Dataflow Analysis

Guido Wachsmuth

Course IN4303, 2012/13
Compiler Construction

Overview

today's lecture

control flow graphs

Overview

today's lecture

control flow graphs

data flow analyses

- liveness analysis
- reaching definitions
- available expressions

Overview

today's lecture

control flow graphs

data flow analyses

- liveness analysis
- reaching definitions
- available expressions

non-local optimisations

- dead code elimination
- constant & copy propagation
- common subexpression elimination

I

control-flow graphs

Intermediate language

quadruples

store

$a \leftarrow b \oplus c$

$a \leftarrow b$

memory access

$a \leftarrow M[b]$

$M[a] \leftarrow b$

functions

$f(a_1, \ldots, a_n)$

$b \leftarrow f(a_1, \ldots, a_n)$

jumps

L:

goto L

if $a \otimes b$

  goto L_1

else

  goto L_2

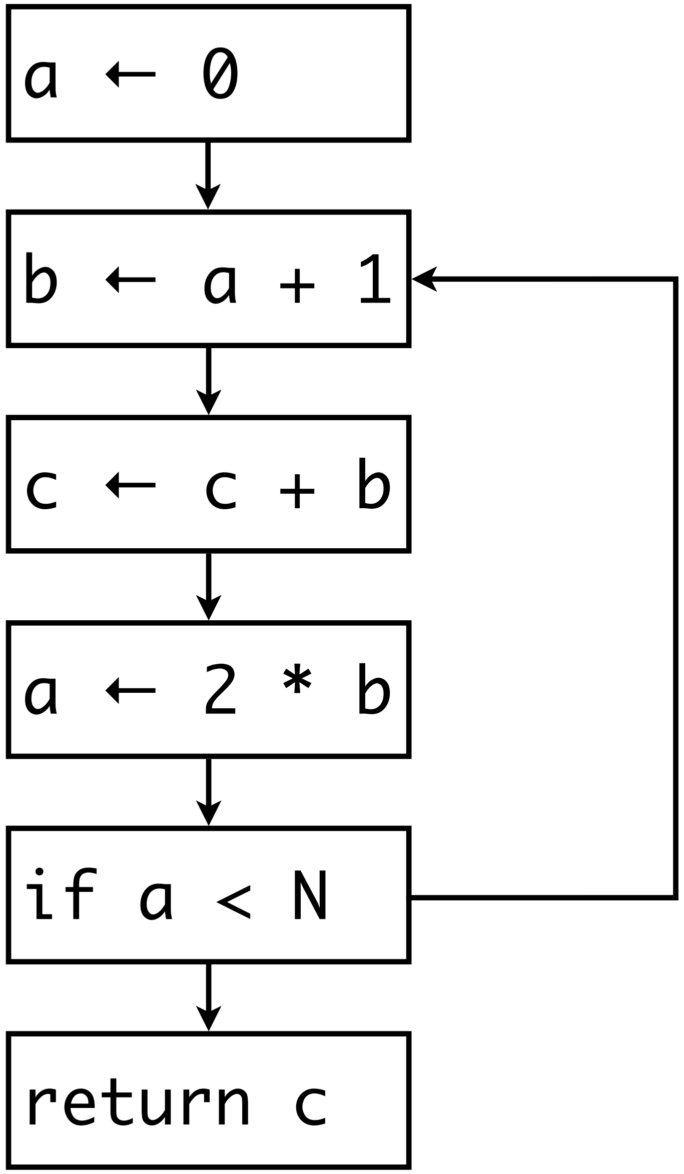
Control-flow graphs
example

```
    a ← 0
L1: b ← a + 1
    c ← c + b
    a ← 2 * b
    if a < N
       goto L1
    else
       goto L2
L2: return c
```

Control-flow graphs

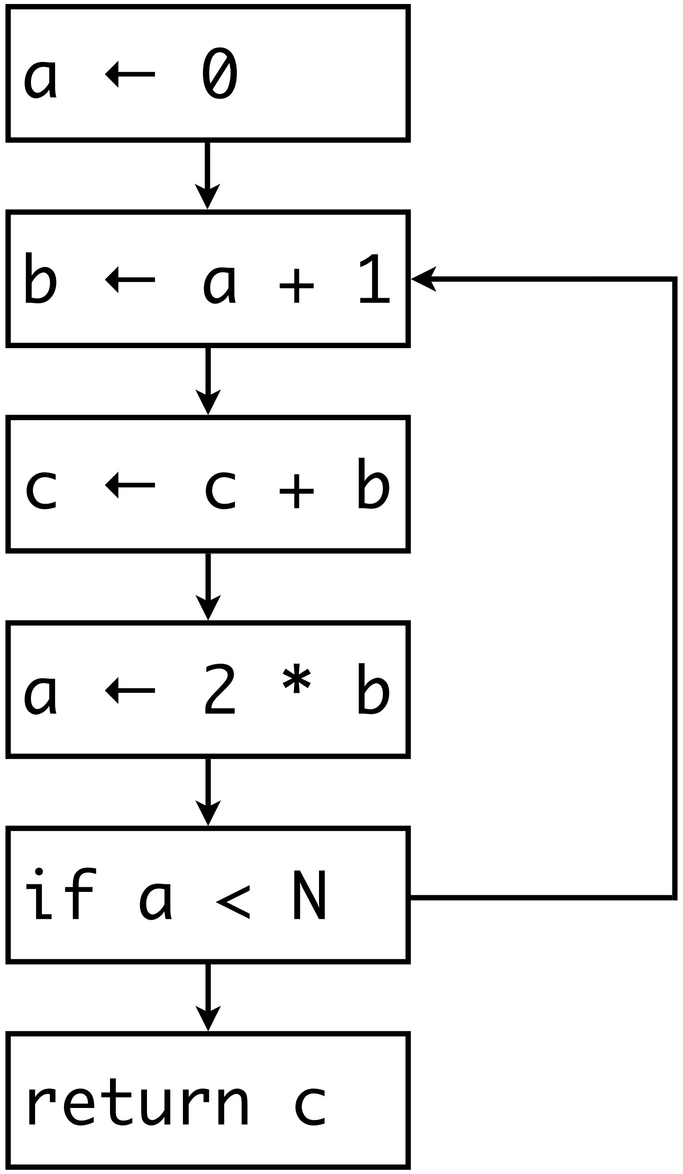
example

```
    a ← 0
L1: b ← a + 1
    c ← c + b
    a ← 2 * b
    if a < N
       goto L1
    else
       goto L2
L2: return c
```



