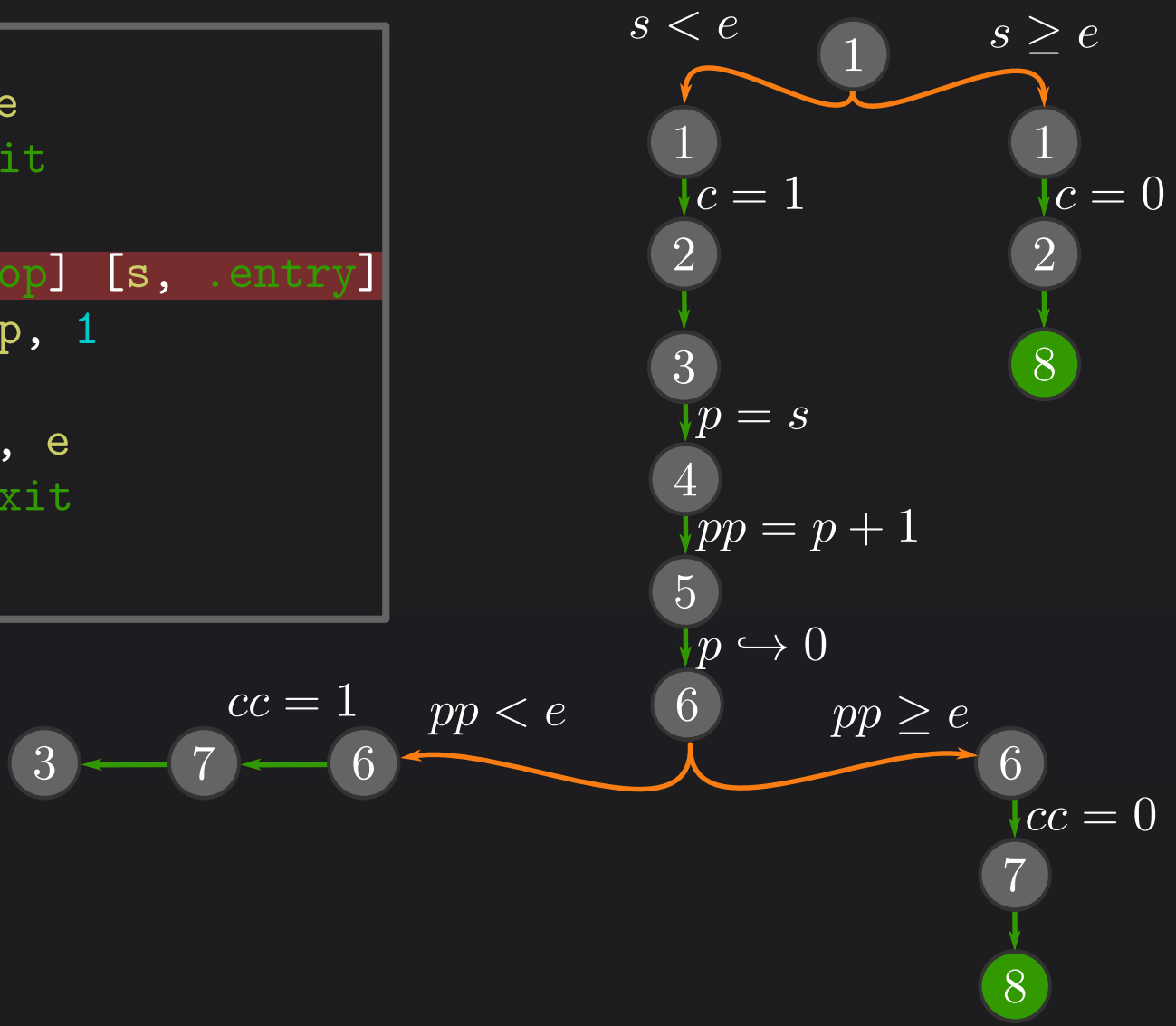
Code

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



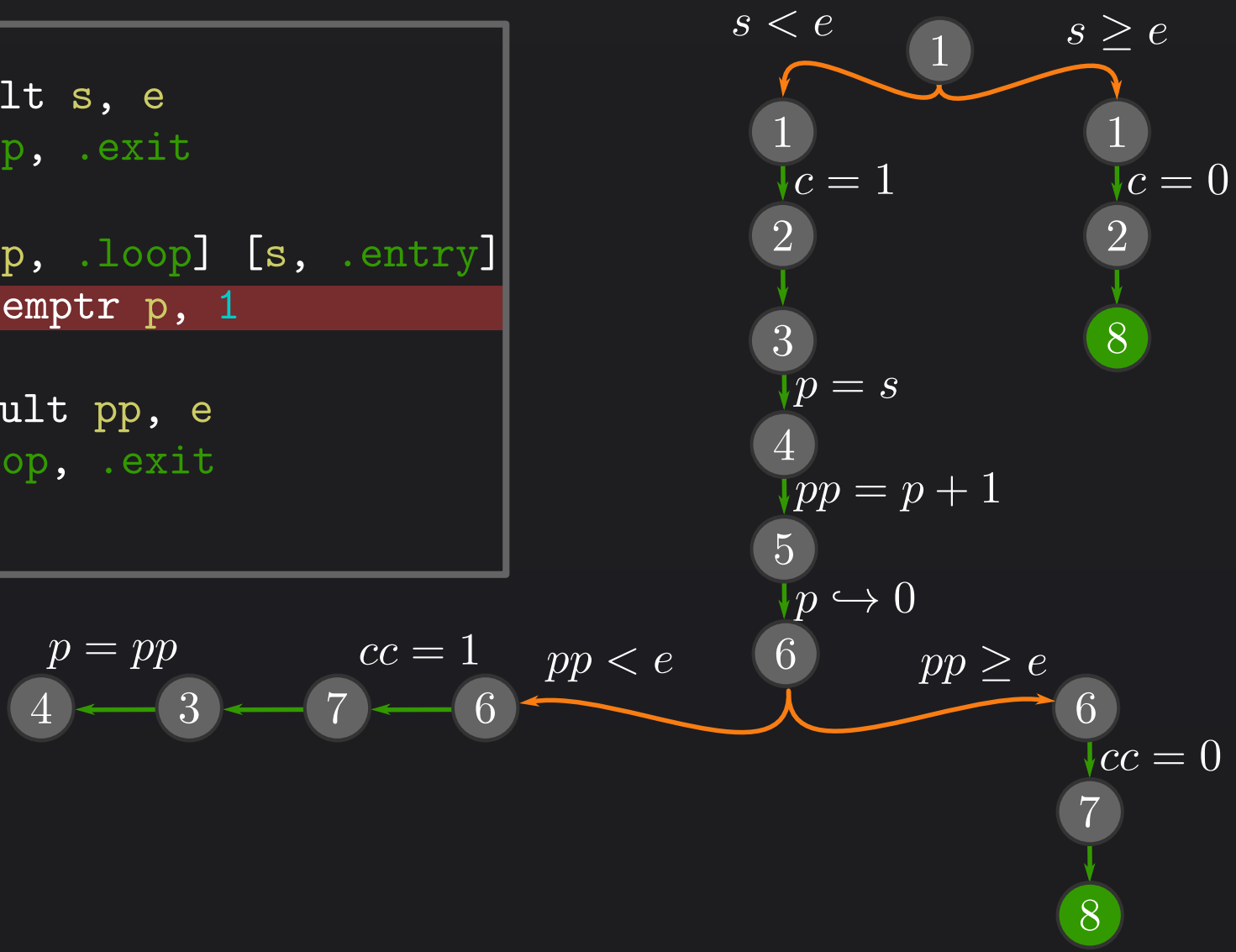
Code

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



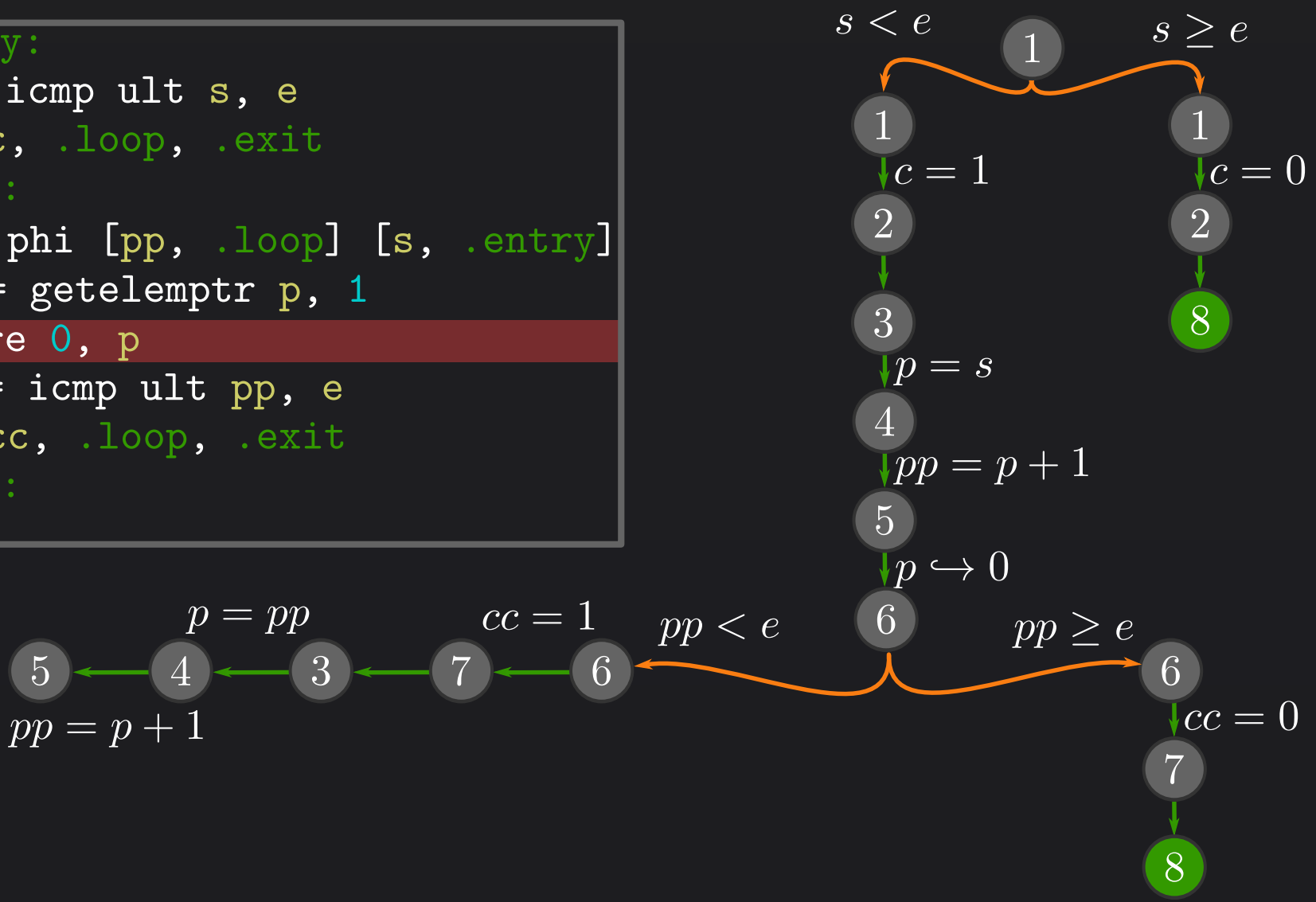
Code

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



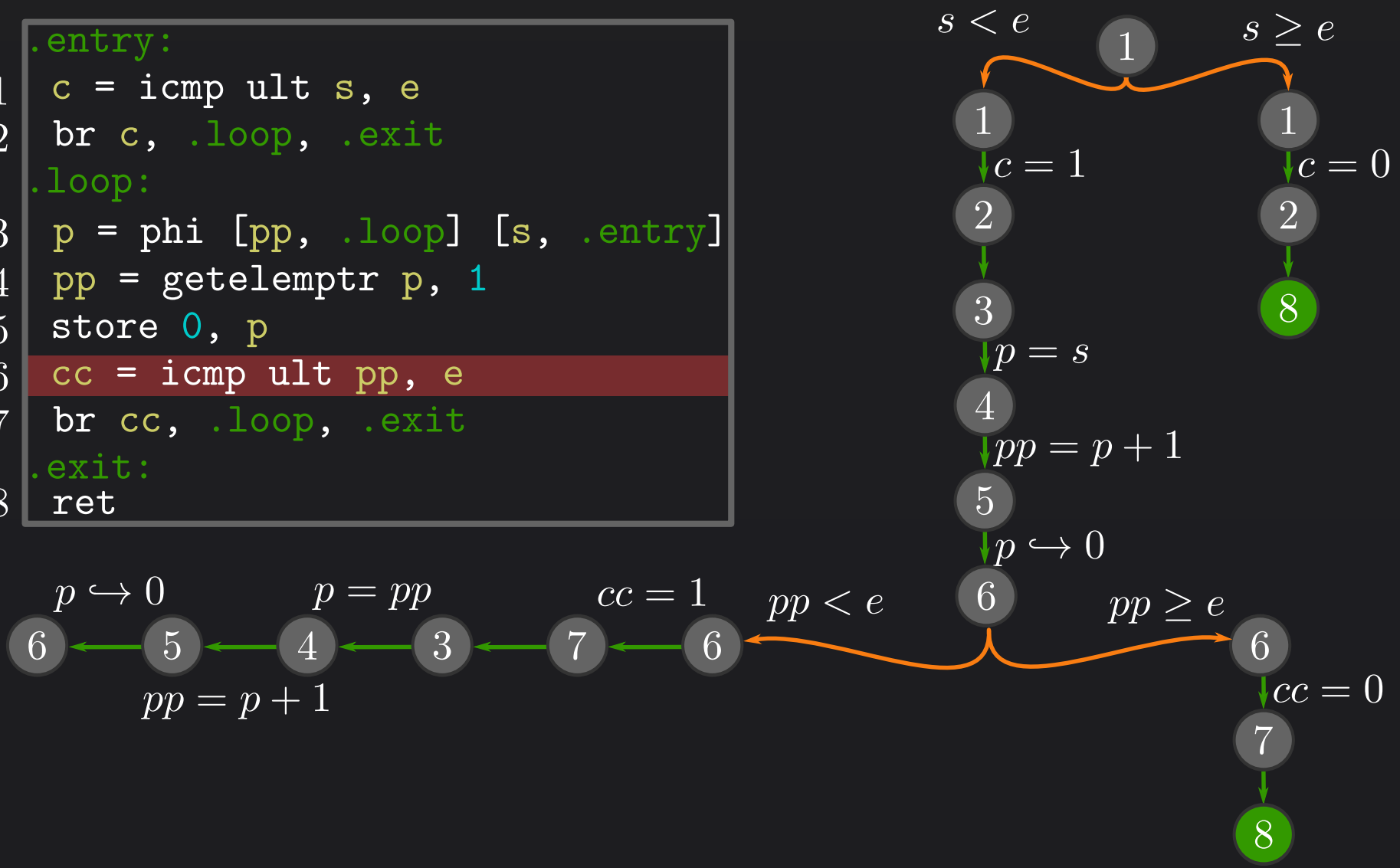
Code

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



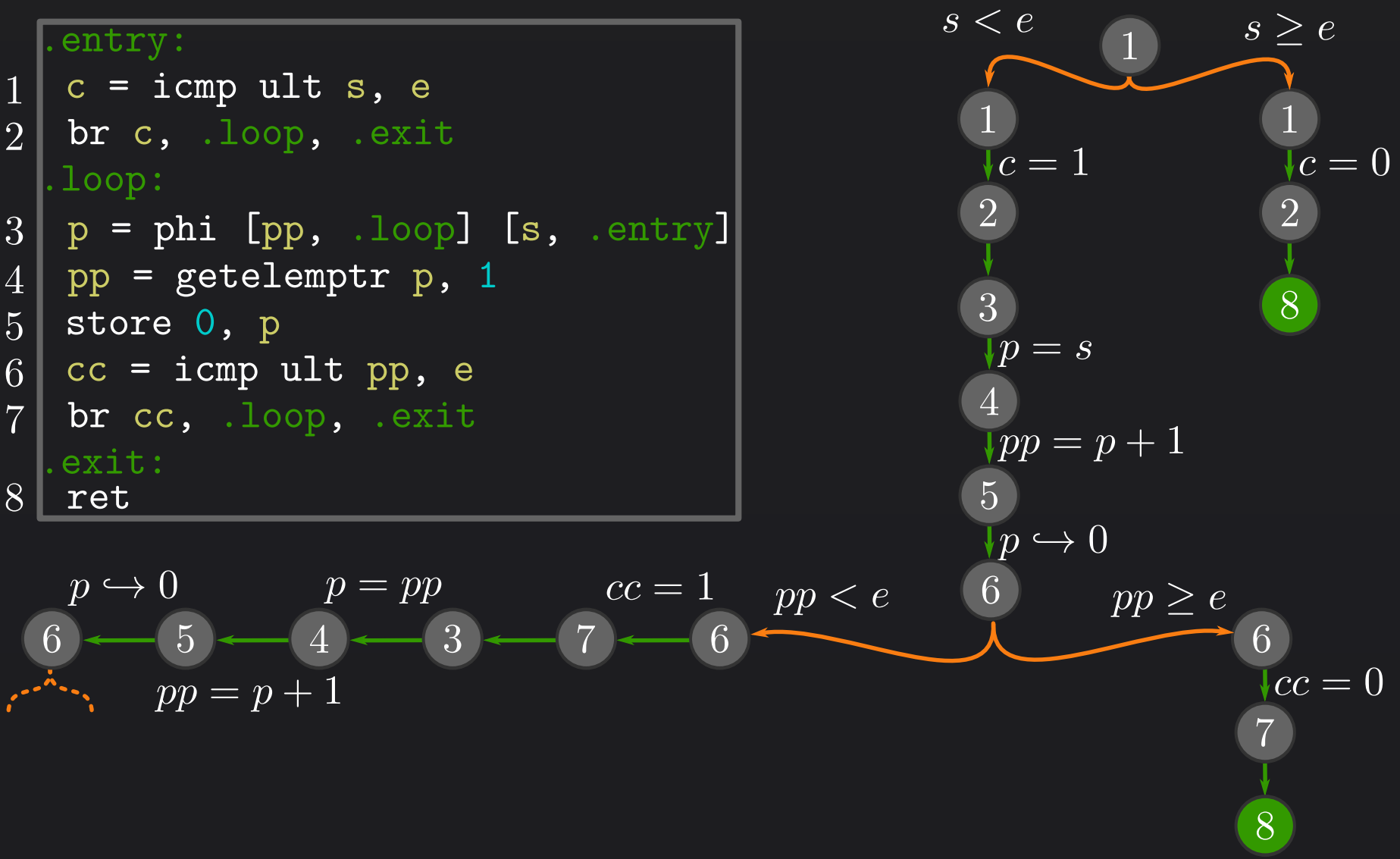
Code

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



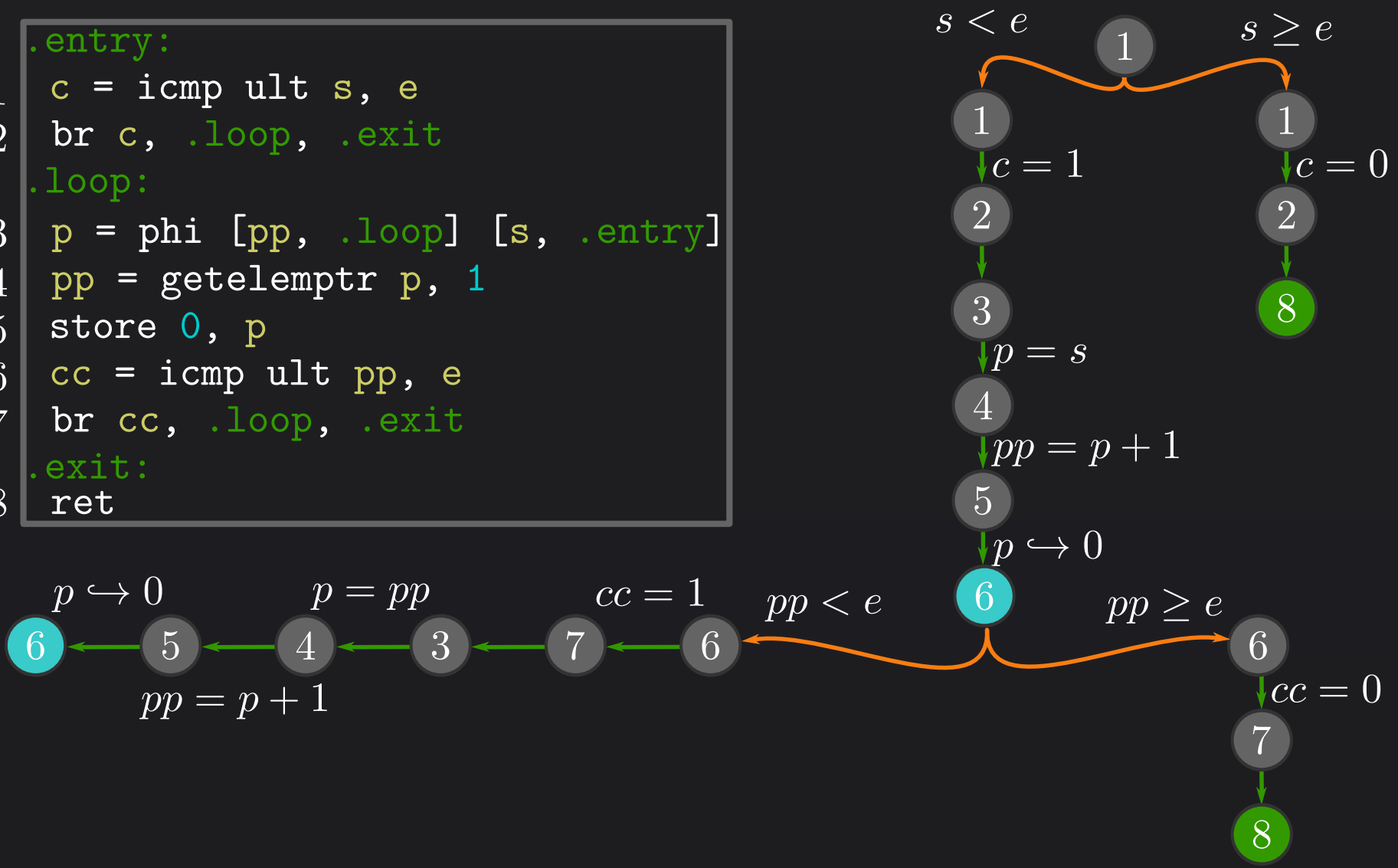
Code

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



Code

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



Merge

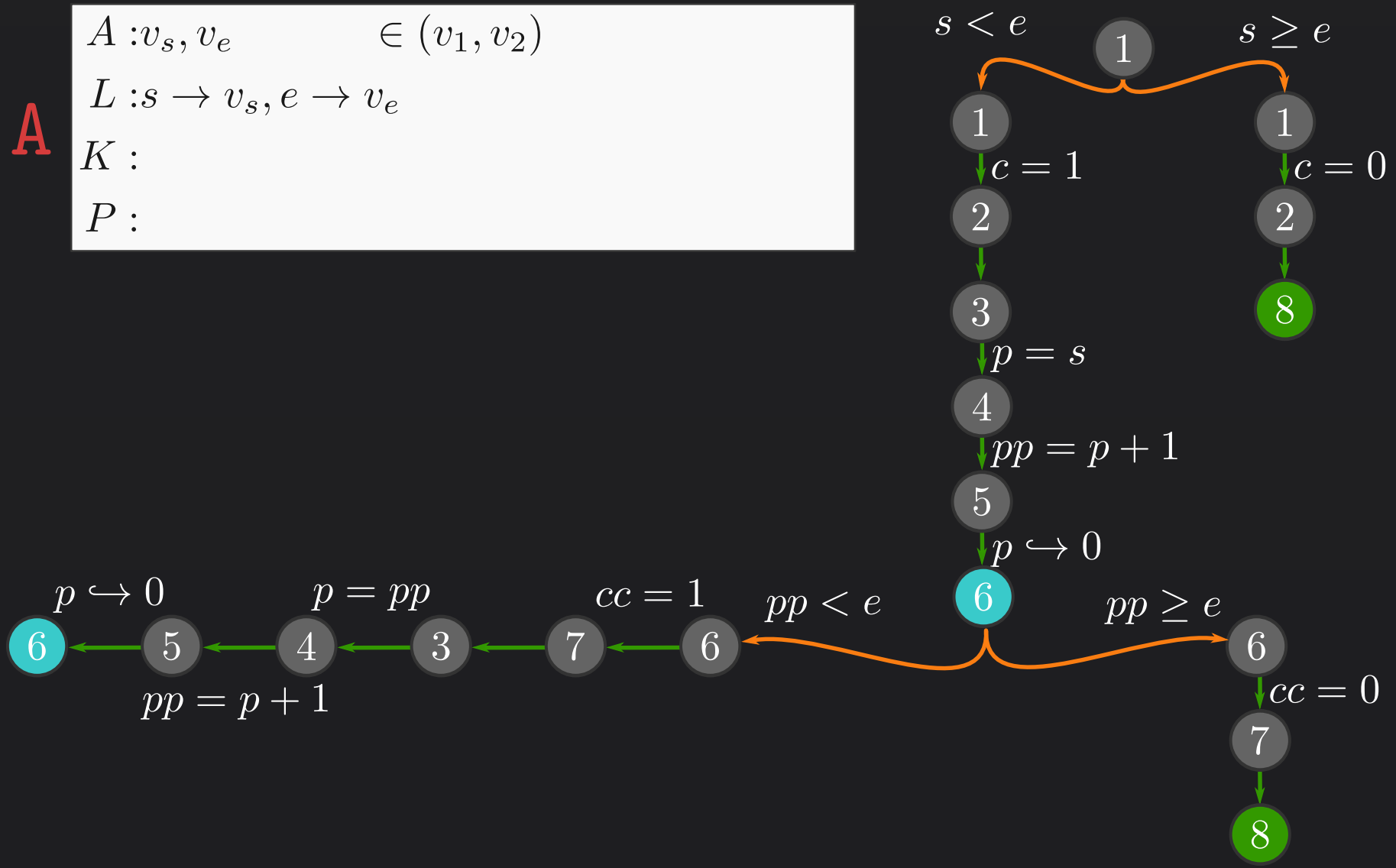
A

$A : v_s, v_e \in (v_1, v_2)$

$L : s \to v_s, e \to v_e$

$K :$

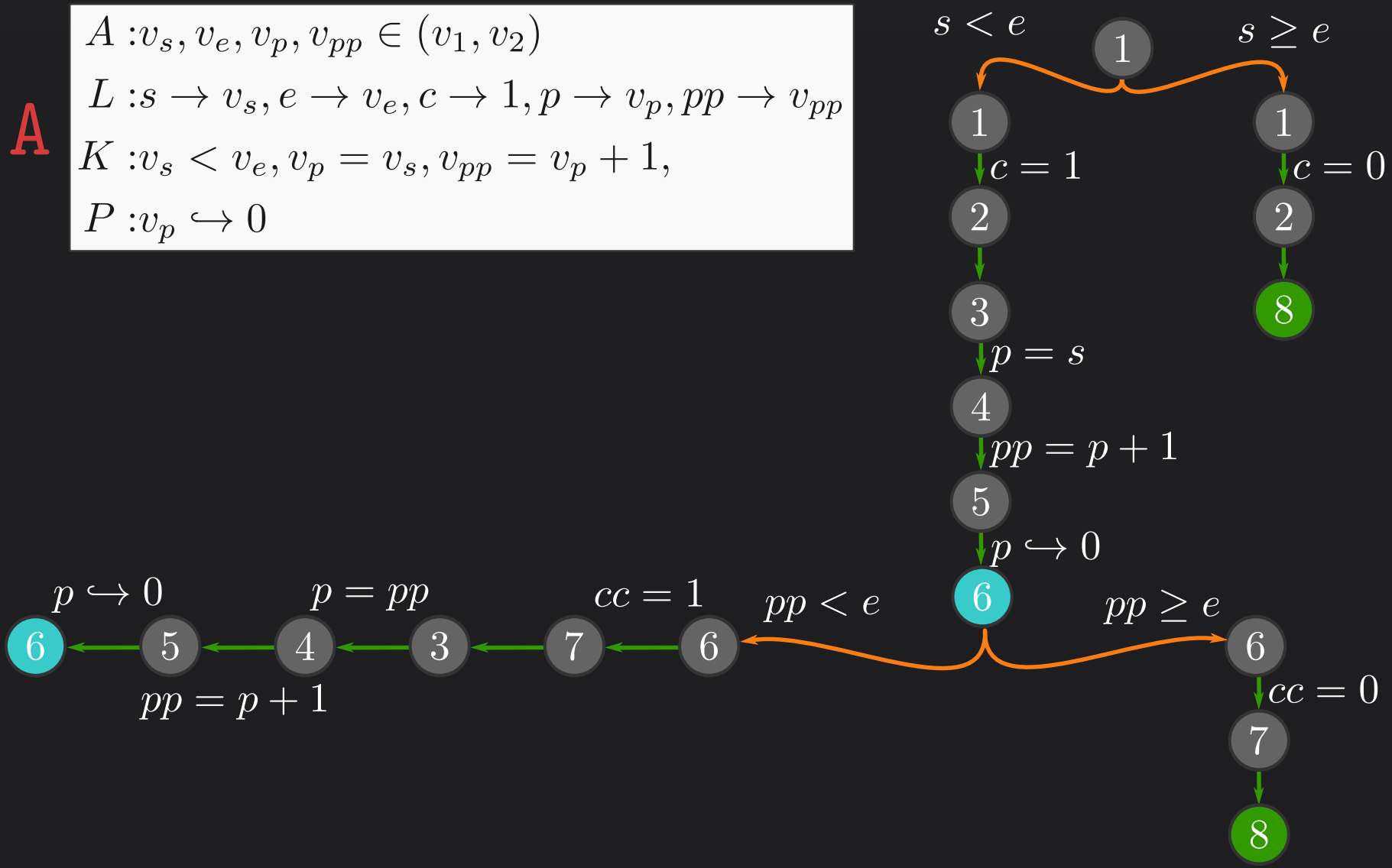
$P :$



Merge

A

$$A : v_s, v_e, v_p, v_{pp} \in (v_1, v_2)$$
$$L : s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp}$$
$$K : v_s < v_e, v_p = v_s, v_{pp} = v_p + 1,$$
$$P : v_p \hookrightarrow 0$$



Merge

A

$A : v_s, v_e, v_p, v_{pp} \in (v_1, v_2)$

$L : s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp}$

$K : v_s < v_e, v_p = v_s, v_{pp} = v_p + 1,$

$P : v_p \hookrightarrow 0$

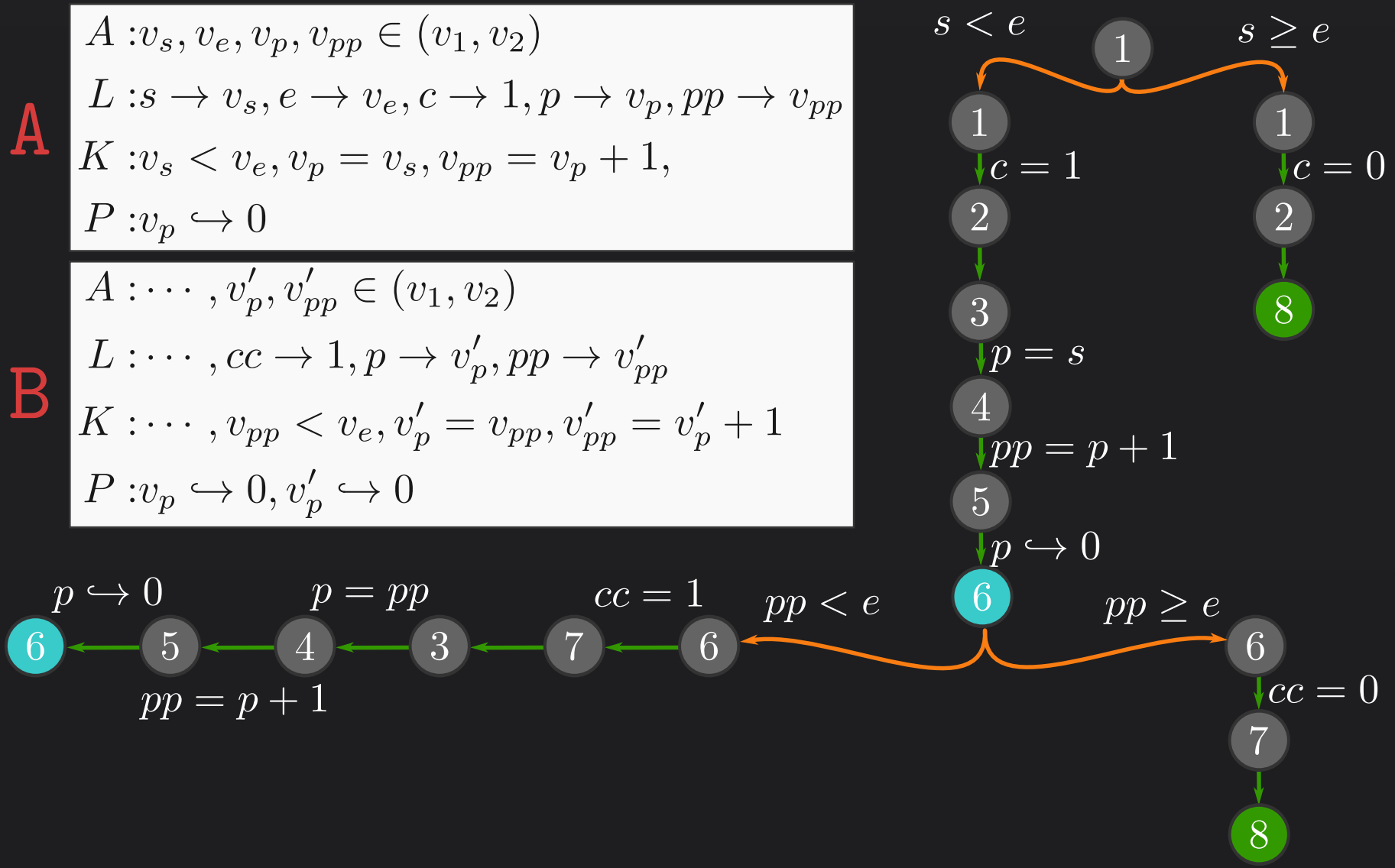
B

$A : \cdots, v'_p, v'_{pp} \in (v_1, v_2)$

$L : \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp}$

$K : \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1$

$P : v_p \hookrightarrow 0, v'_p \hookrightarrow 0$



Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

$$
\begin{aligned}
&(v_1, v_1), (v_2, v_2), \\
&(v_e, v_e), (v_s, v_s),
\end{aligned}
$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

C

$$
\begin{aligned}
A &: v_s, v_e \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1 \\
K &: \\
P &:
\end{aligned}
$$

$$
\begin{aligned}
&(v_1, v_1), (v_2, v_2), \\
&(v_e, v_e), (v_s, v_s),
\end{aligned}
$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

C

$$
\begin{aligned}
A &: v_s, v_e \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1 \\
K &: \\
P &:
\end{aligned}
$$

$$
\begin{aligned}