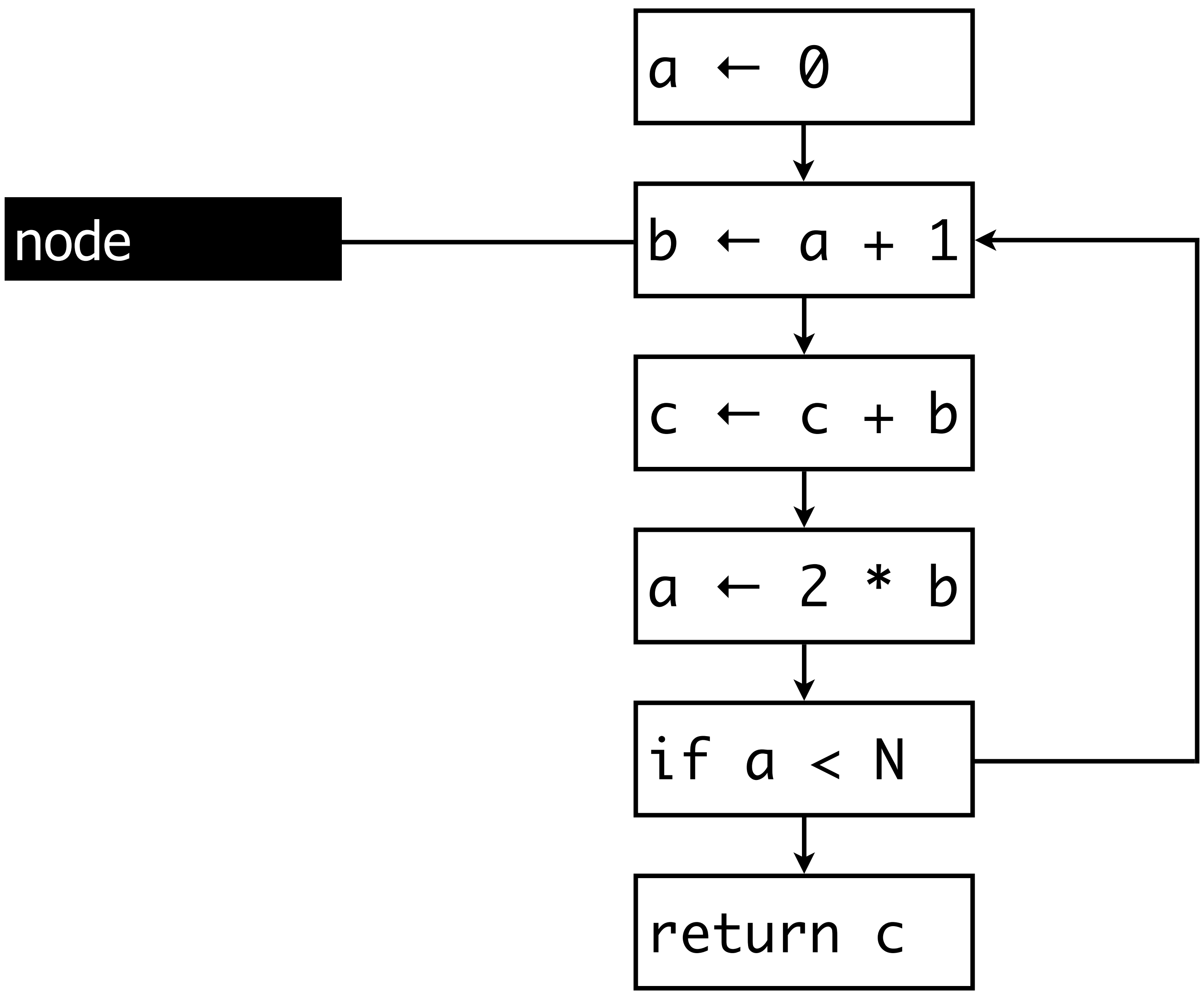
Control-flow graphs

terminology

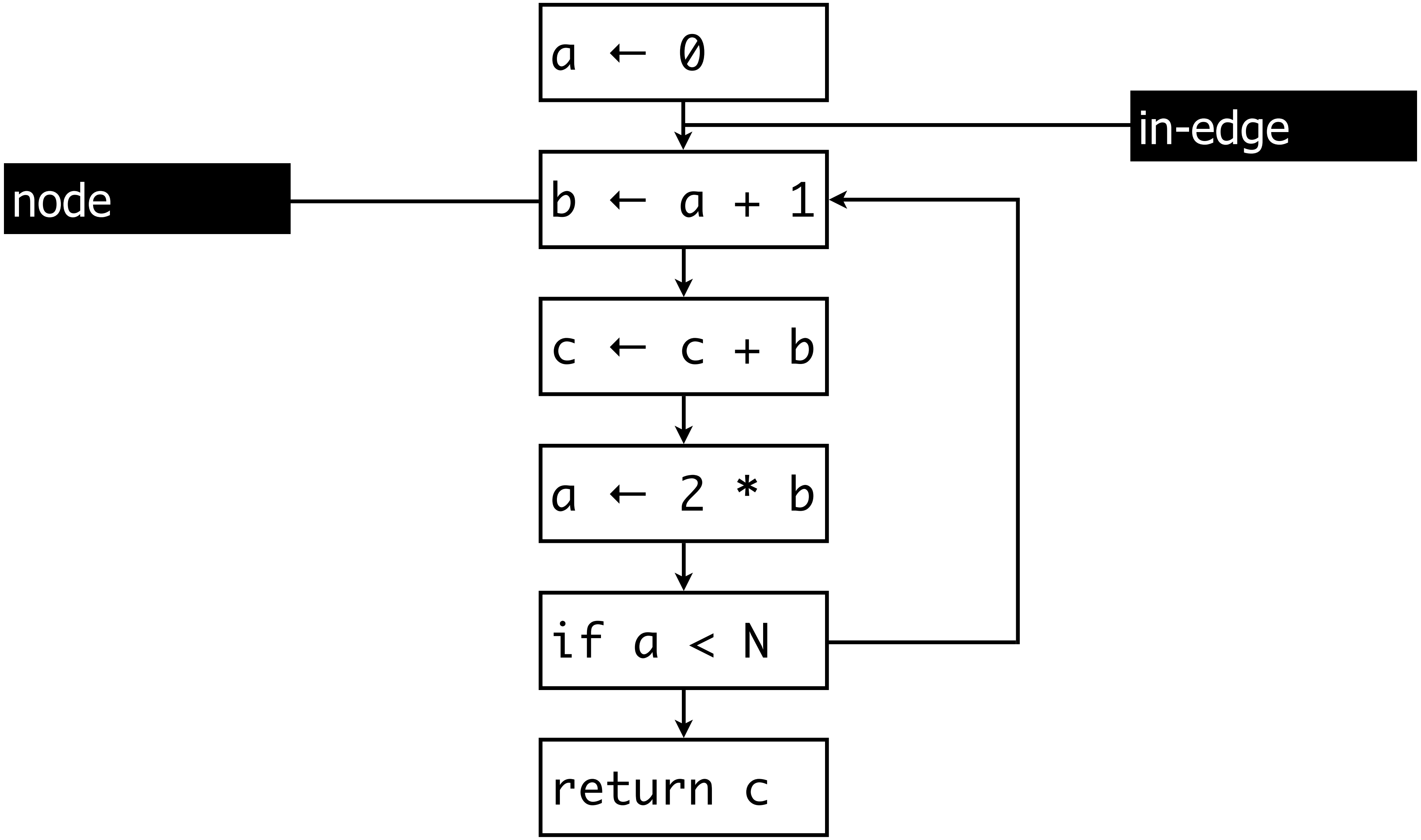