&(v_1, v_1), (v_2, v_2), \\
&(v_e, v_e), (v_s, v_s), \\
&(v_p, v'_p), (v_{pp}, v'_{pp})
\end{aligned}
$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

C

$$
\begin{aligned}
A &: v_s, v_e, c_p, c_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to c_p, pp \to c_{pp} \\
K &: \\
P &: c_p \hookrightarrow 0
\end{aligned}
$$

$$
\begin{aligned}
&(v_1, v_1), (v_2, v_2), \\
&(v_e, v_e), (v_s, v_s), \\
&(v_p, v'_p), (v_{pp}, v'_{pp})
\end{aligned}
$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

C

$$
\begin{aligned}
A &: v_s, v_e, c_p, c_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to c_p, pp \to c_{pp} \\
K &: \\
P &: c_p \hookrightarrow 0
\end{aligned}
$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

C

$$
\begin{aligned}
A &: v_s, v_e, c_p, c_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to c_p, pp \to c_{pp} \\
K &: \\
P &: c_p \hookrightarrow 0
\end{aligned}
$$

$$v_{pp} = v_p + 1$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

C

$$
\begin{aligned}
A &: v_s, v_e, c_p, c_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to c_p, pp \to c_{pp} \\
K &: \\
P &: c_p \hookrightarrow 0
\end{aligned}
$$

$$c_{pp} = c_p + 1$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

$$c_{pp} = c_p + 1$$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

$$v'_{pp} = v'_p + 1$$

C

$$
\begin{aligned}
A &: v_s, v_e, c_p, c_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to c_p, pp \to c_{pp} \\
K &: \\
P &: c_p \hookrightarrow 0
\end{aligned}
$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

$$c_{pp} = c_p + 1$$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

$$c_{pp} = c_p + 1$$

C

$$
\begin{aligned}
A &: v_s, v_e, c_p, c_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to c_p, pp \to c_{pp} \\
K &: \\
P &: c_p \hookrightarrow 0
\end{aligned}
$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

C

$$
\begin{aligned}
A &: v_s, v_e, c_p, c_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to c_p, pp \to c_{pp} \\
K &: c_{pp} = c_p + 1 \\
P &: c_p \hookrightarrow 0
\end{aligned}
$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

C

$$
\begin{aligned}
A &: v_s, v_e, c_p, c_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to c_p, pp \to c_{pp} \\
K &: c_{pp} = c_p + 1 \\
P &: c_p \hookrightarrow 0
\end{aligned}
$$

$$v_p = v_s$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

$$v_p \geq v_s$$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

C

$$
\begin{aligned}
A &: v_s, v_e, c_p, c_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to c_p, pp \to c_{pp} \\
K &: c_{pp} = c_p + 1 \\
P &: c_p \hookrightarrow 0
\end{aligned}
$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

$$c_p \geq v_s$$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

C

$$
\begin{aligned}
A &: v_s, v_e, c_p, c_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to c_p, pp \to c_{pp} \\
K &: c_{pp} = c_p + 1 \\
P &: c_p \hookrightarrow 0
\end{aligned}
$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

$$c_p \geq v_s$$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

$$v'_p = v_{pp}$$

C

$$
\begin{aligned}
A &: v_s, v_e, c_p, c_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to c_p, pp \to c_{pp} \\
K &: c_{pp} = c_p + 1 \\
P &: c_p \hookrightarrow 0
\end{aligned}
$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

$$c_p \geq v_s$$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

$$v'_p = v_{pp} = v_p + 1$$

C

$$
\begin{aligned}
A &: v_s, v_e, c_p, c_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to c_p, pp \to c_{pp} \\
K &: c_{pp} = c_p + 1 \\
P &: c_p \hookrightarrow 0
\end{aligned}
$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

$$c_p \geq v_s$$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

$$v'_p = v_p + 1 \geq v_s$$

C

$$
\begin{aligned}
A &: v_s, v_e, c_p, c_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to c_p, pp \to c_{pp} \\
K &: c_{pp} = c_p + 1 \\
P &: c_p \hookrightarrow 0
\end{aligned}
$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

$c_p \geq v_s$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

$c_p \geq v_s$

C

$$
\begin{aligned}
A &: v_s, v_e, c_p, c_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to c_p, pp \to c_{pp} \\
K &: c_{pp} = c_p + 1 \\
P &: c_p \hookrightarrow 0
\end{aligned}
$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

B

$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

C

$$
\begin{aligned}
A &: v_s, v_e, c_p, c_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to c_p, pp \to c_{pp} \\
K &: c_{pp} = c_p + 1, c_p \geq v_s \\
P &: c_p \hookrightarrow 0
\end{aligned}
$$

Merge

A

$$
\begin{aligned}
A &: v_s, v_e, v_p, v_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to v_p, pp \to v_{pp} \\
K &: v_s < v_e, v_p = v_s, v_{pp} = v_p + 1, \\
P &: v_p \hookrightarrow 0
\end{aligned}
$$

B

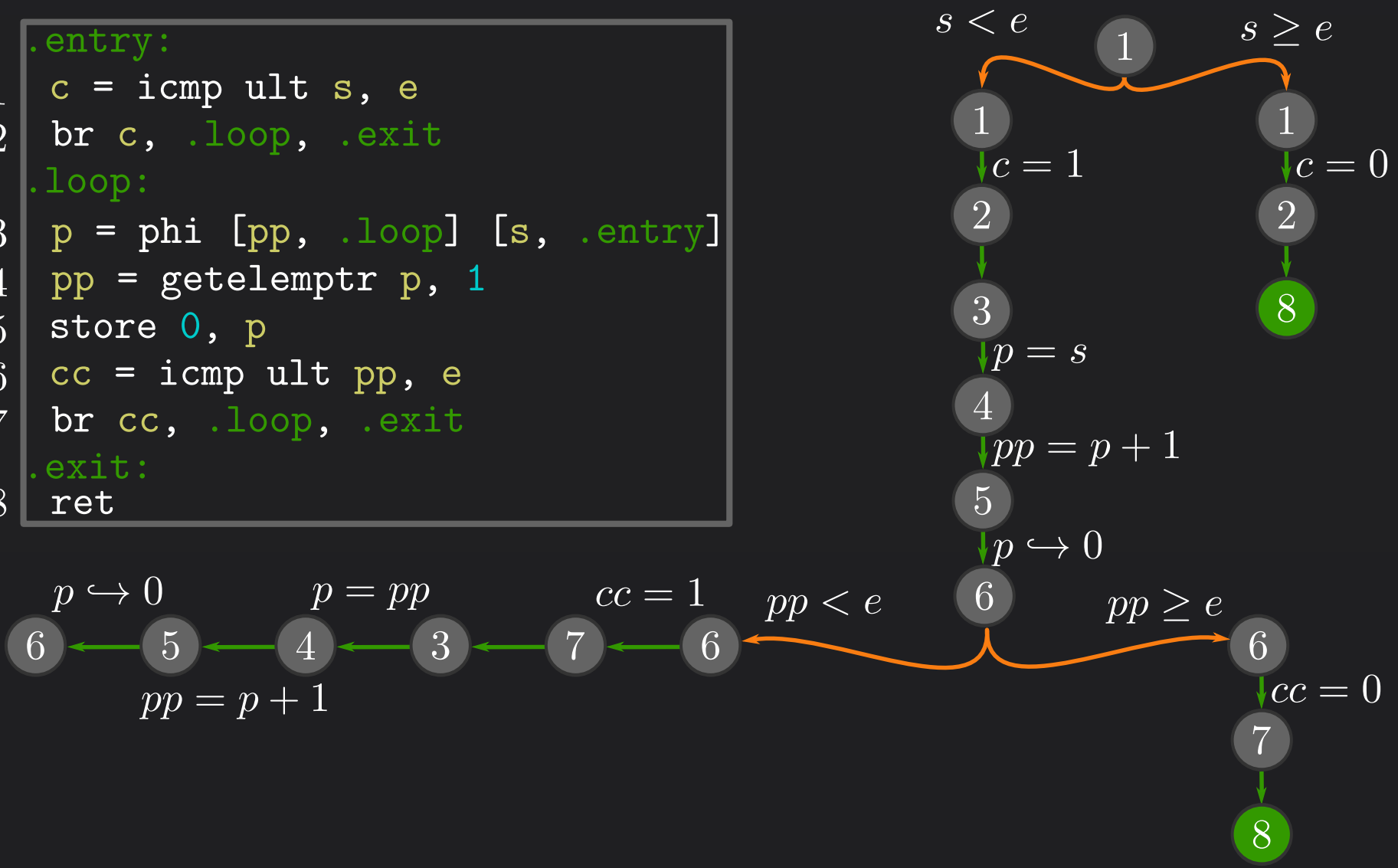
$$
\begin{aligned}
A &: \cdots, v'_p, v'_{pp} \in (v_1, v_2) \\
L &: \cdots, cc \to 1, p \to v'_p, pp \to v'_{pp} \\
K &: \cdots, v_{pp} < v_e, v'_p = v_{pp}, v'_{pp} = v'_p + 1 \\
P &: v_p \hookrightarrow 0, v'_p \hookrightarrow 0
\end{aligned}
$$

C

$$
\begin{aligned}
A &: v_s, v_e, c_p, c_{pp} \in (v_1, v_2) \\
L &: s \to v_s, e \to v_e, c \to 1, p \to c_p, pp \to c_{pp} \\
K &: c_{pp} = c_p + 1, c_p \geq v_s, c_p < v_e \\
P &: c_p \hookrightarrow 0
\end{aligned}
$$