Control-flow graphs

terminology

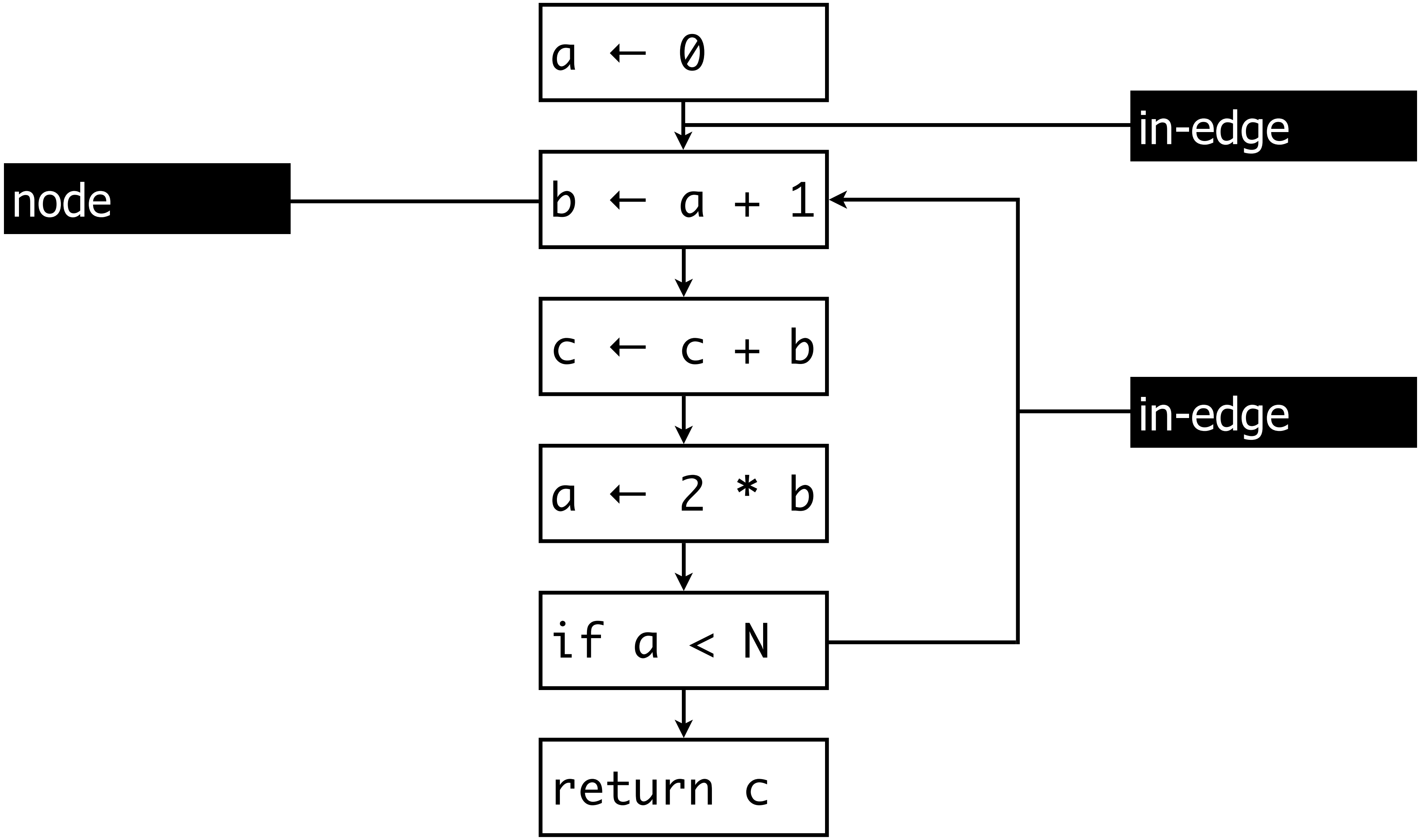


Control-flow graphs
terminology