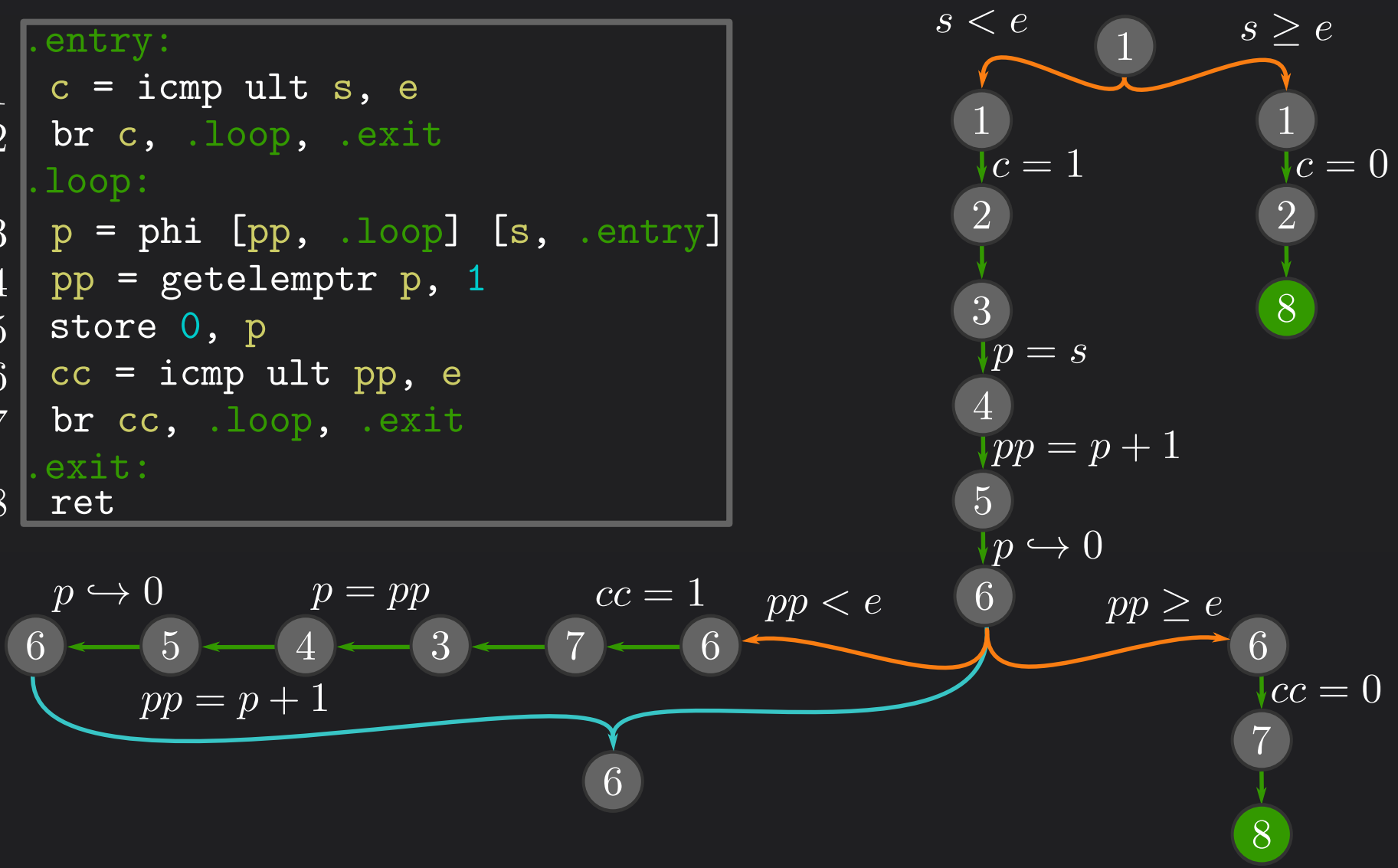
Merge

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



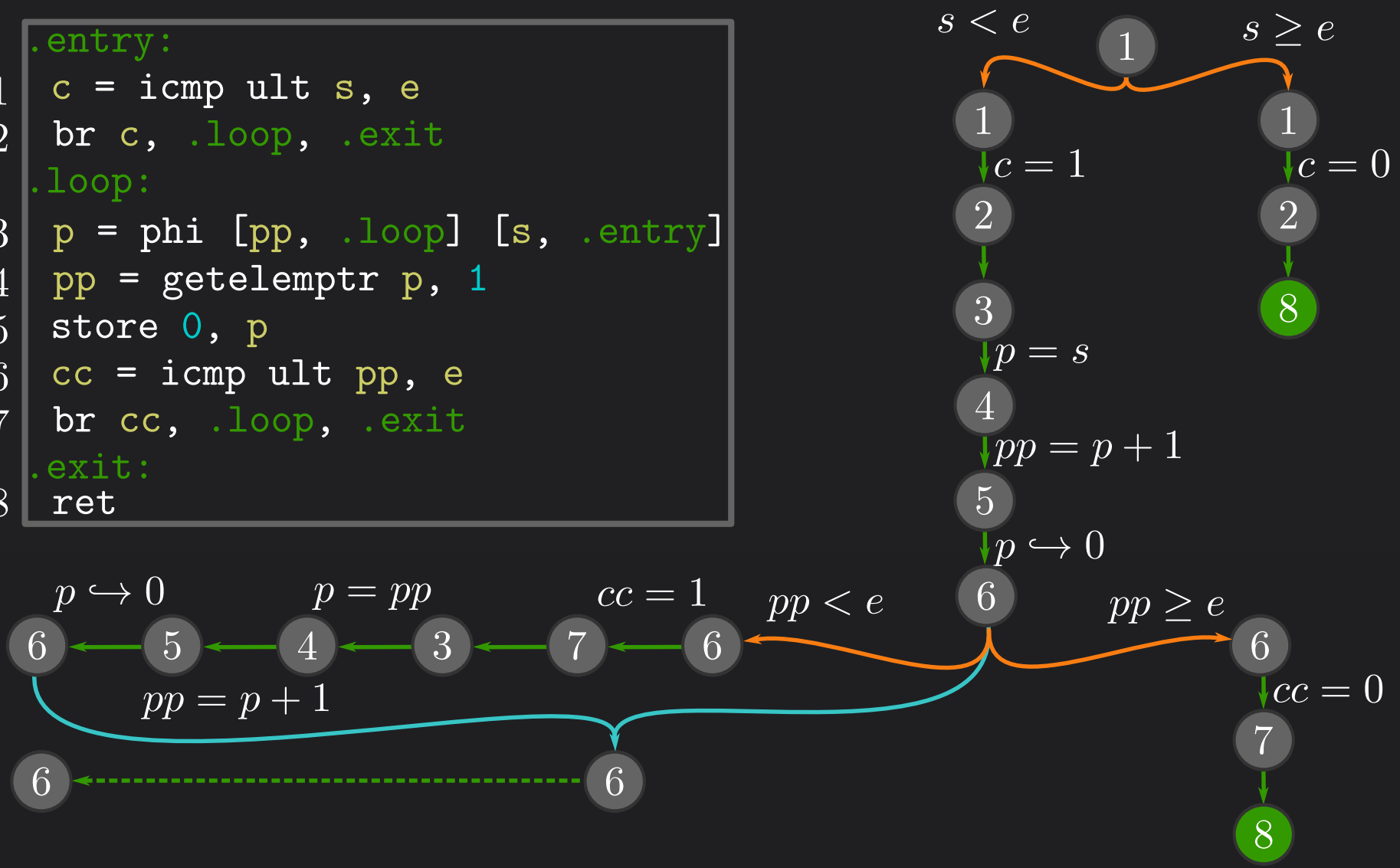
Merge

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



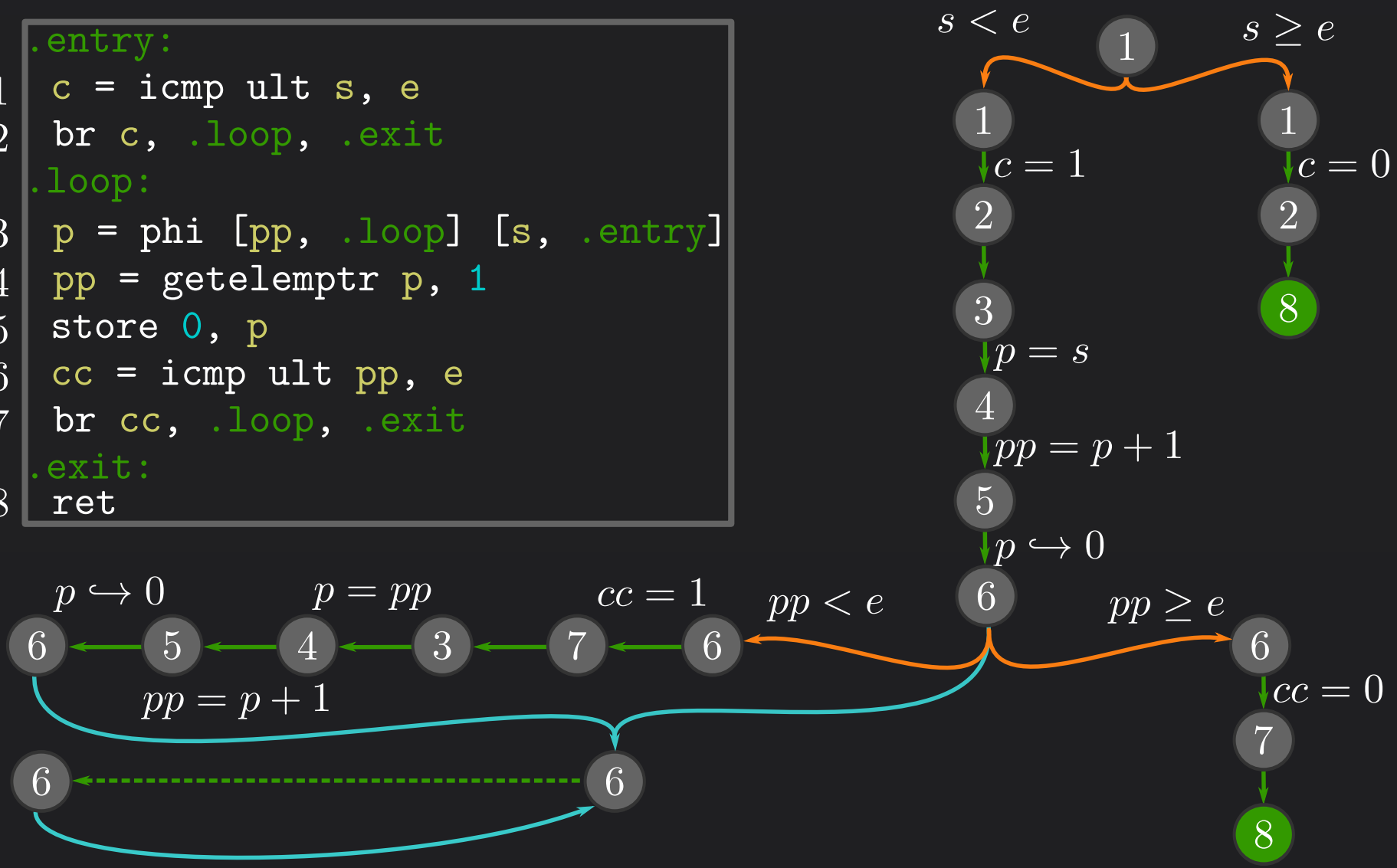
Merge

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



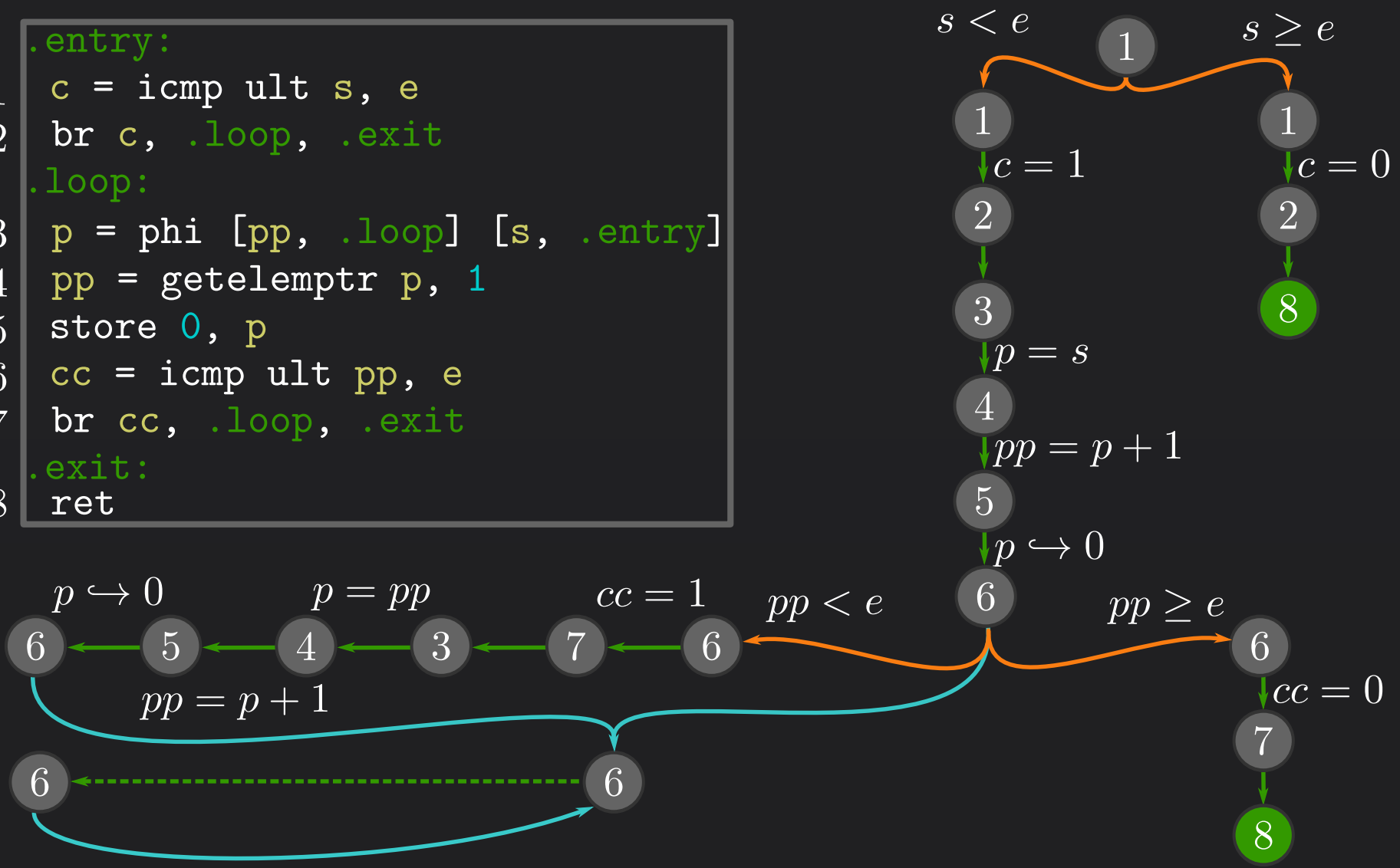
Merge

```
   .entry:
1   c = icmp ult s, e
2   br c, .loop, .exit
   .loop:
3   p = phi [pp, .loop] [s, .entry]
4   pp = getelemptr p, 1
5   store 0, p
6   cc = icmp ult pp, e
7   br cc, .loop, .exit
   .exit:
8   ret
```