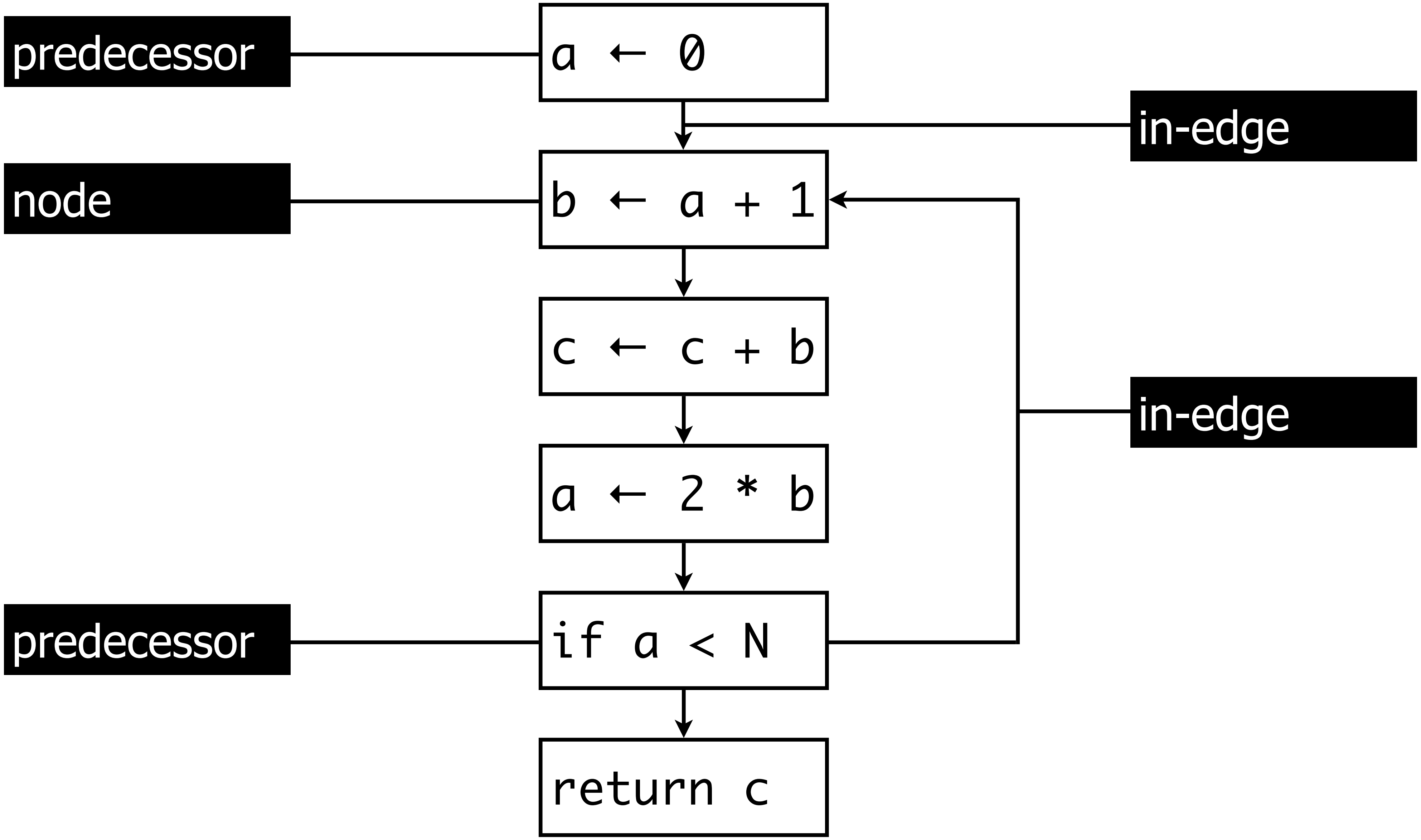


Control-flow graphs

terminology

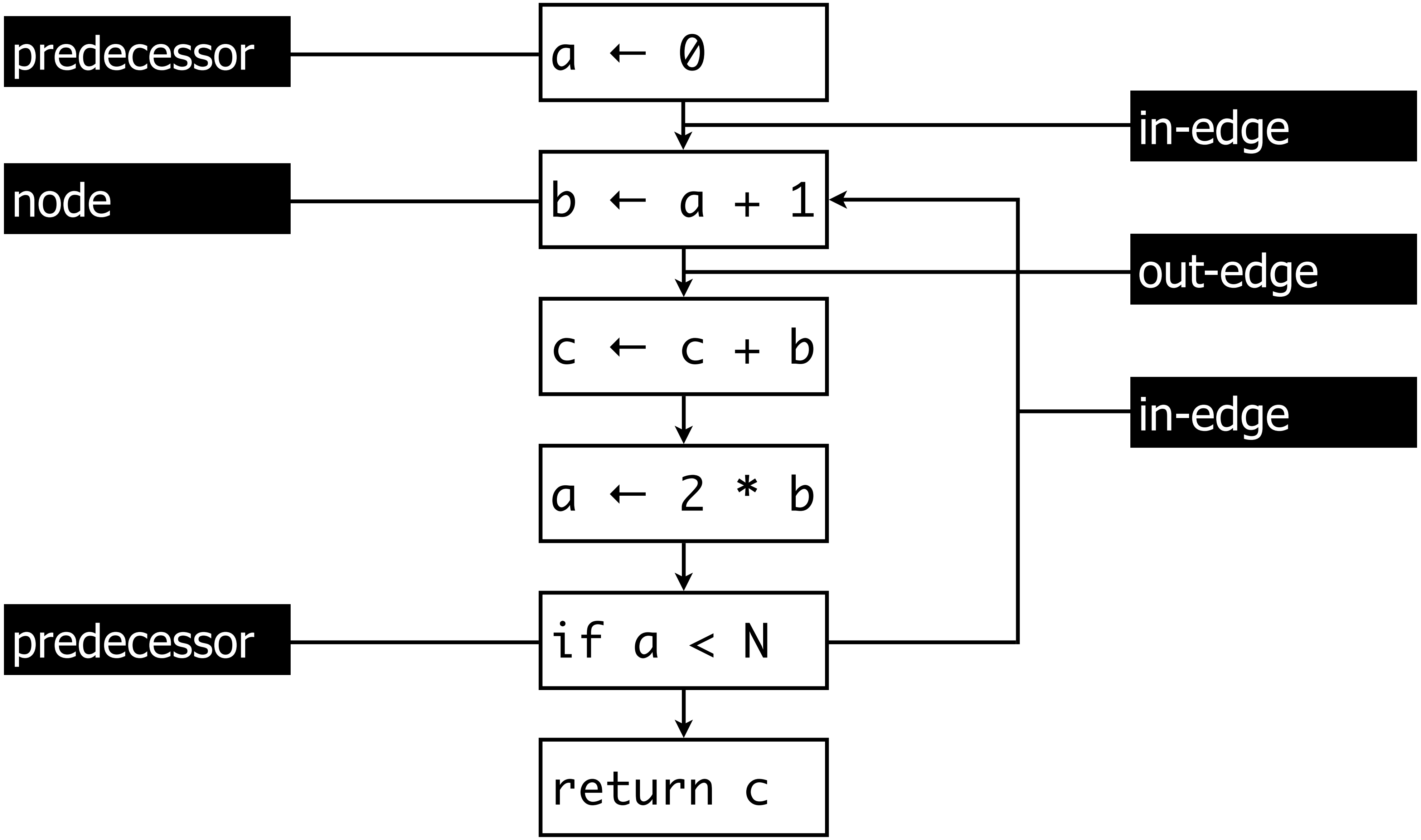