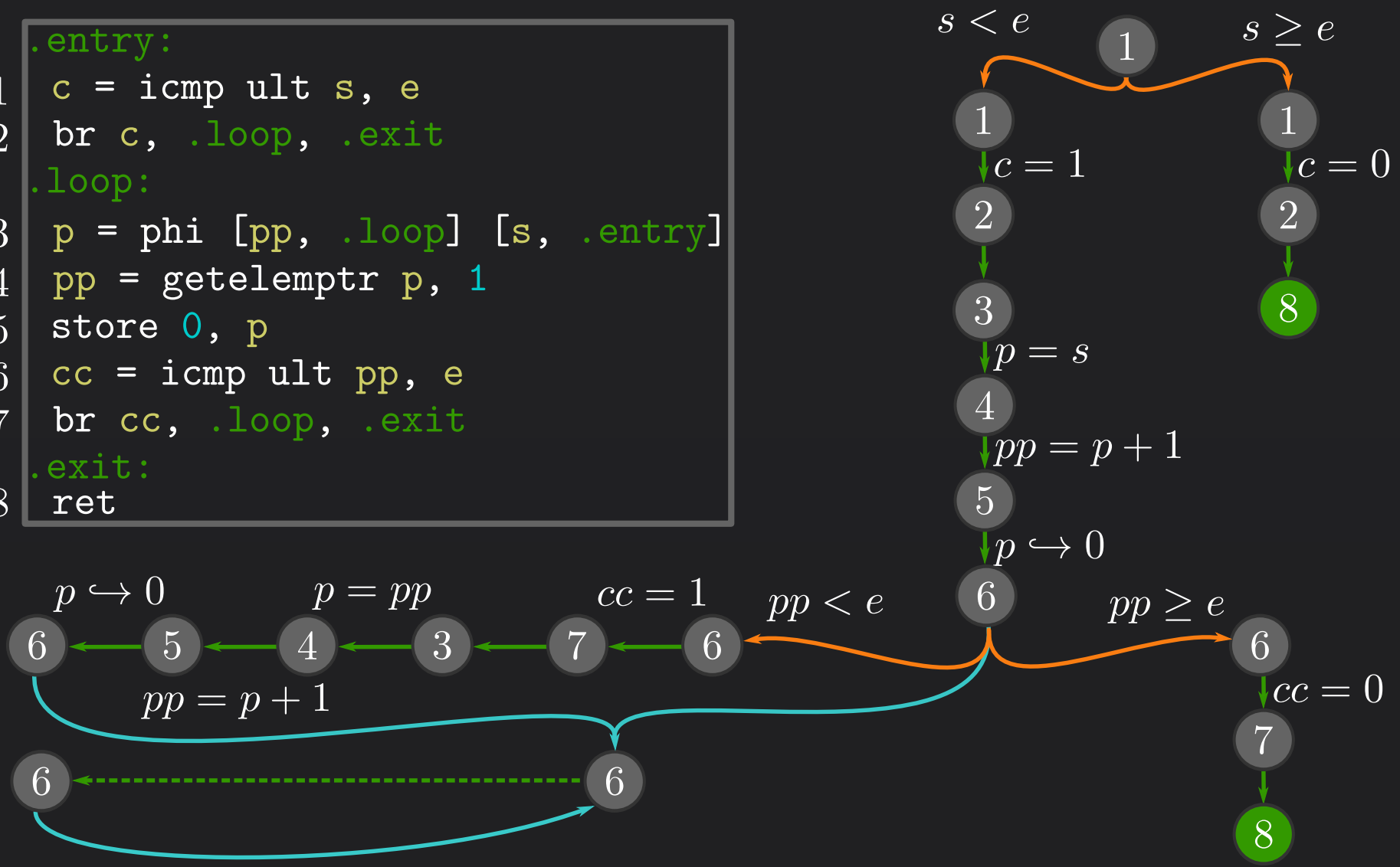
What now?

```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



ITS

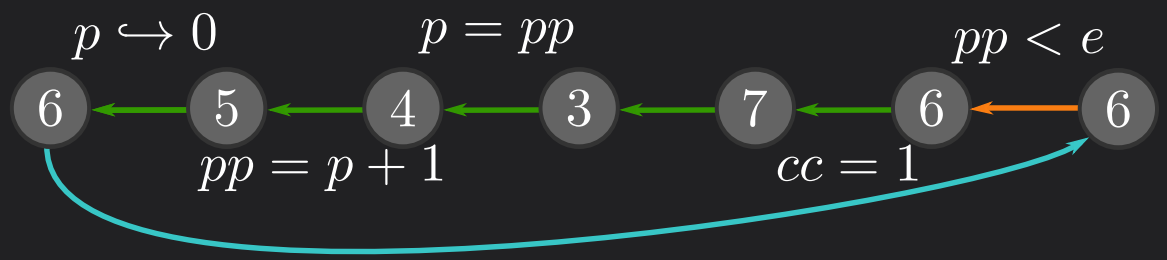
```
.entry:
1  c = icmp ult s, e
2  br c, .loop, .exit
.loop:
3  p = phi [pp, .loop] [s, .entry]
4  pp = getelemptr p, 1
5  store 0, p
6  cc = icmp ult pp, e
7  br cc, .loop, .exit
.exit:
8  ret
```



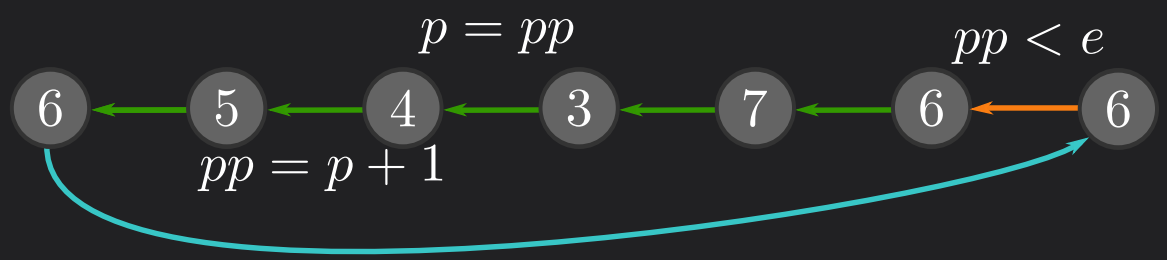
ITS

ITS

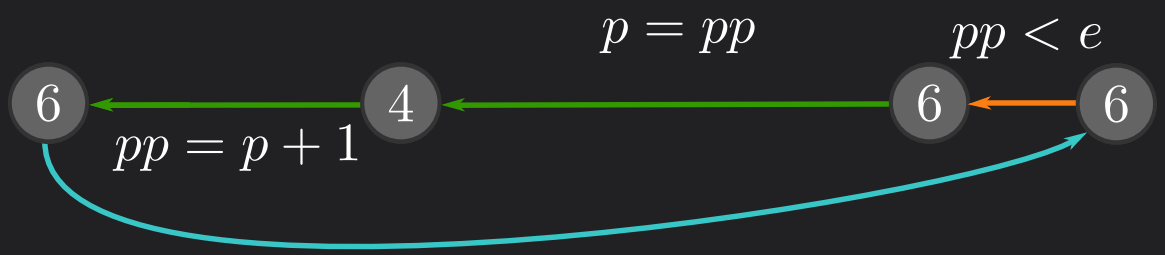
ITS



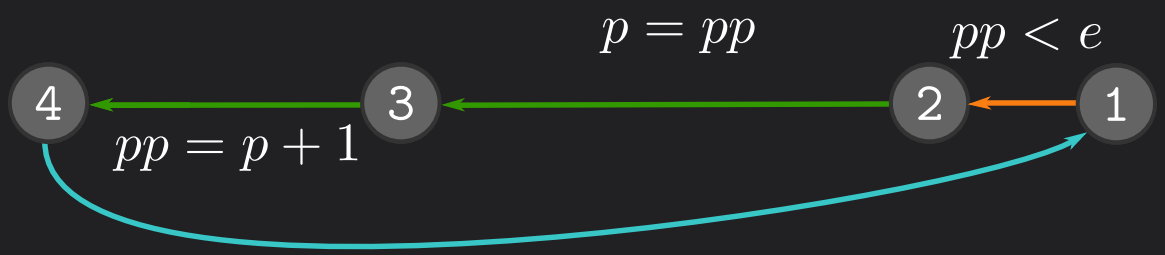
ITS



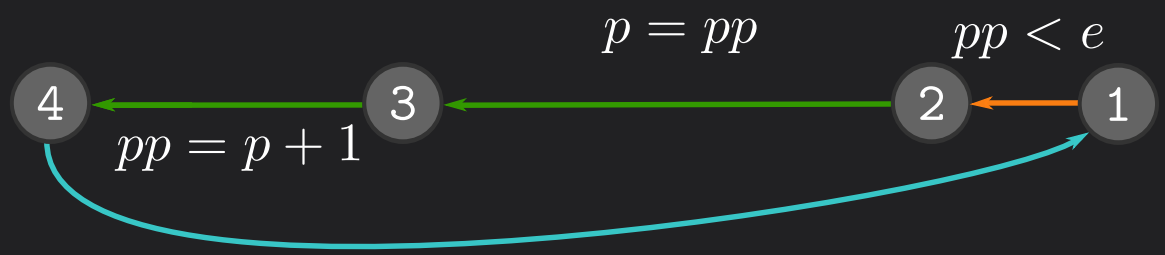
ITS



ITS

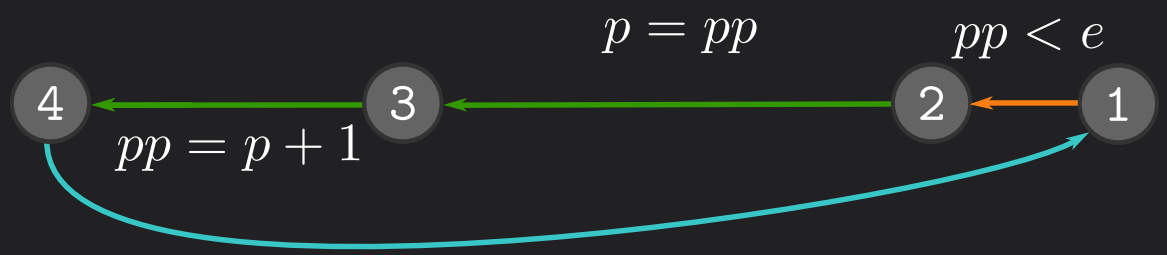


ITS



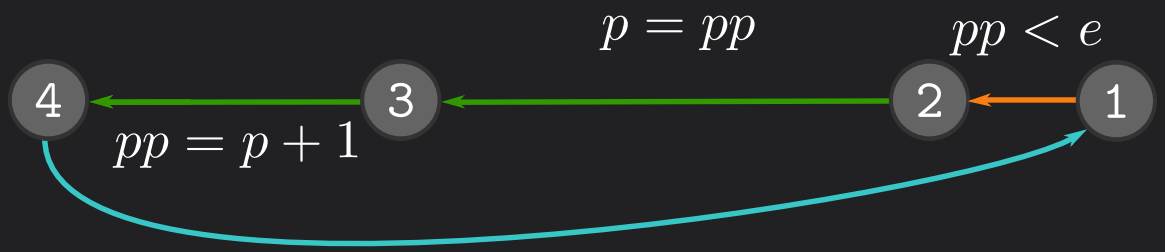
$$f_1(e, p, pp)$$

ITS



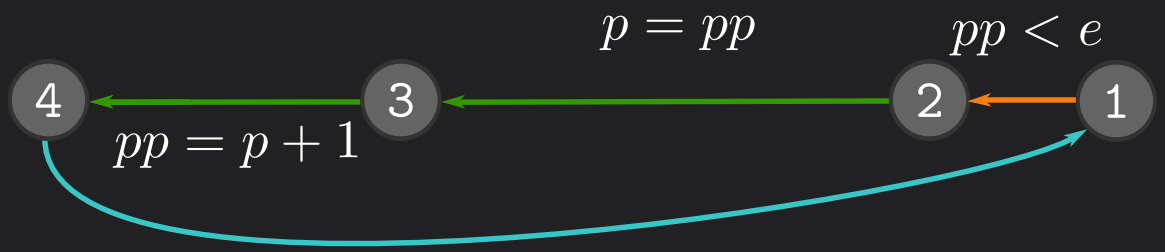
$$f_1(e, p, pp) \rightarrow f_2(e, p, pp) : | : pp < e$$

ITS



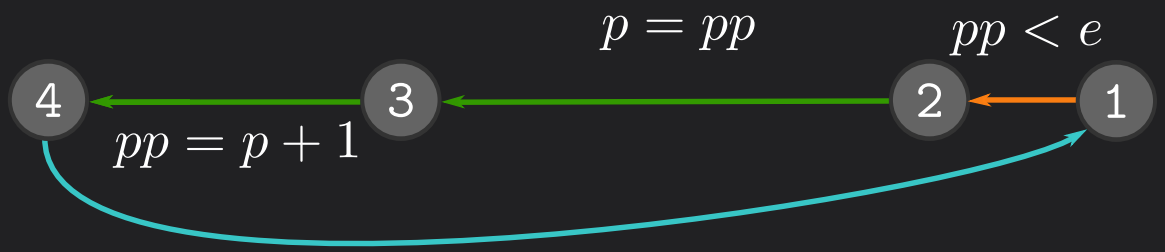
$$f_1(e, p, pp) \to f_2(e, p, pp) : | : pp < e$$
$$f_2(e, p, pp) \to f_3(e, p', pp) : | : p' = pp$$

ITS



$$f_1(e, p, pp) \to f_2(e, p, pp) : | : pp < e$$
$$f_2(e, p, pp) \to f_3(e, p', pp) : | : p' = pp$$
$$f_3(e, p, pp) \to f_4(e, p, pp') : | : pp' = p + 1$$

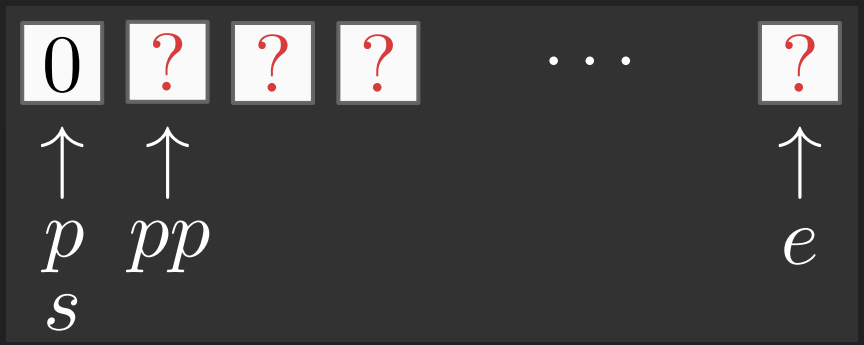
ITS



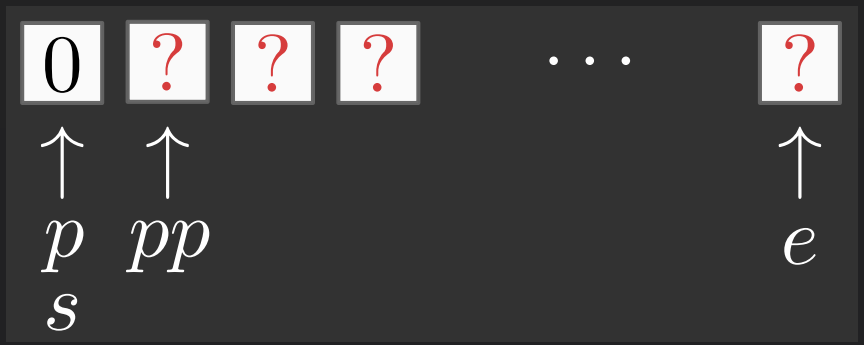
$$f_1(e, p, pp) \to f_2(e, p, pp) : | : pp < e$$
$$f_2(e, p, pp) \to f_3(e, p', pp) : | : p' = pp$$
$$f_3(e, p, pp) \to f_4(e, p, pp') : | : pp' = p + 1$$
$$f_4(e, p, pp) \to f_1(e, p, pp)$$

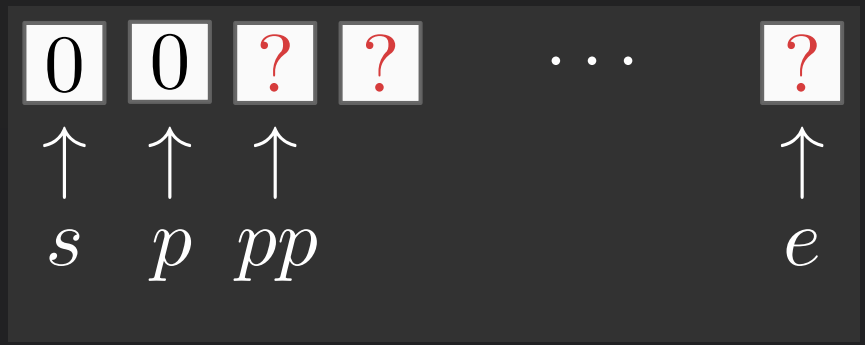
Old Version

Heap Invariants

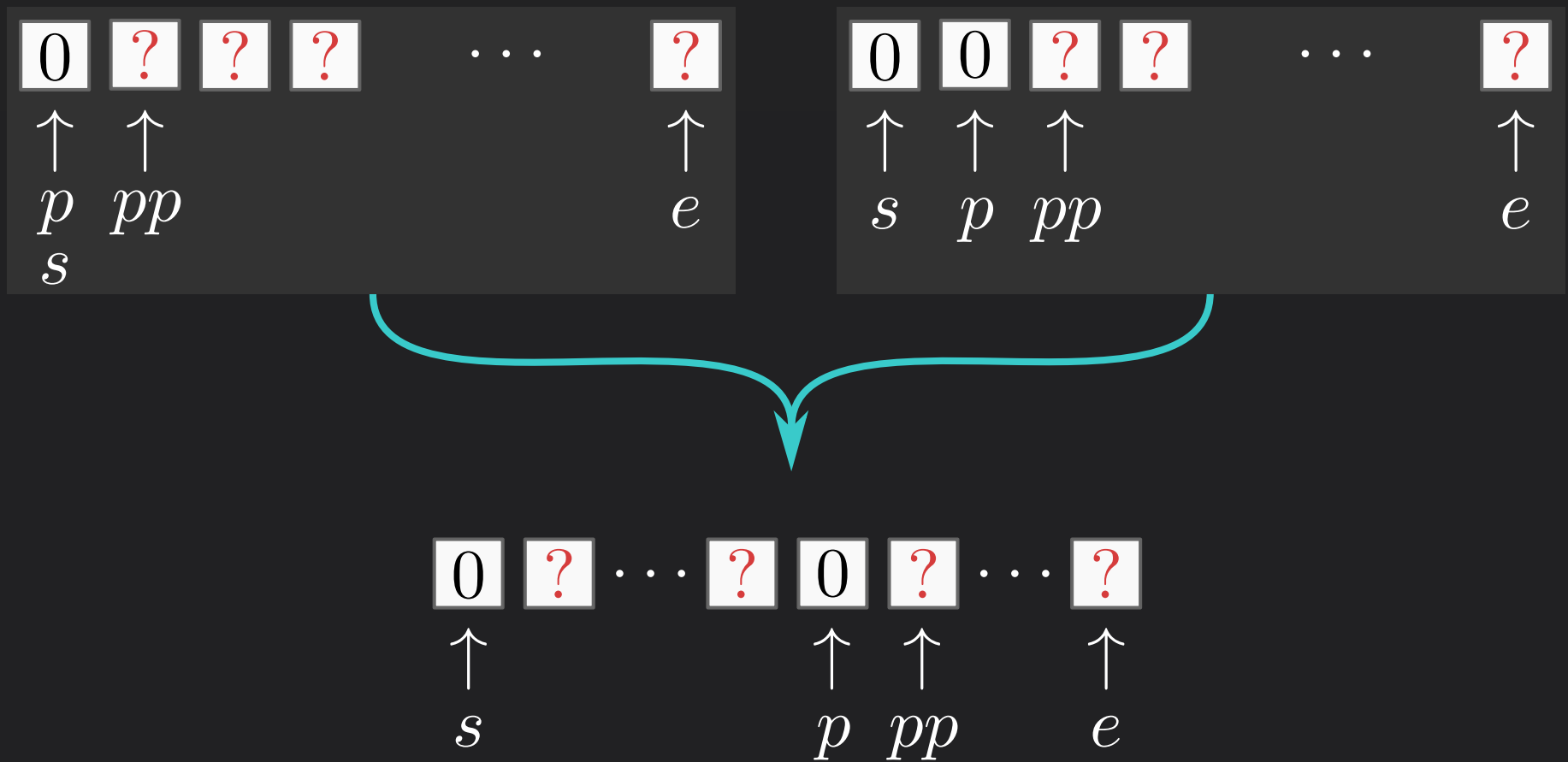


Heap Invariants

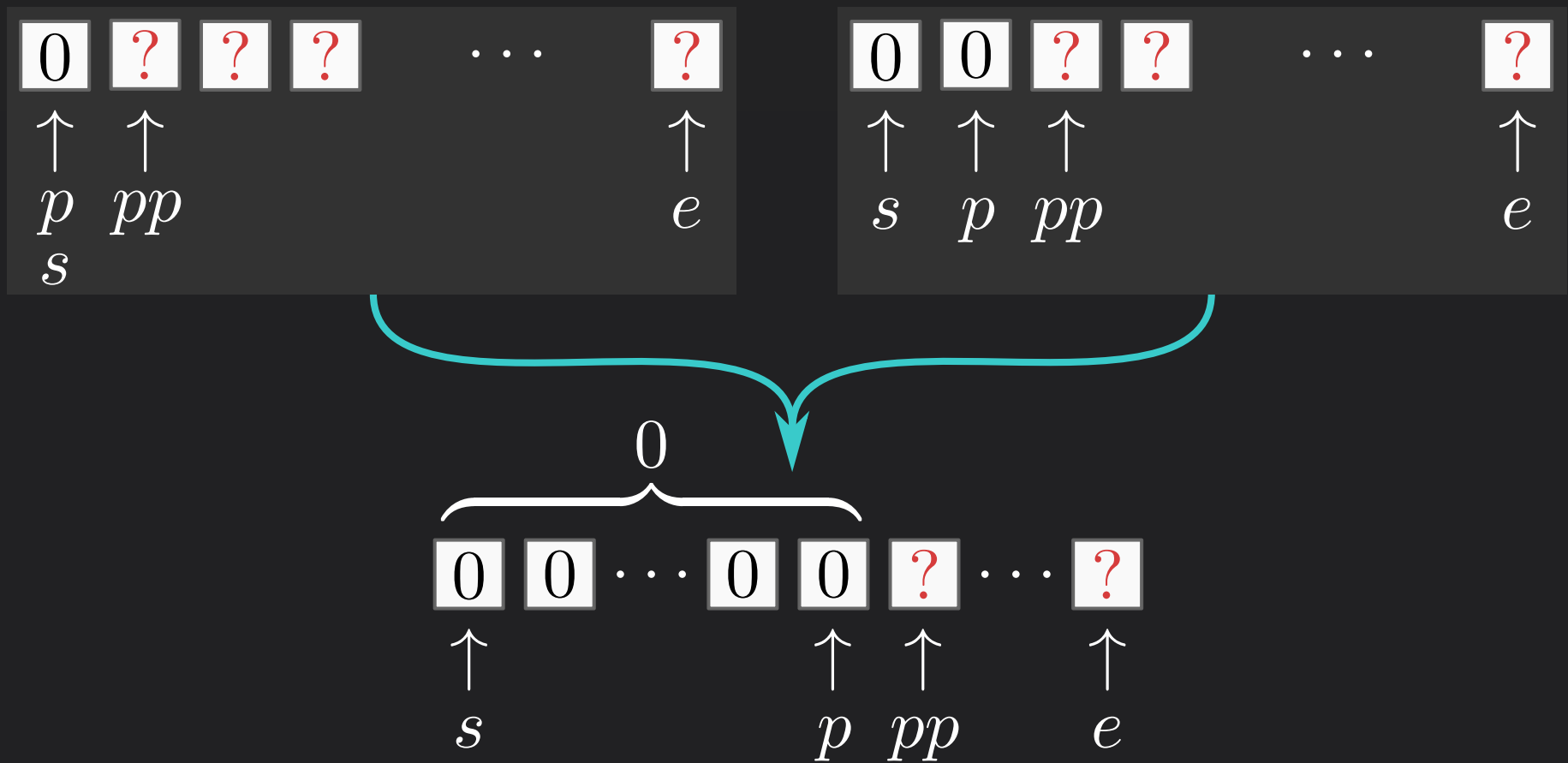




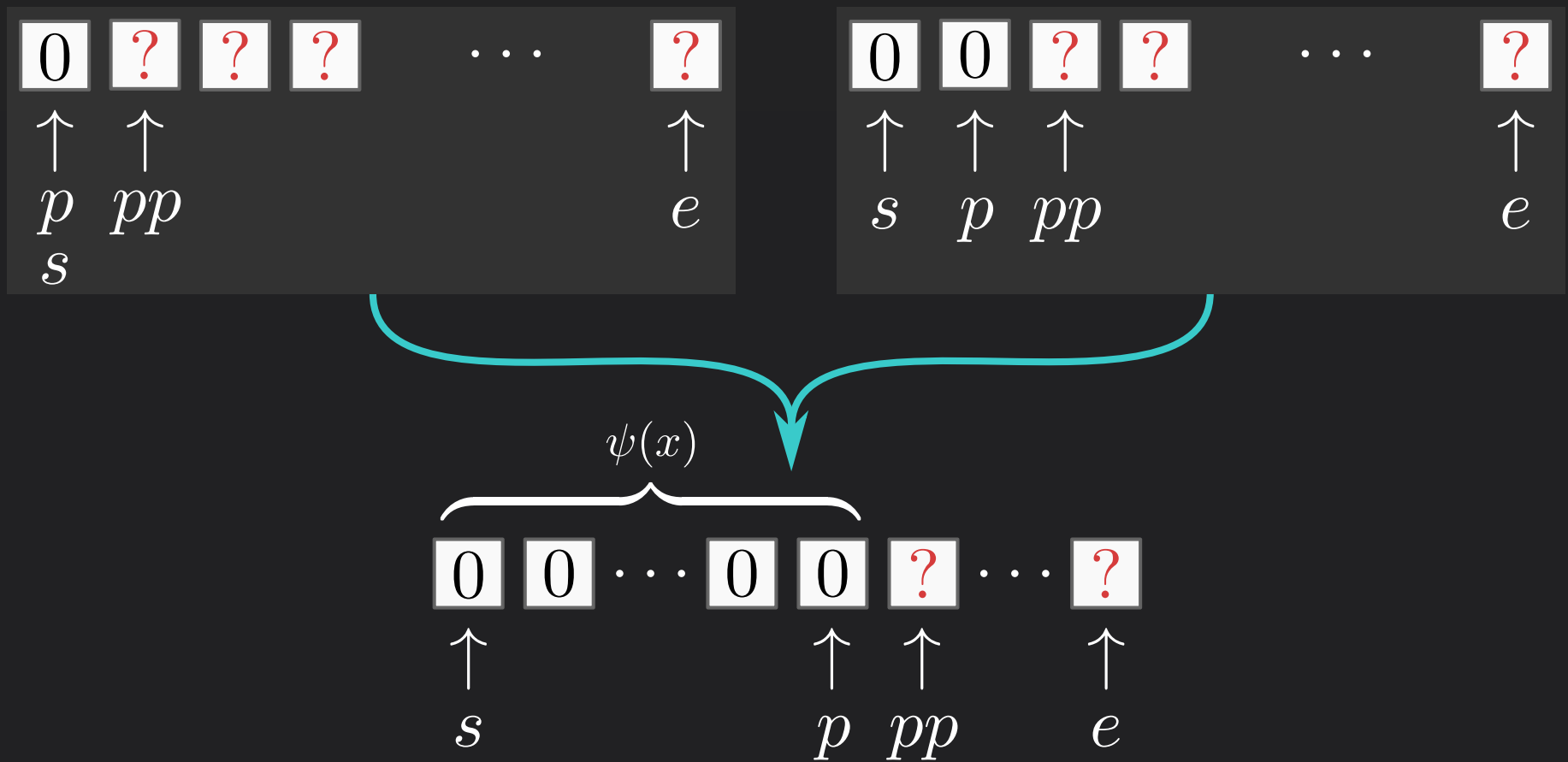
Heap Invariants



Heap Invariants



Heap Invariants



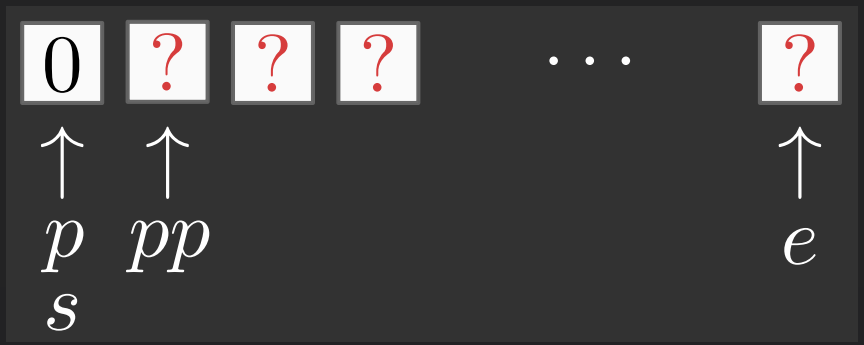
Heap Invariants

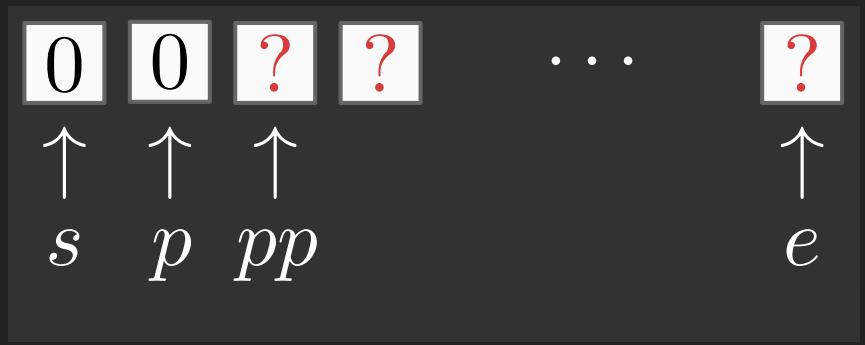
$$\mathtt{ad} \hookrightarrow y$$

Heap Invariants

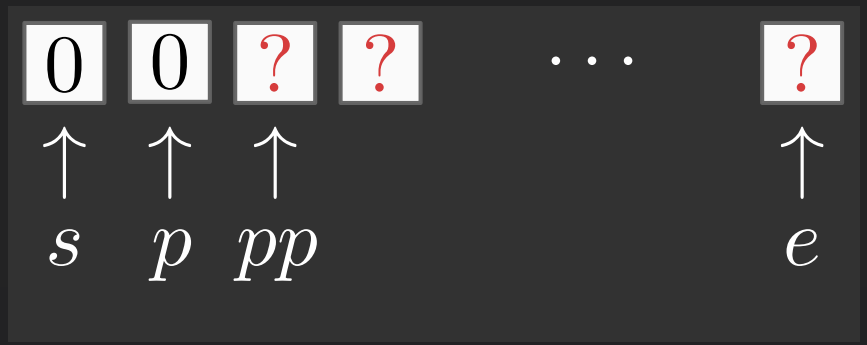
$$[\mathtt{ad}_1, \mathtt{ad}_2] \hookrightarrow v$$

Heap Invariants

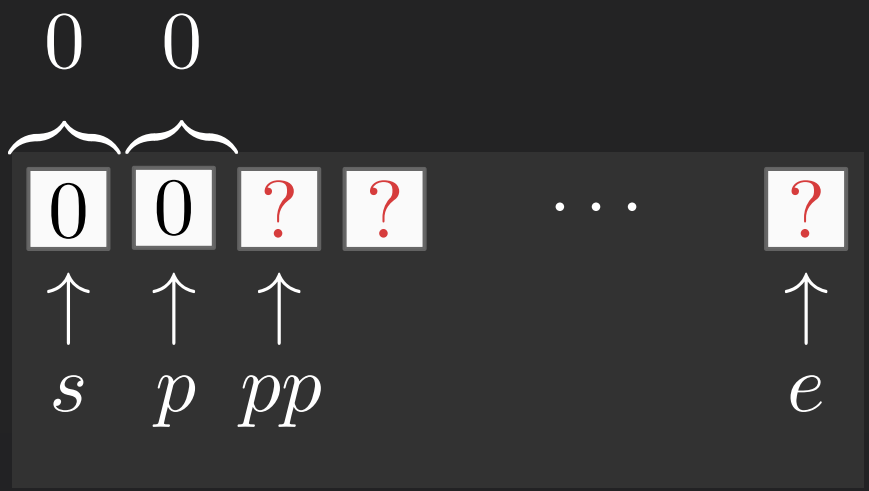




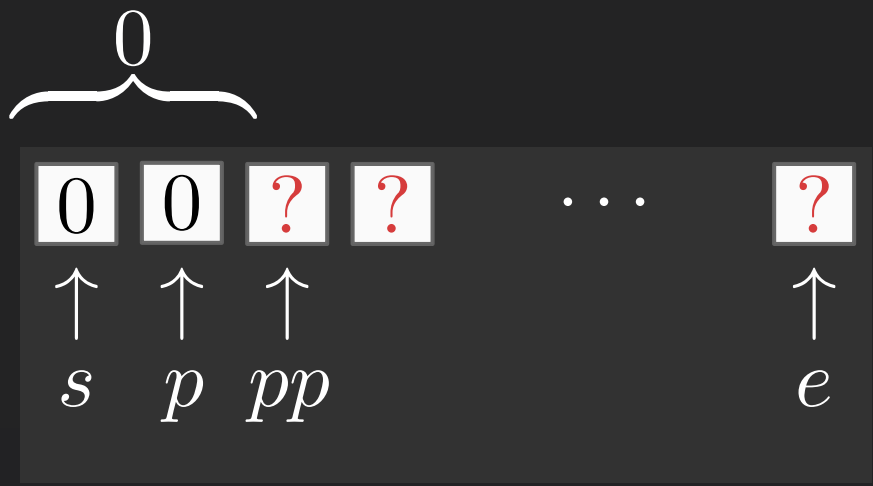
Heap Invariants



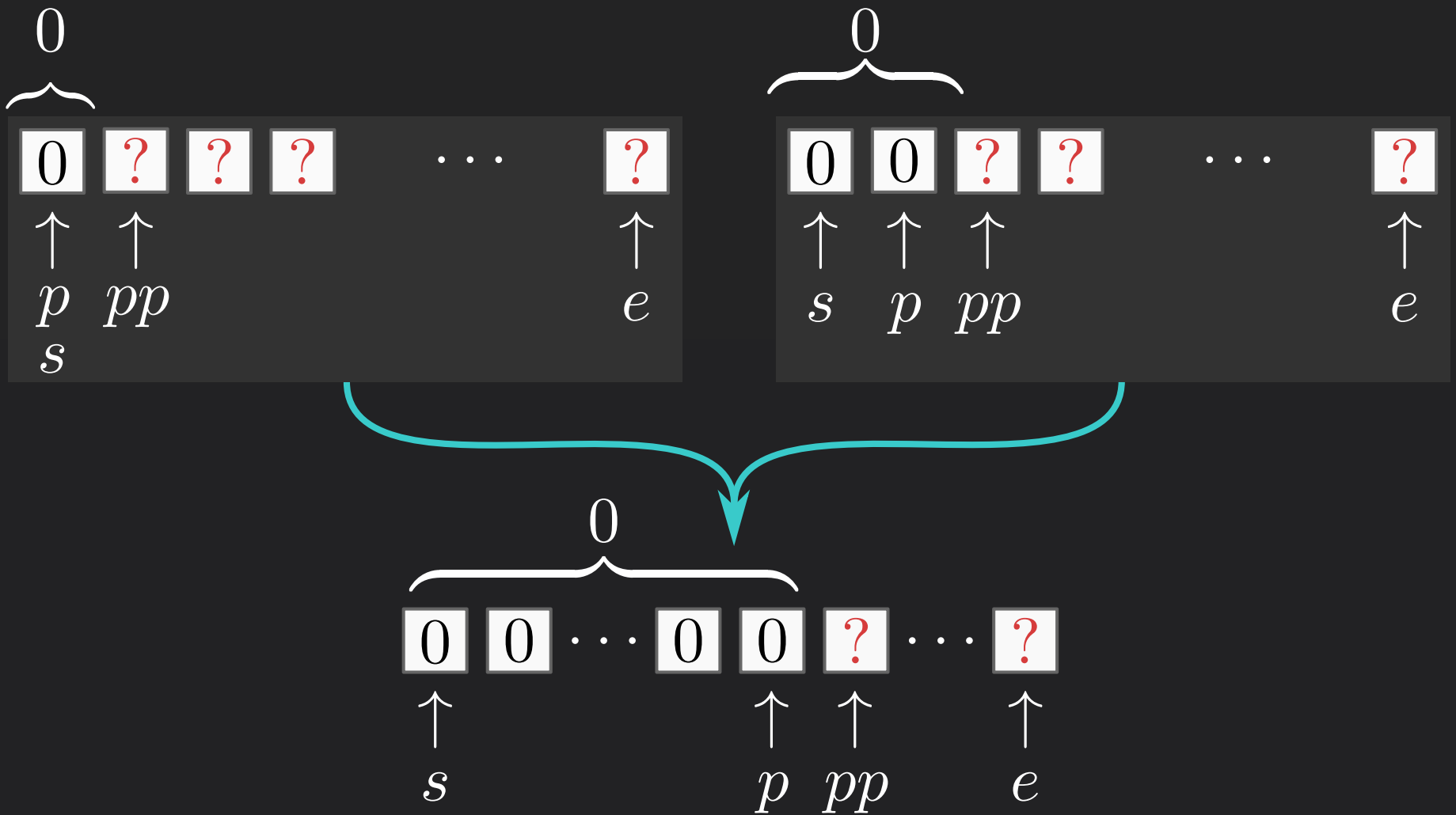
Heap Invariants



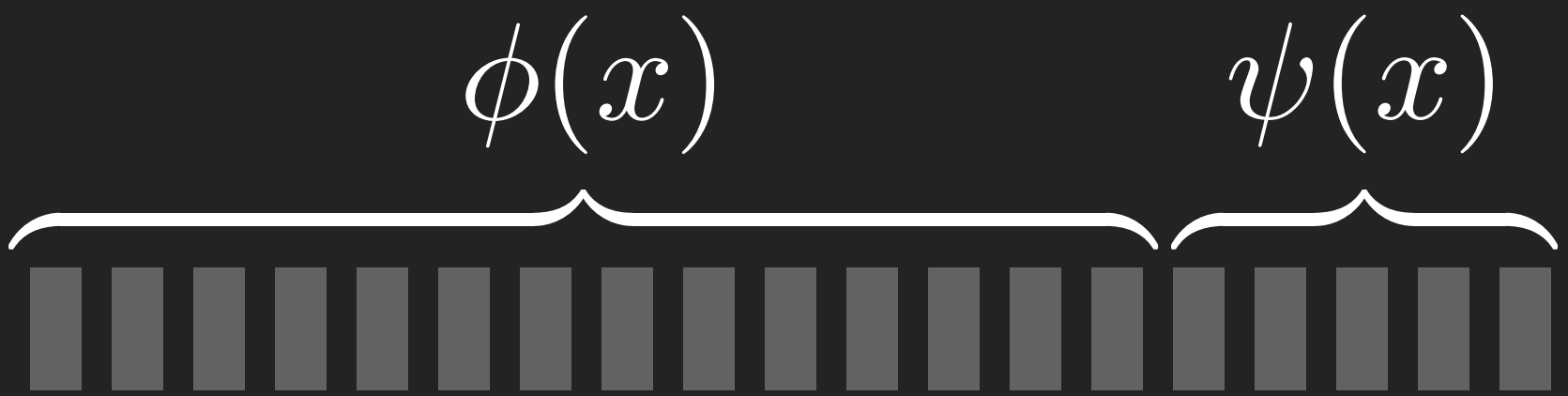
Heap Invariants



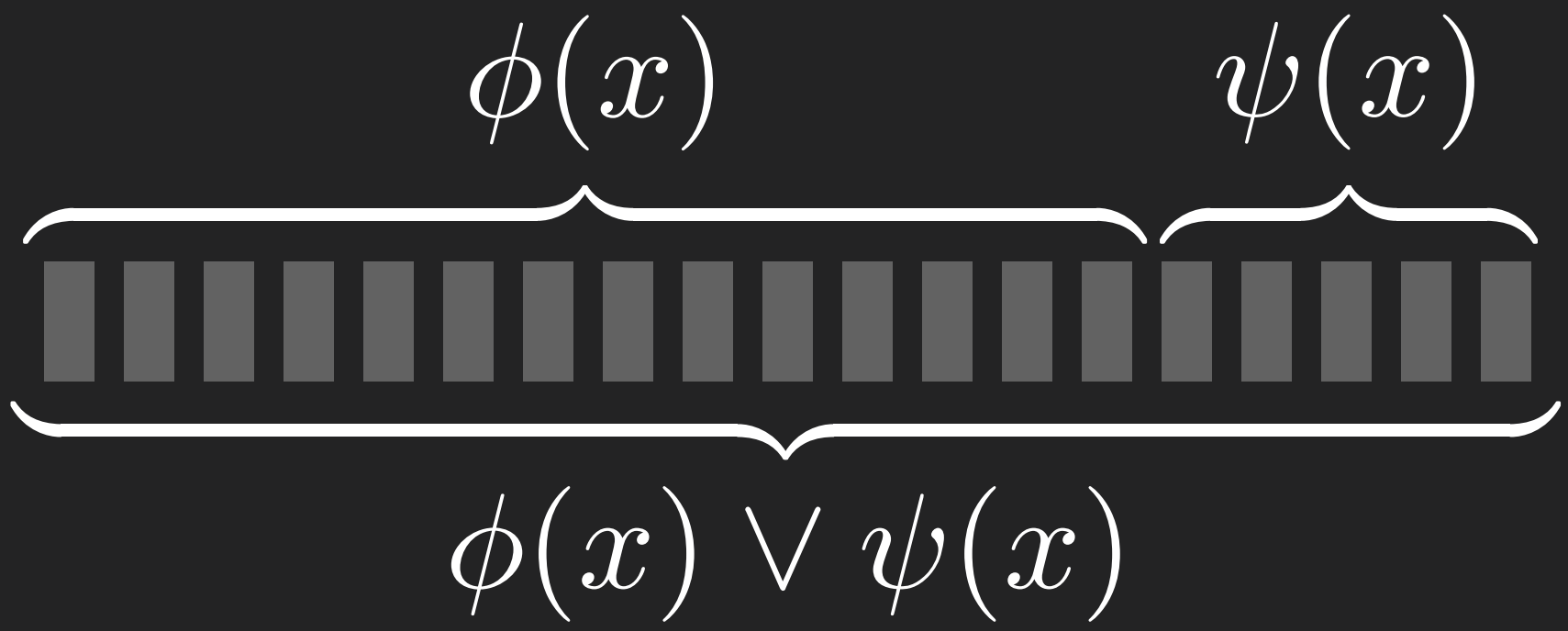
Heap Invariants



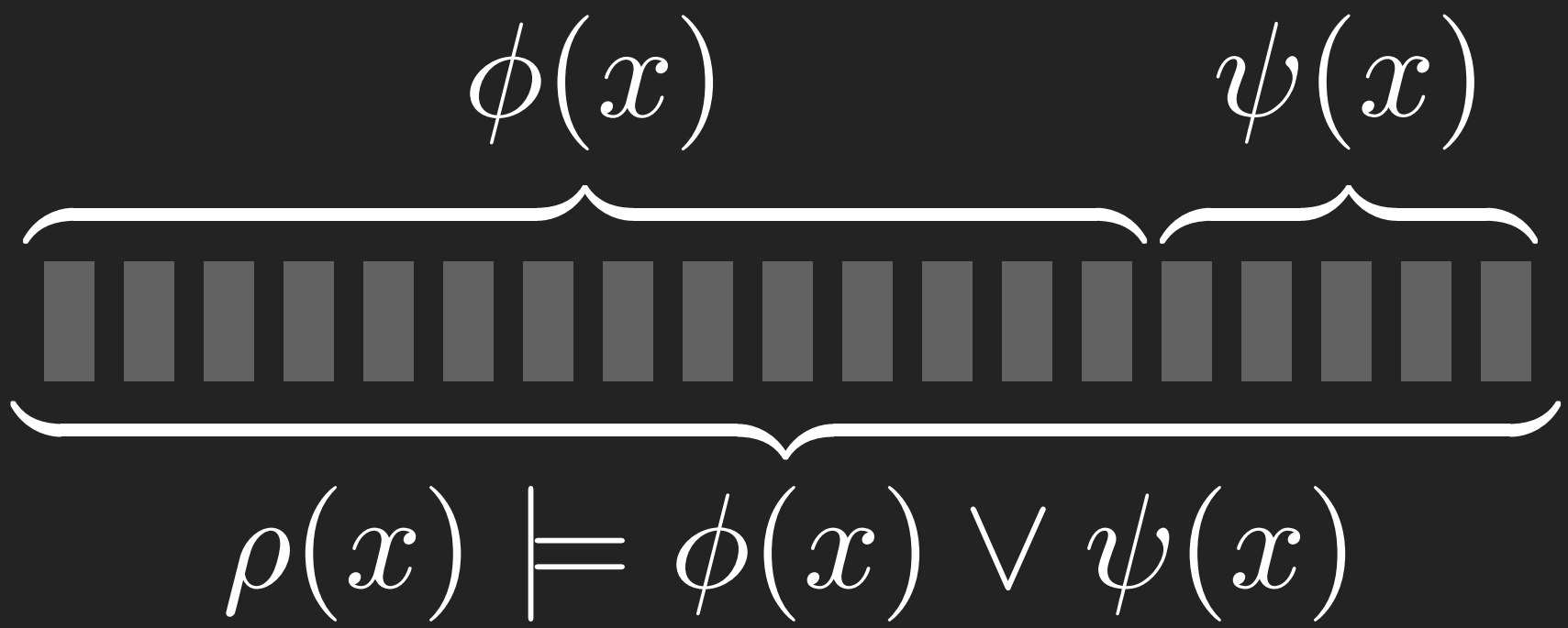
Join



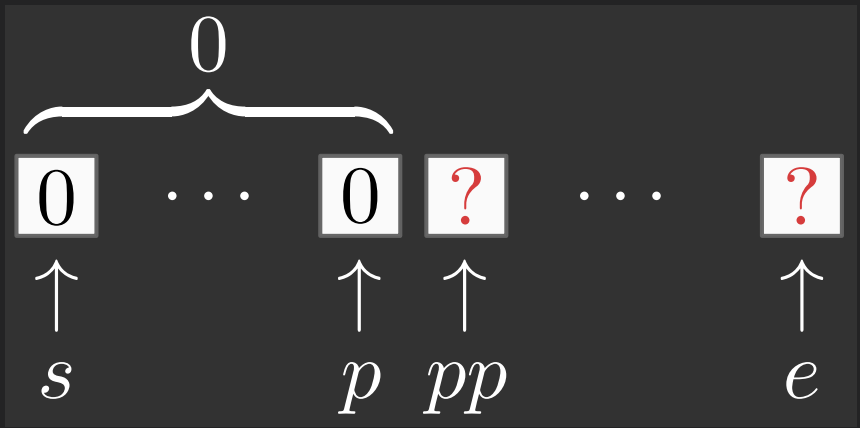
Join



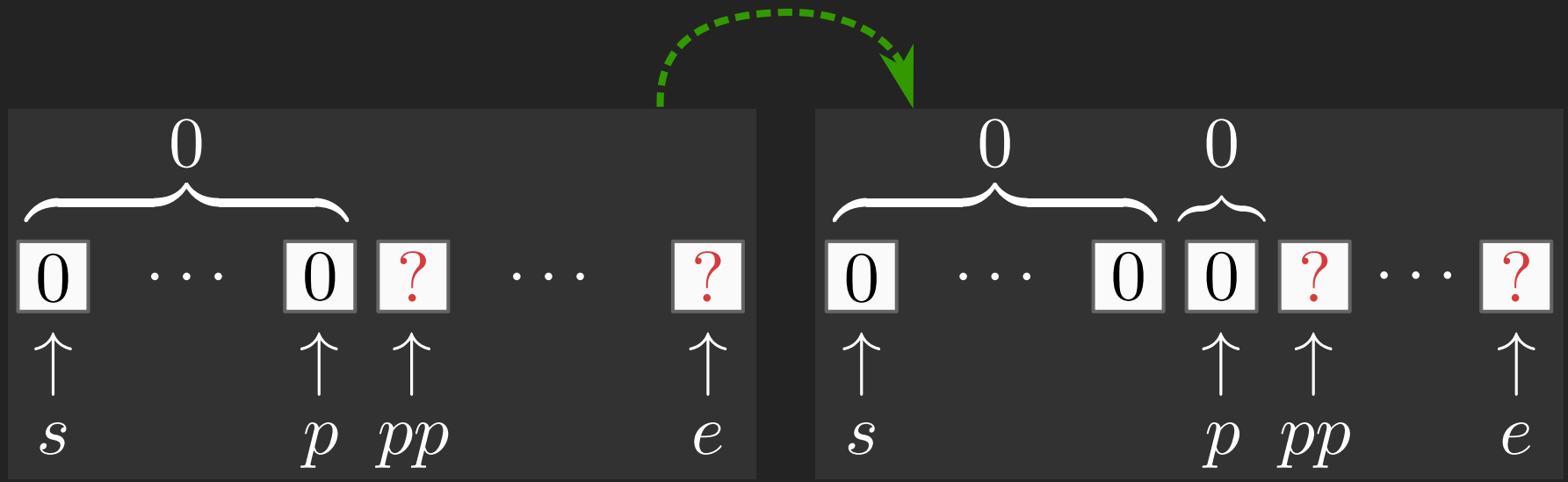
Join



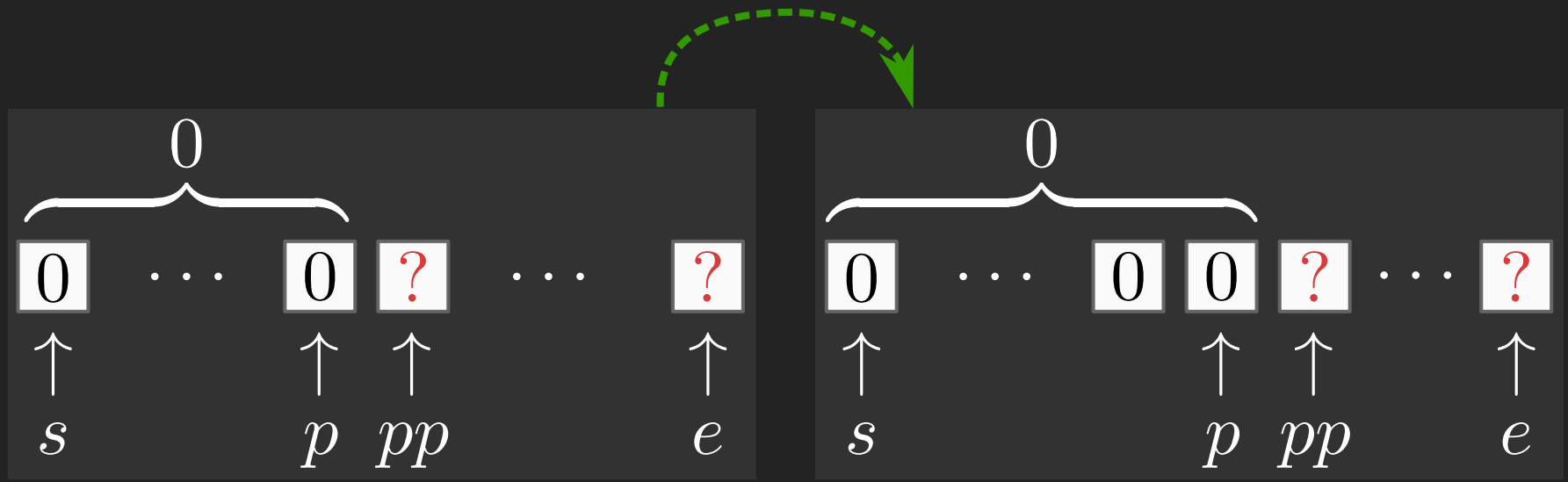
Induction Proof



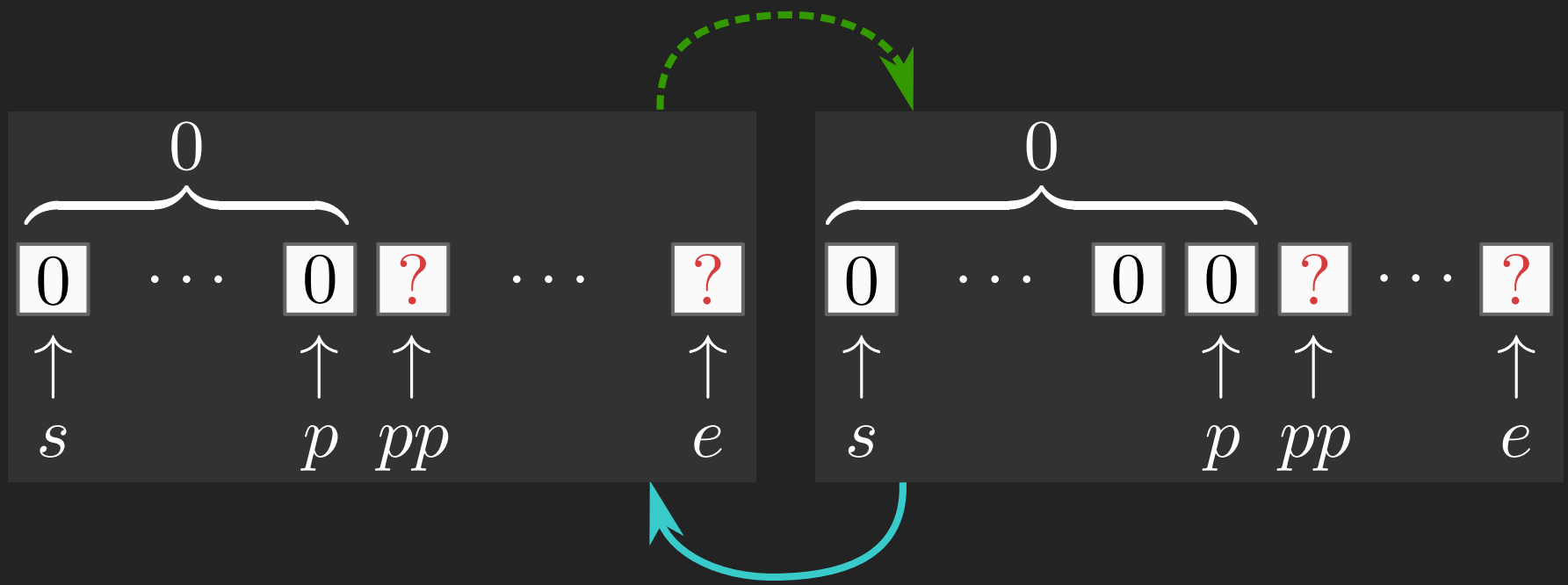
Induction Proof



Induction Proof



Induction Proof



Samples

Samples