Control-flow graphs
terminology



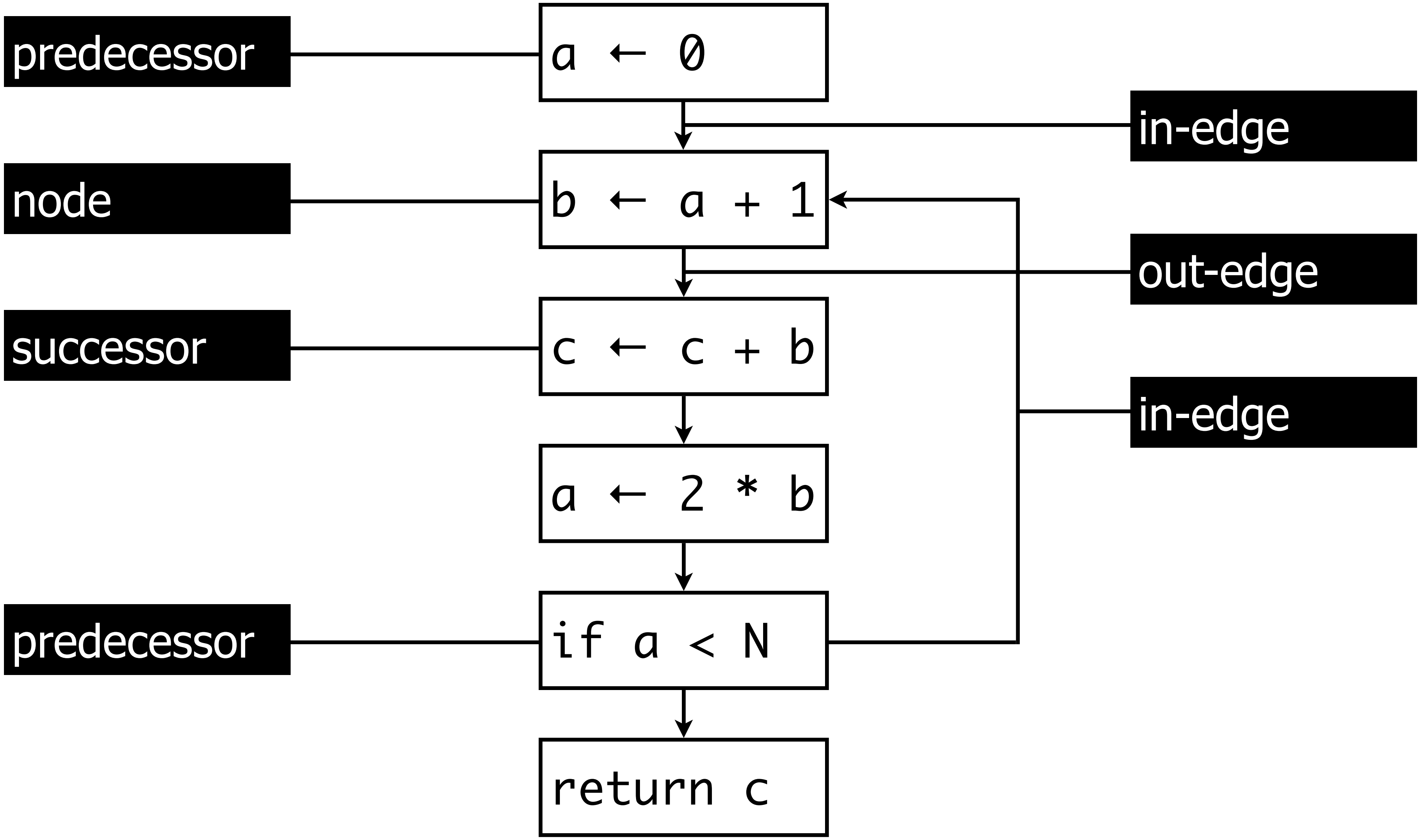
Control-flow graphs