```
char* memset(char *s, char *a, char *e) {
        while(s < e) {*s = 0; s++;}
        return e;
}
```

Samples

```
char* memset(char *s, char *a, char *e) {
        while(s < e) {*s = 0; s++;}
        if( s < a && a < e ) {
                if( *a != 0) { violation();}
        }
        return a;
}
```

Samples

```
void printlines(char *str, char **saveptr){
        int c = 0;
        int i = 0;
        saveptr[i++] = str;
        while(str[c] != 0){
                if(str[c] == '\n'){
                        saveptr[i++] = &str[c+1];
                        str[c] = 0;
                }
                c++;
        }
        for(int p = 0; p < i; p++){
                printf("got line %s\n",saveptr[p]);
        }
}
```

Samples

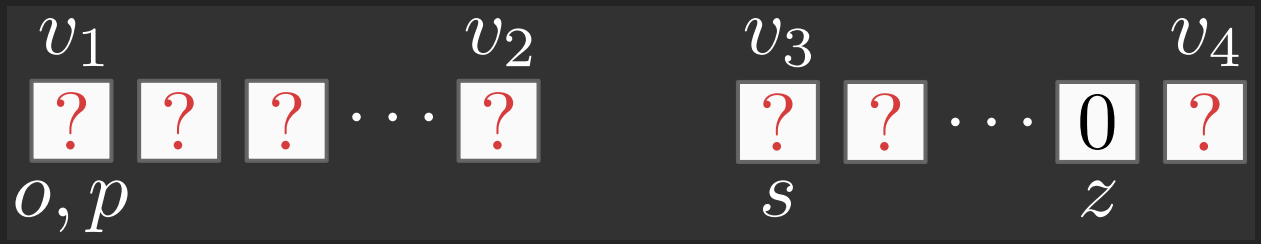
```
char** osaveptr = saveptr;
while(str < zero){
        *saveptr++ = str++
}
while(osaveptr < saveptr){
        printf("got line %s\n",*osaveptr++);
}
```

Build

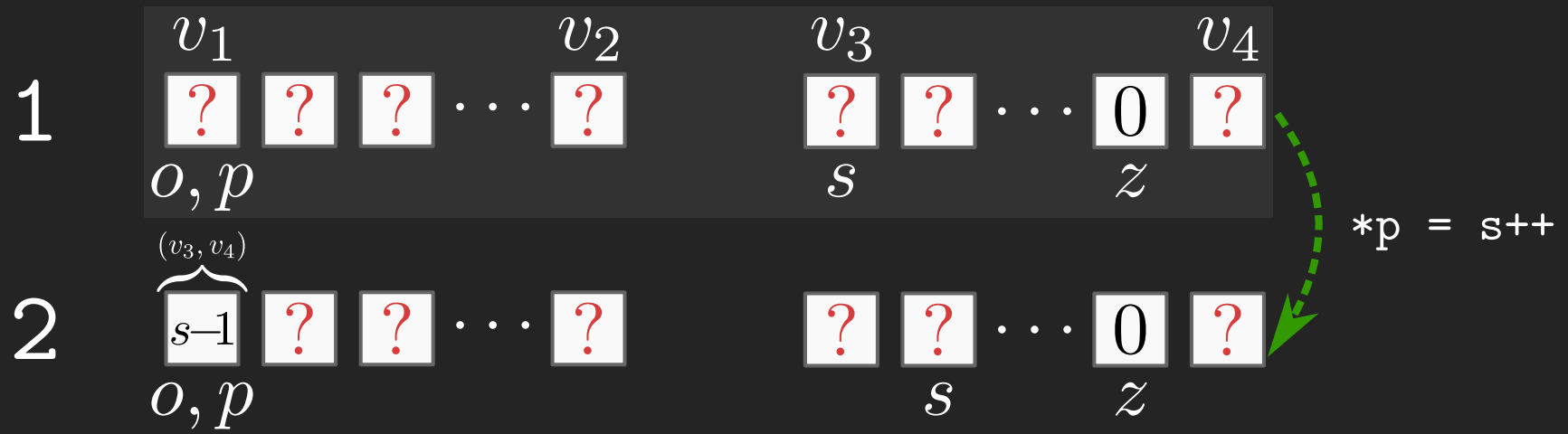
```
char** osaveptr = saveptr;
while(str < zero){
        *saveptr++ = str++
}
while(osaveptr < saveptr){
        printf("got line %s\n",*osaveptr++);
}
```

Build

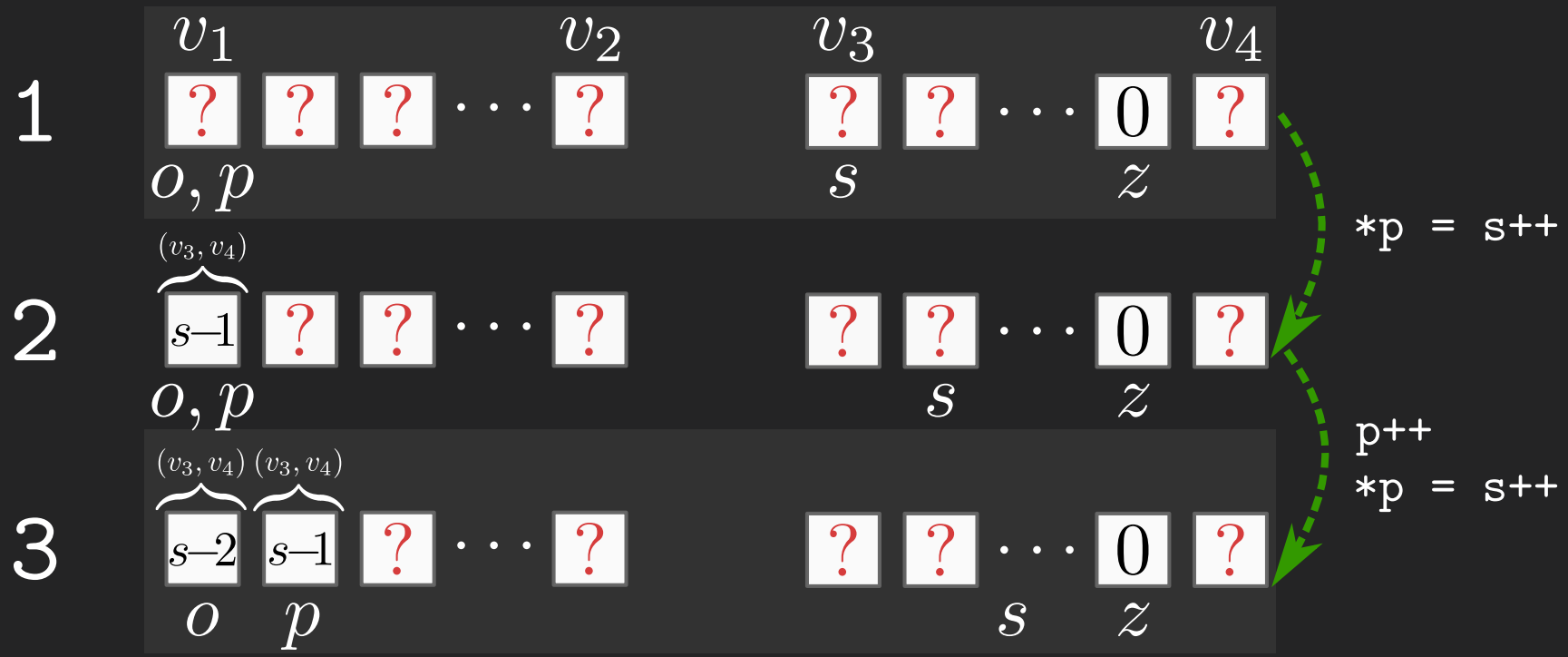
1



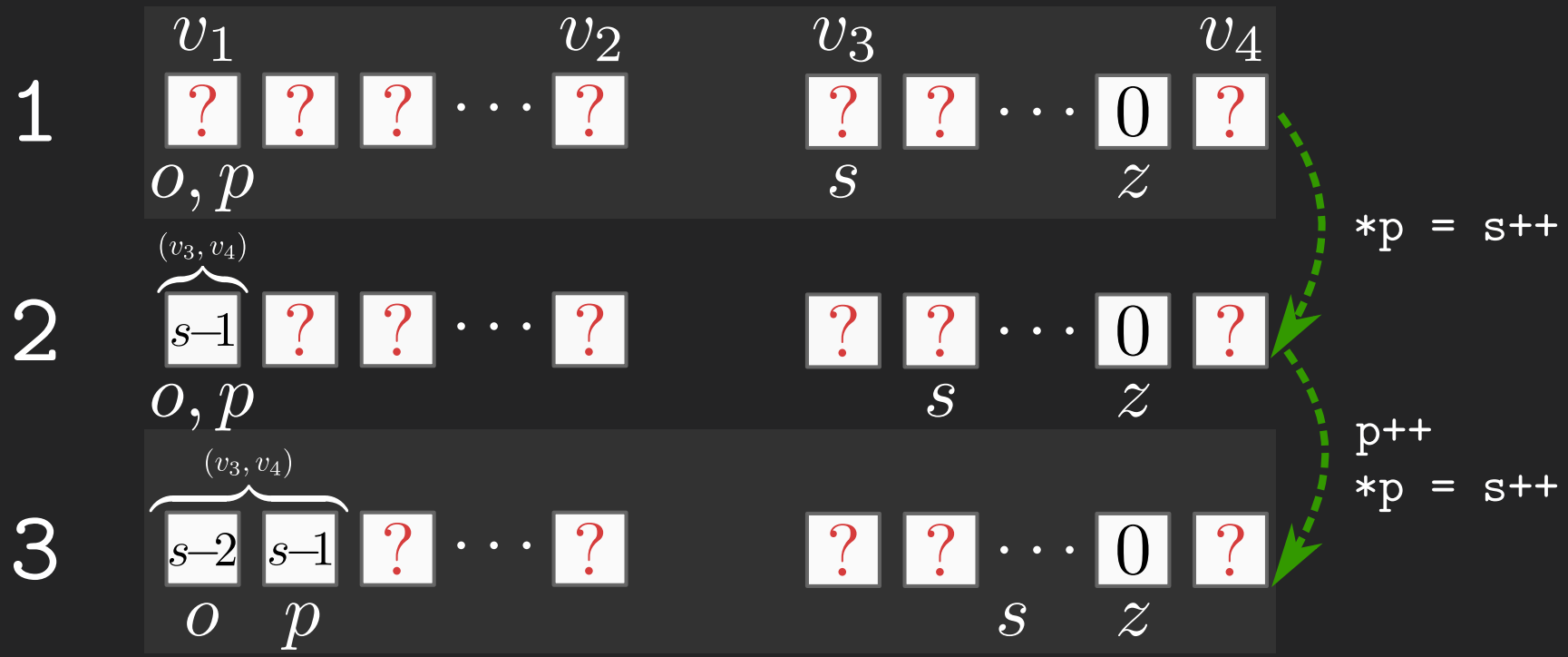
Build



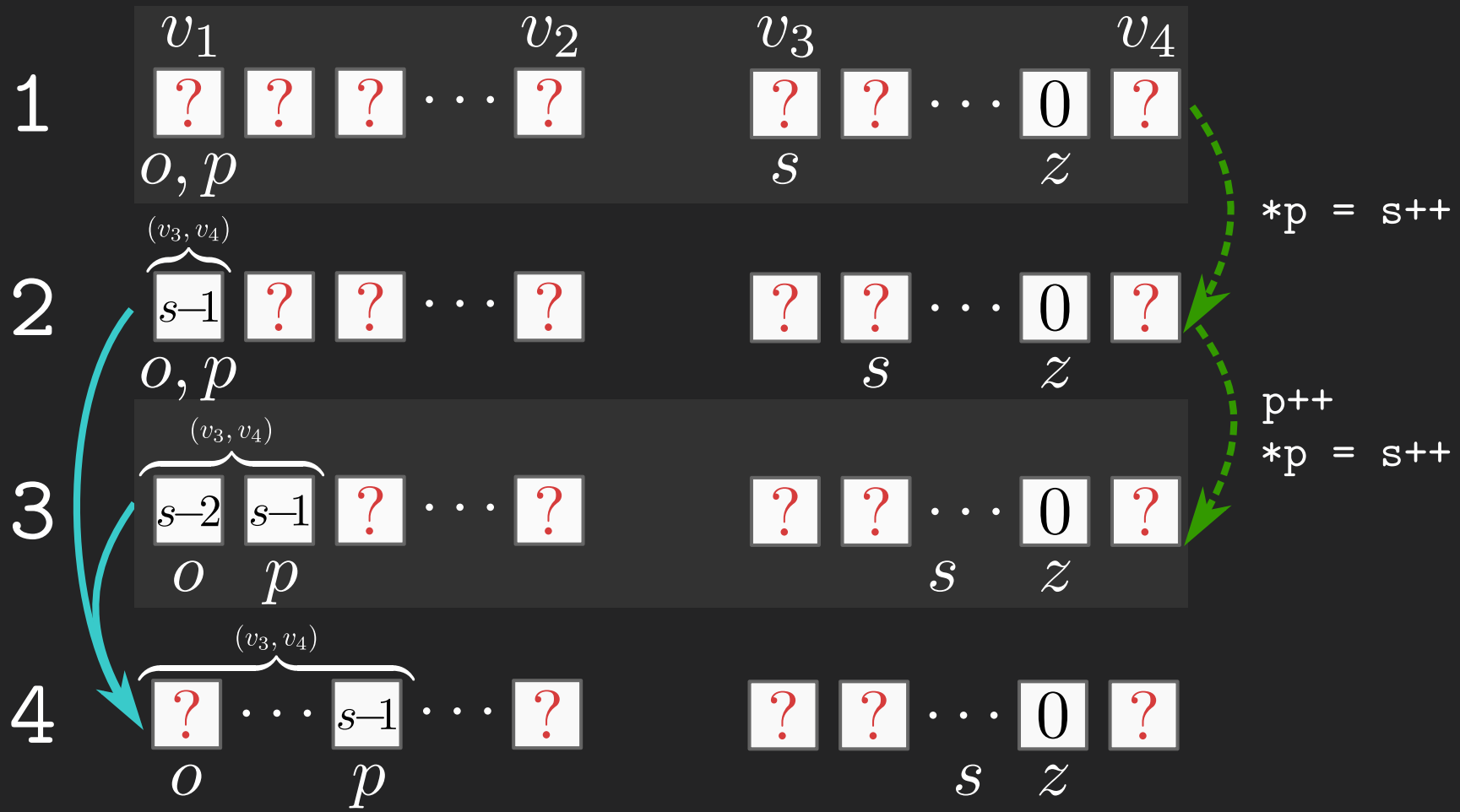
Build



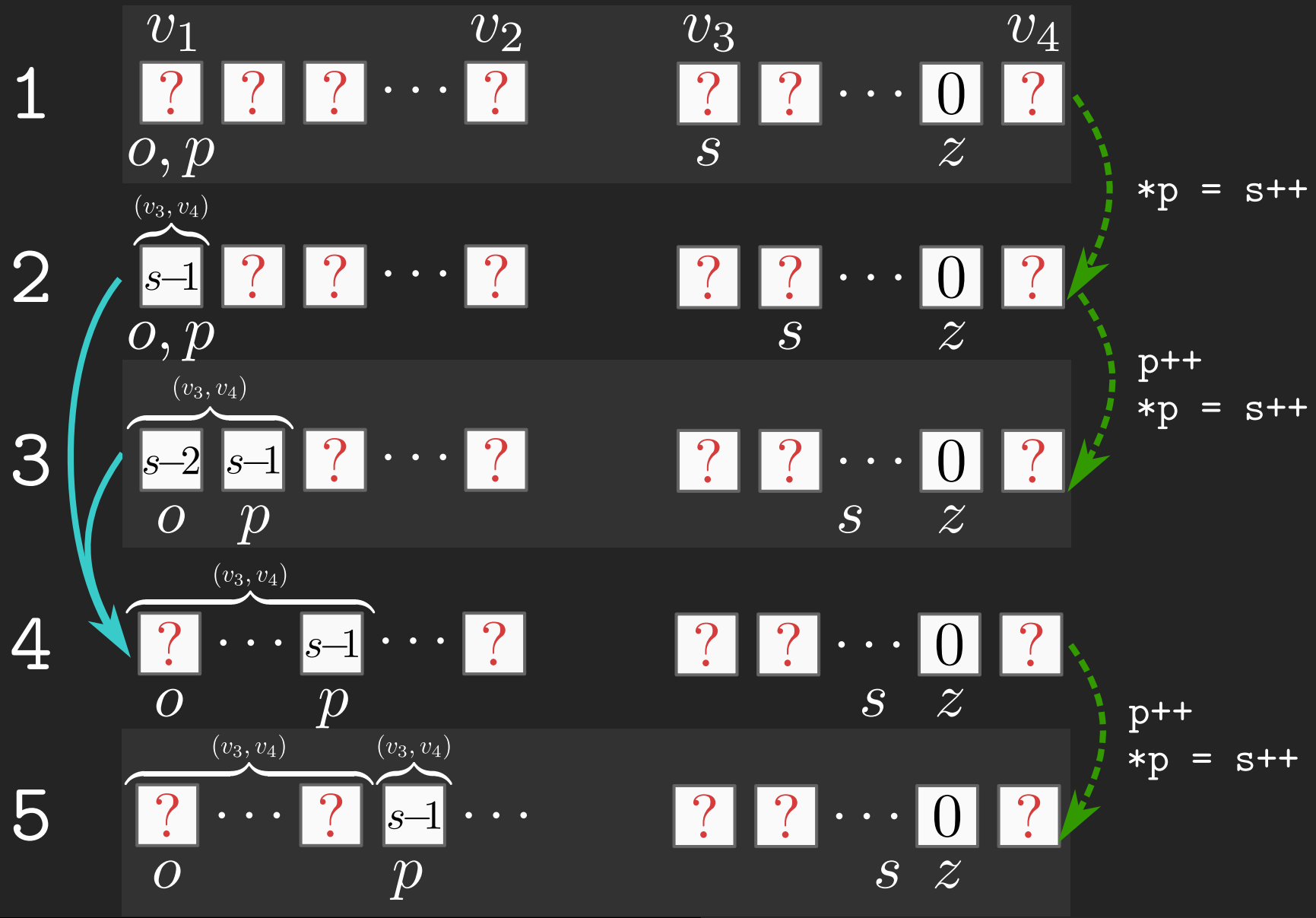
Build



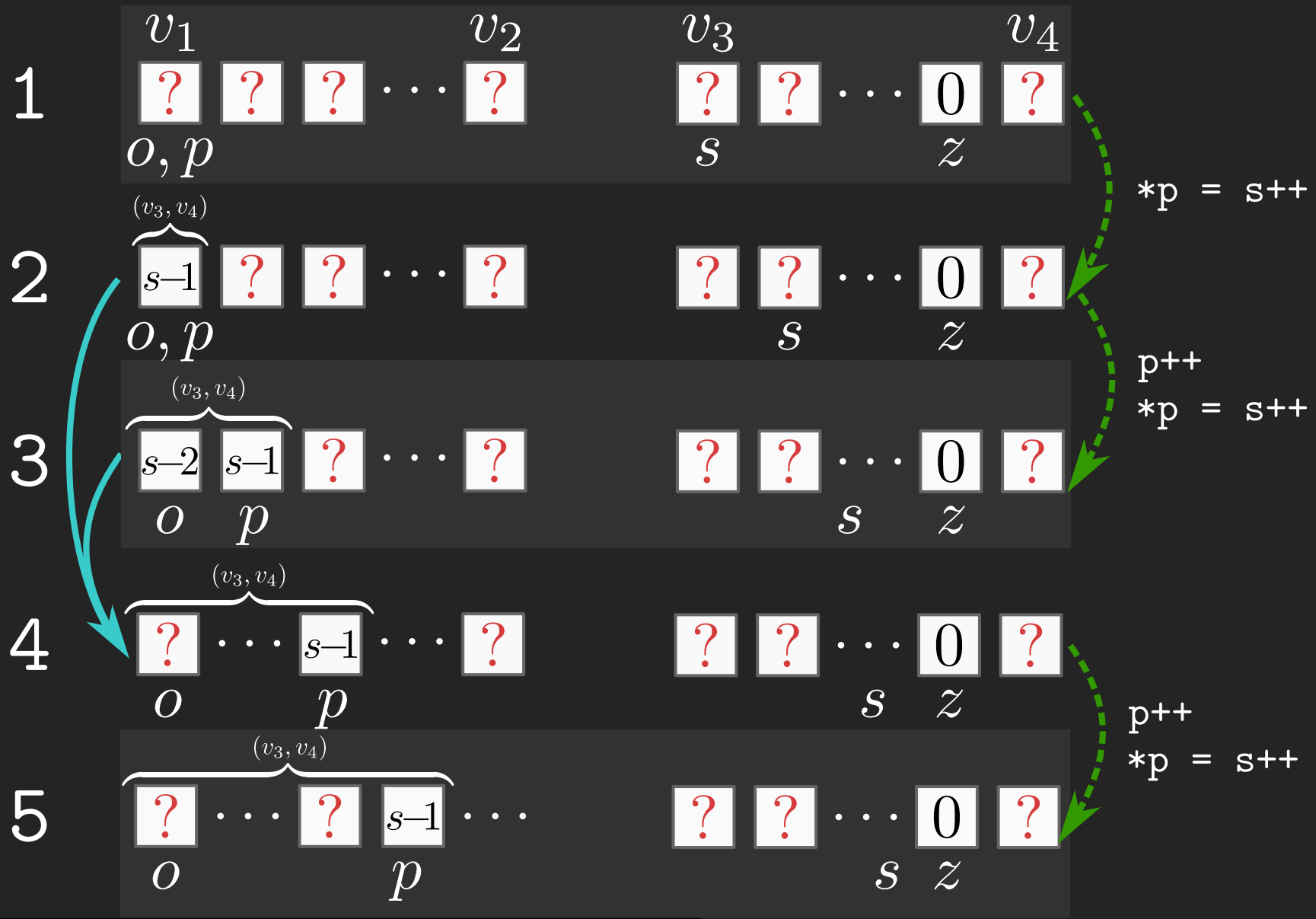
Build



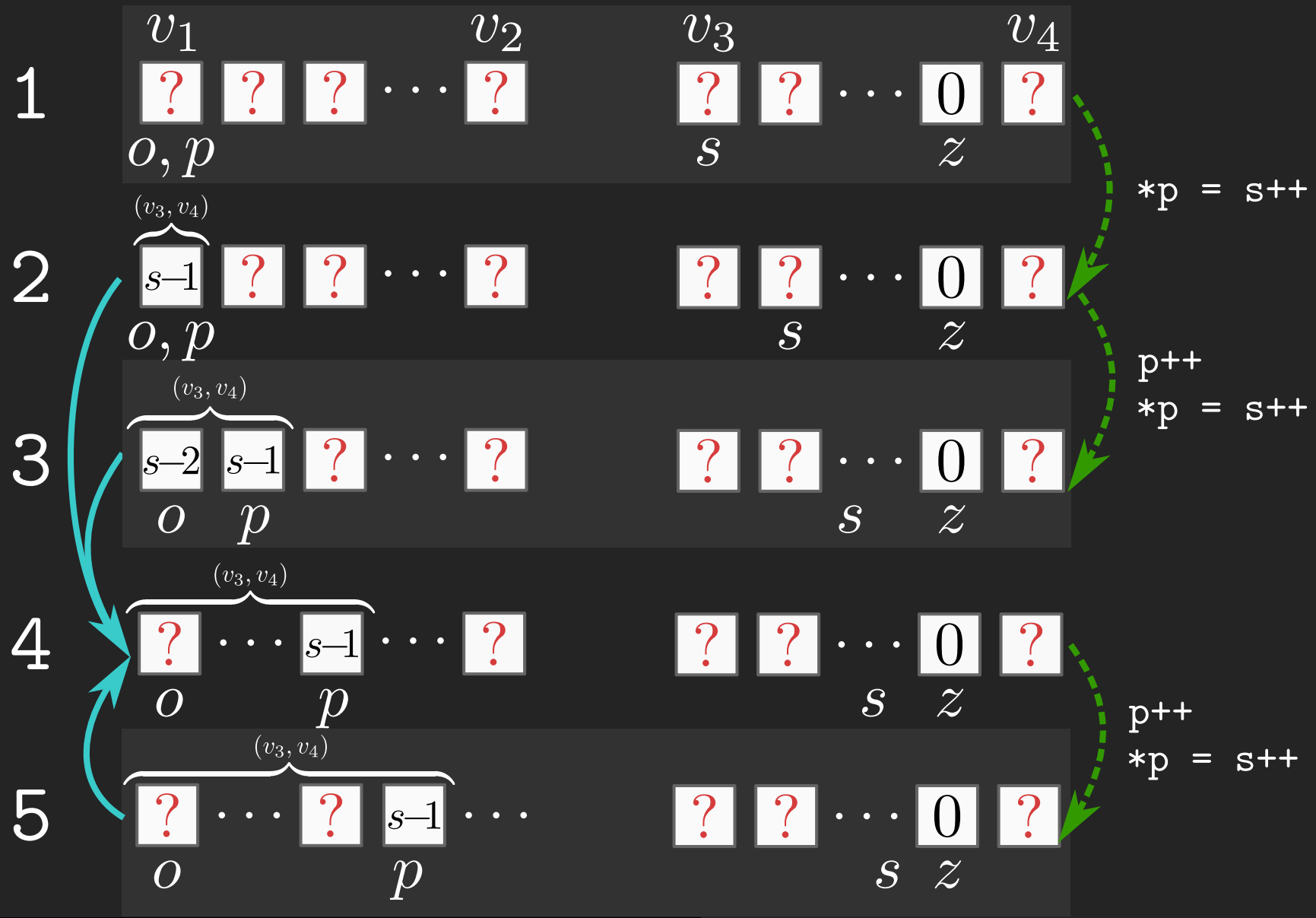
Build



Build



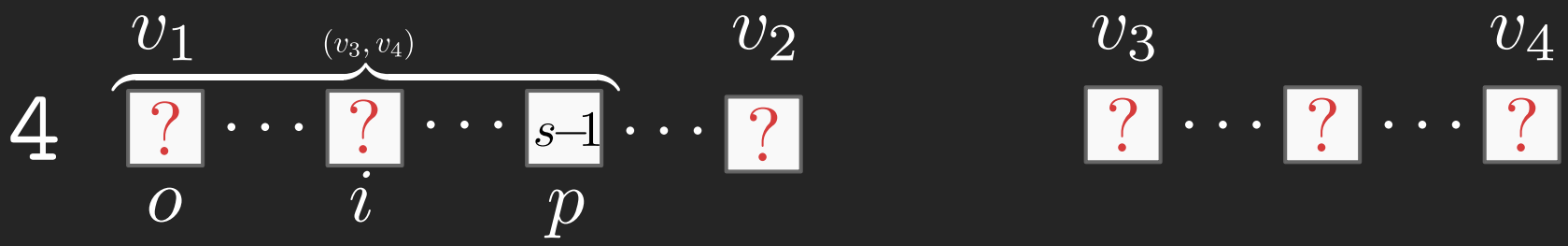
Build



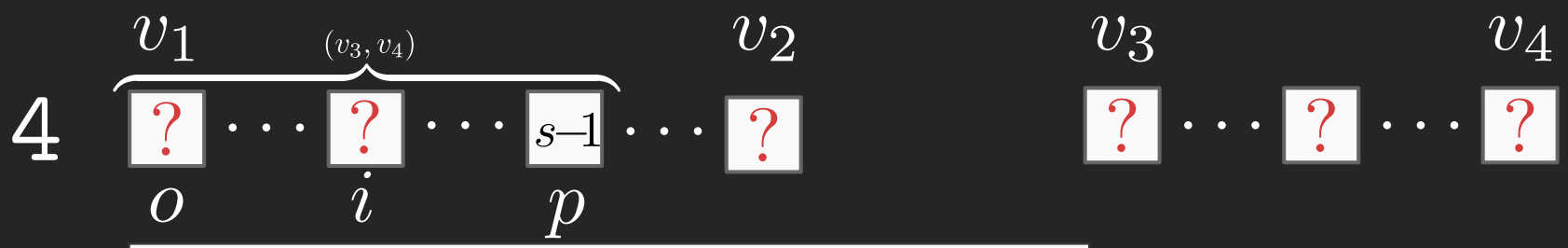
Use