terminology



Control-flow graphs

terminology

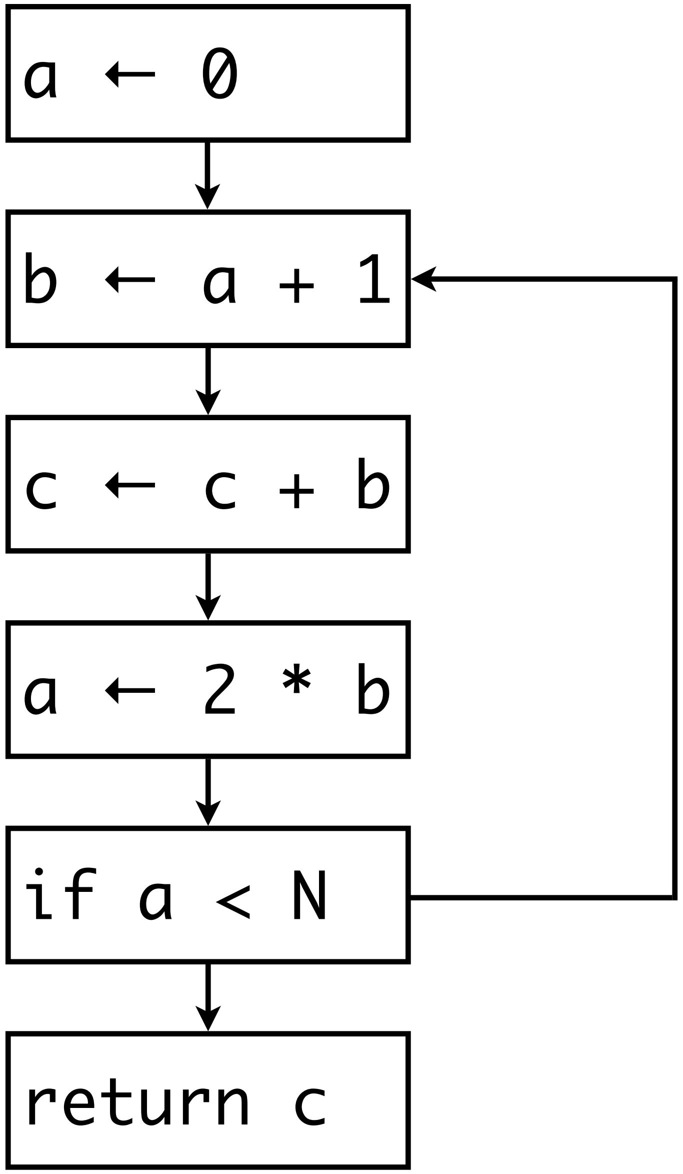


II

liveness analysis