```
char** osaveptr = saveptr;
while(str < zero){
        *saveptr++ = str++
}
while(osaveptr < saveptr){
        printf("got line %s\n",*osaveptr++);
}
```

Use



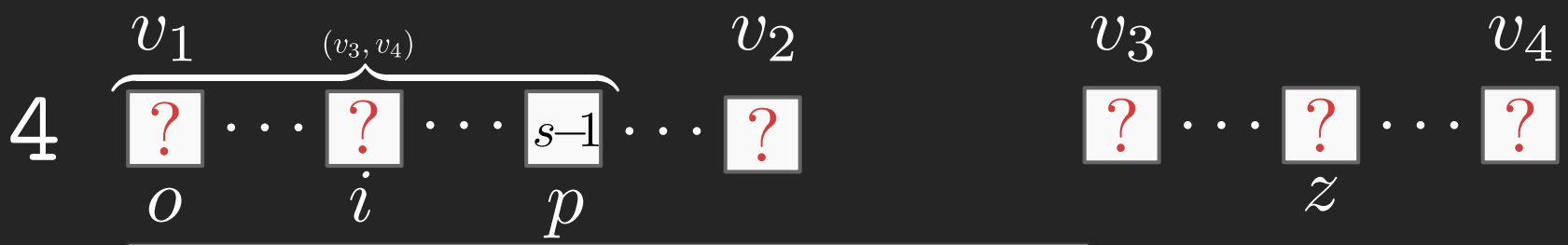
Use



4

$$A : o, i, p \in (v_1, v_2), v \in (v_3, v_4)$$
$$K : o \leq i \leq p, v_3 \leq v \leq v_4$$
$$P : [o, p] \hookrightarrow v$$

Use



4

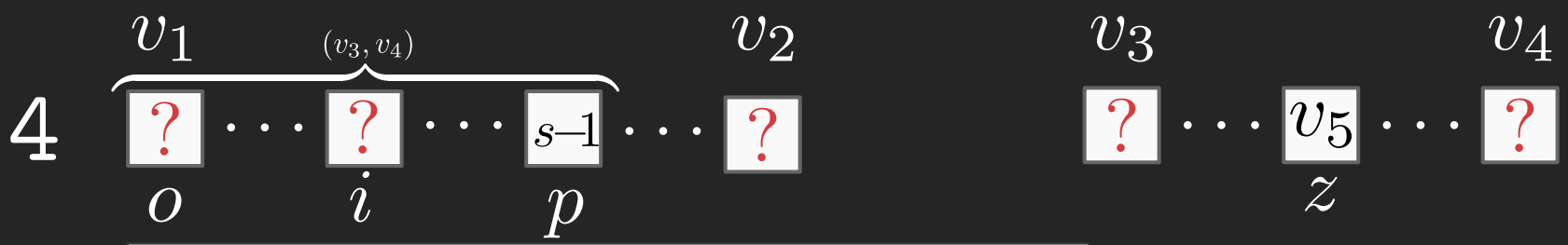
$$A : o, i, p \in (v_1, v_2), v \in (v_3, v_4)$$
$$K : o \leq i \leq p, v_3 \leq v \leq v_4$$
$$P : [o, p] \hookrightarrow v$$

`z = load i`

8

$$A : o, i, p \in (v_1, v_2), v, z \in (v_3, v_4)$$
$$K : o \leq i \leq p, v_3 \leq v \leq v_4, v_3 \leq z \leq v_4$$
$$P : [o, p] \hookrightarrow v, [i, i] \hookrightarrow z,$$

Use



4

$A : o, i, p \in (v_1, v_2), v \in (v_3, v_4)$

$K : o \leq i \leq p, v_3 \leq v \leq v_4$

$P : [o, p] \hookrightarrow v$

`z = load i`

8

$A : o, i, p \in (v_1, v_2), v, z \in (v_3, v_4)$

$K : o \leq i \leq p, v_3 \leq v \leq v_4, v_3 \leq z \leq v_4$

$P : [o, p] \hookrightarrow v, [i, i] \hookrightarrow z,$

`d = load z`

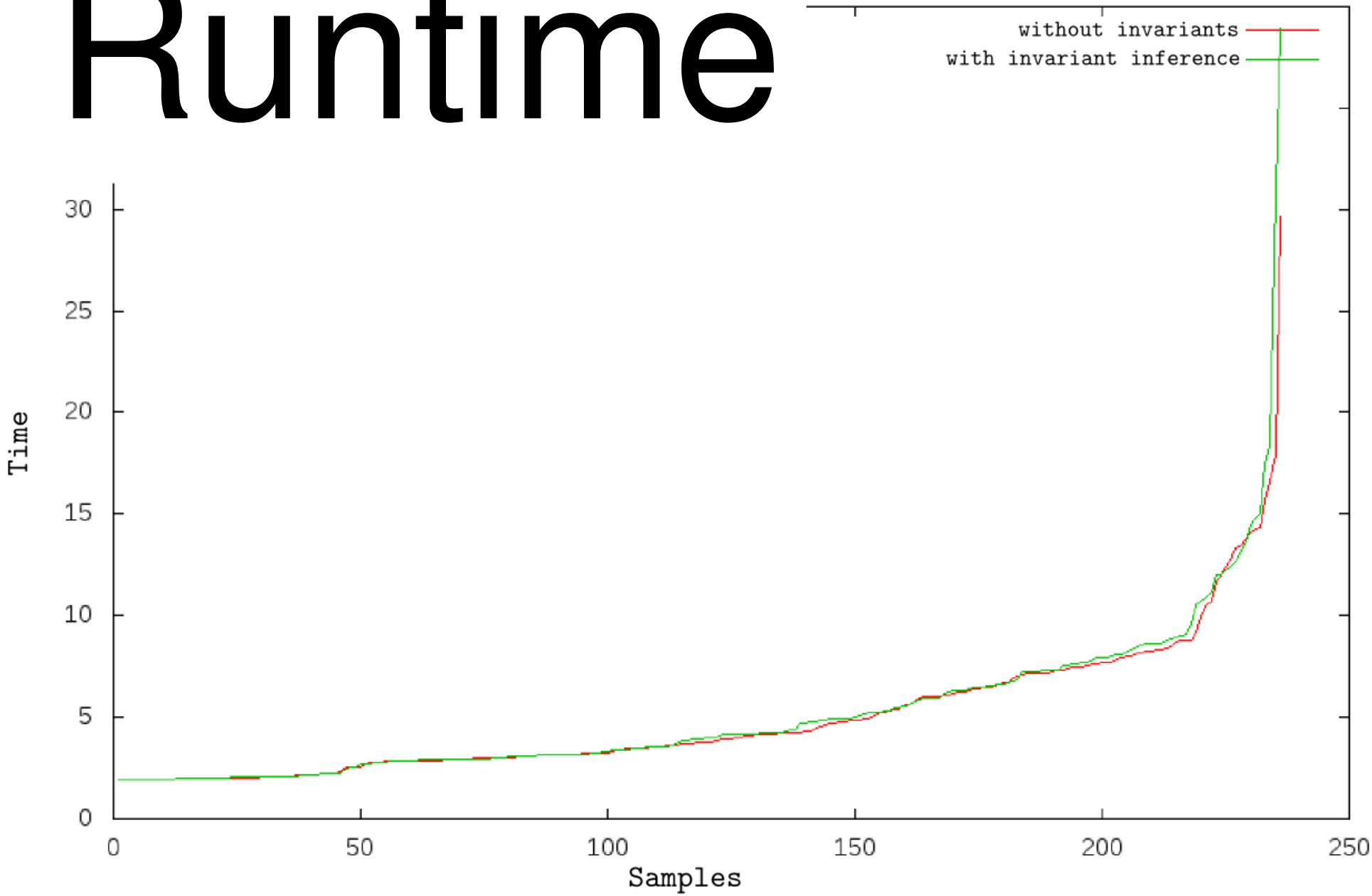
9

$A : o, i, p \in (v_1, v_2), v, z \in (v_3, v_4)$

$K : \cdots, v_3 \leq z \leq v_4, d = v_5$

$P : [o, p] \hookrightarrow v, [i, i] \hookrightarrow z, [z, z] \hookrightarrow v_5$

Runtime


without invariants
with invariant inference
Time
Samples
0
5
10
15
20
25
30
0
50
100
150
200
250

```c
int main(int argc, char* argv[]){
  for(int i=0; i<argc; i++){
    printf("Hallo %s",argv[i]);
  }
}
```

- LLVM
- Symbolic Execution
- ITS
- Invariants