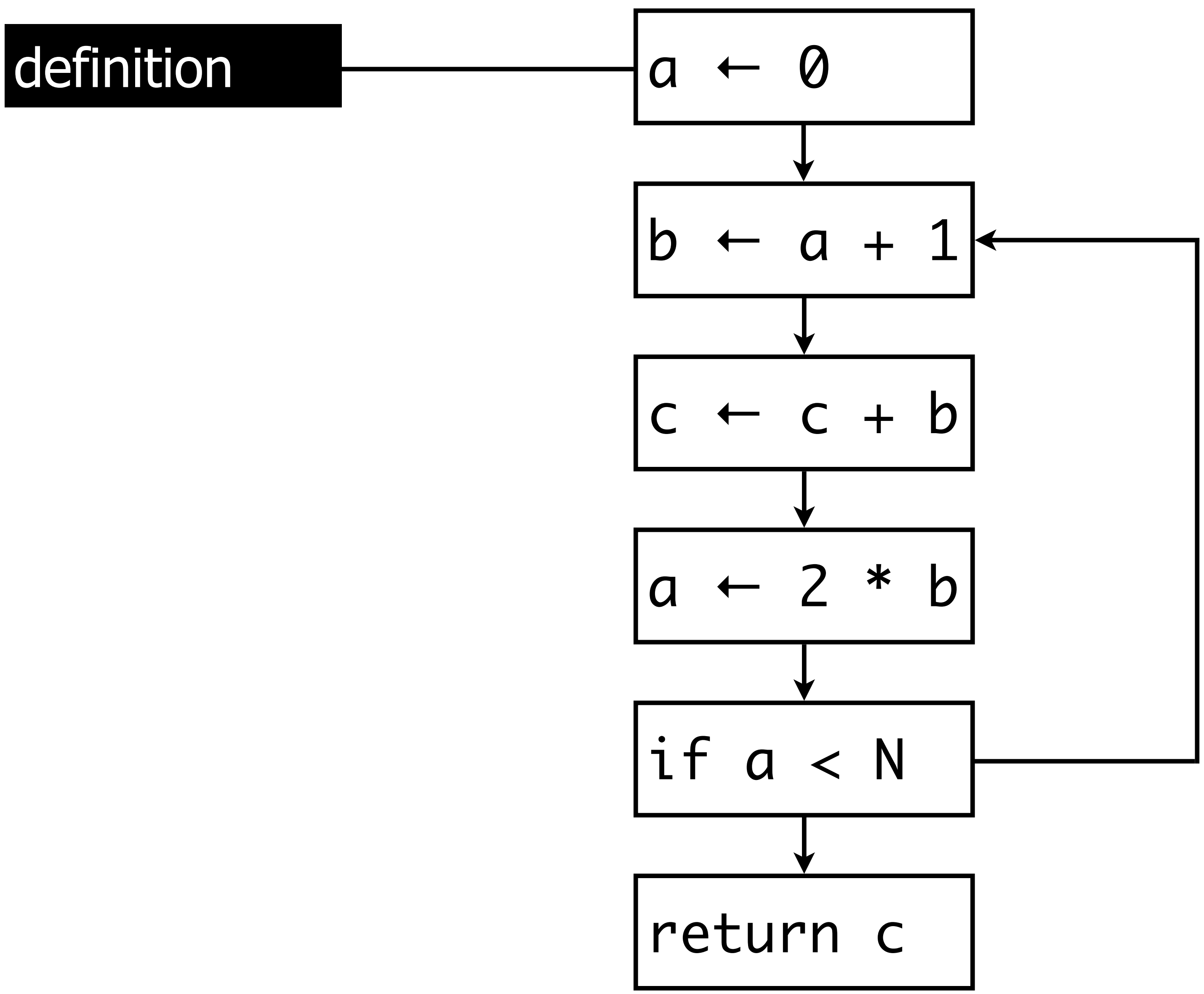
Liveness analysis

terminology



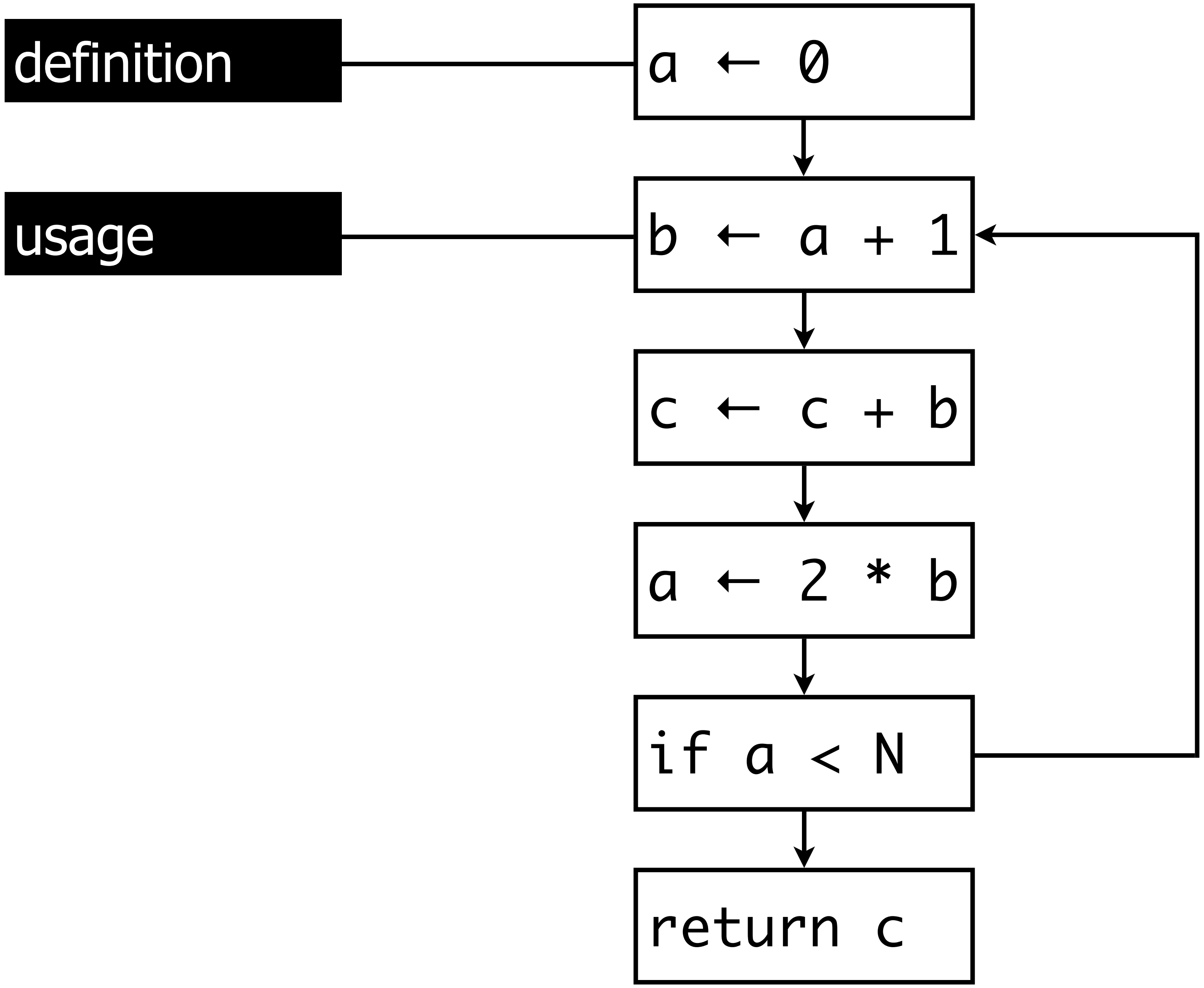
Liveness analysis

terminology

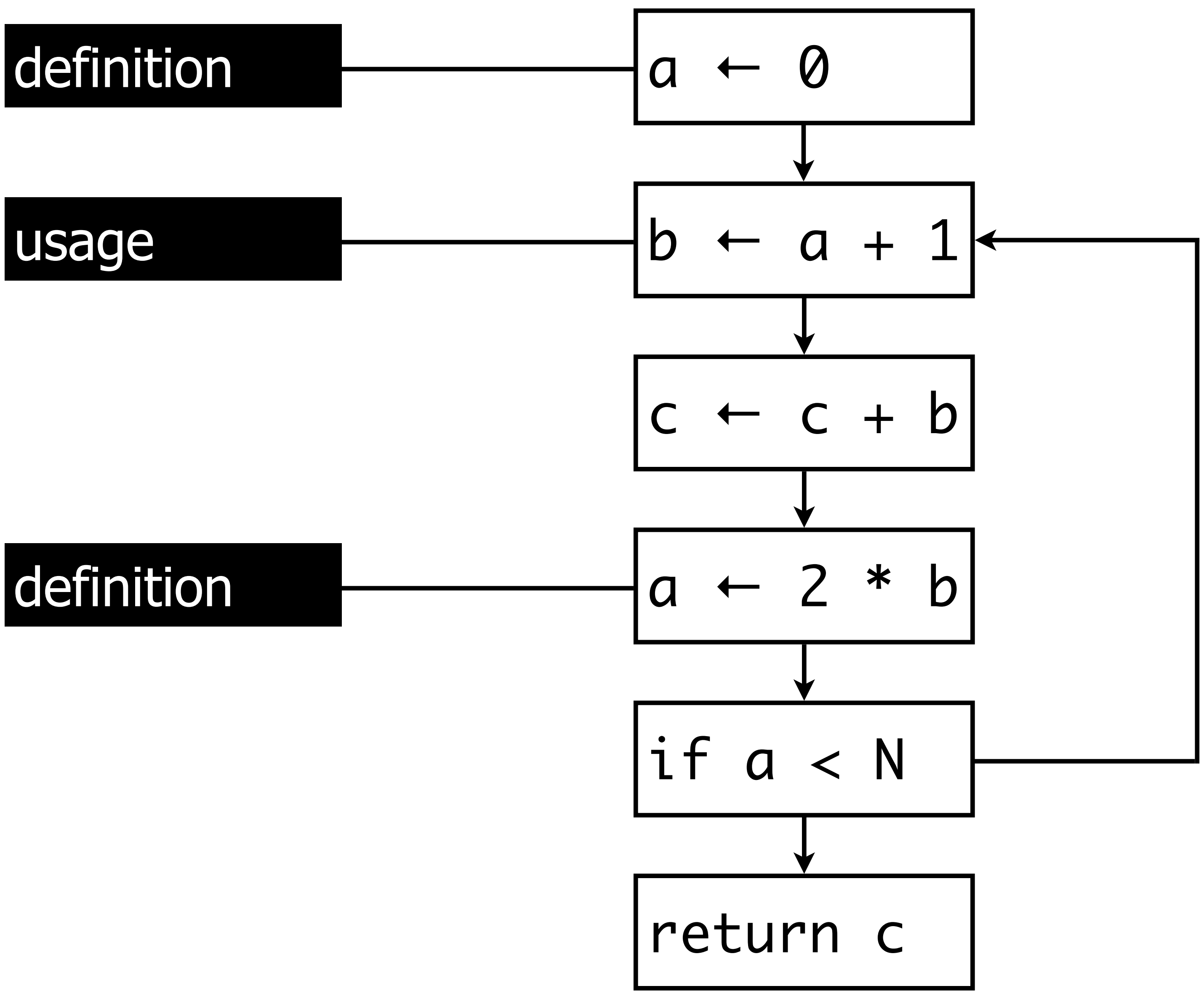


Liveness analysis

terminology

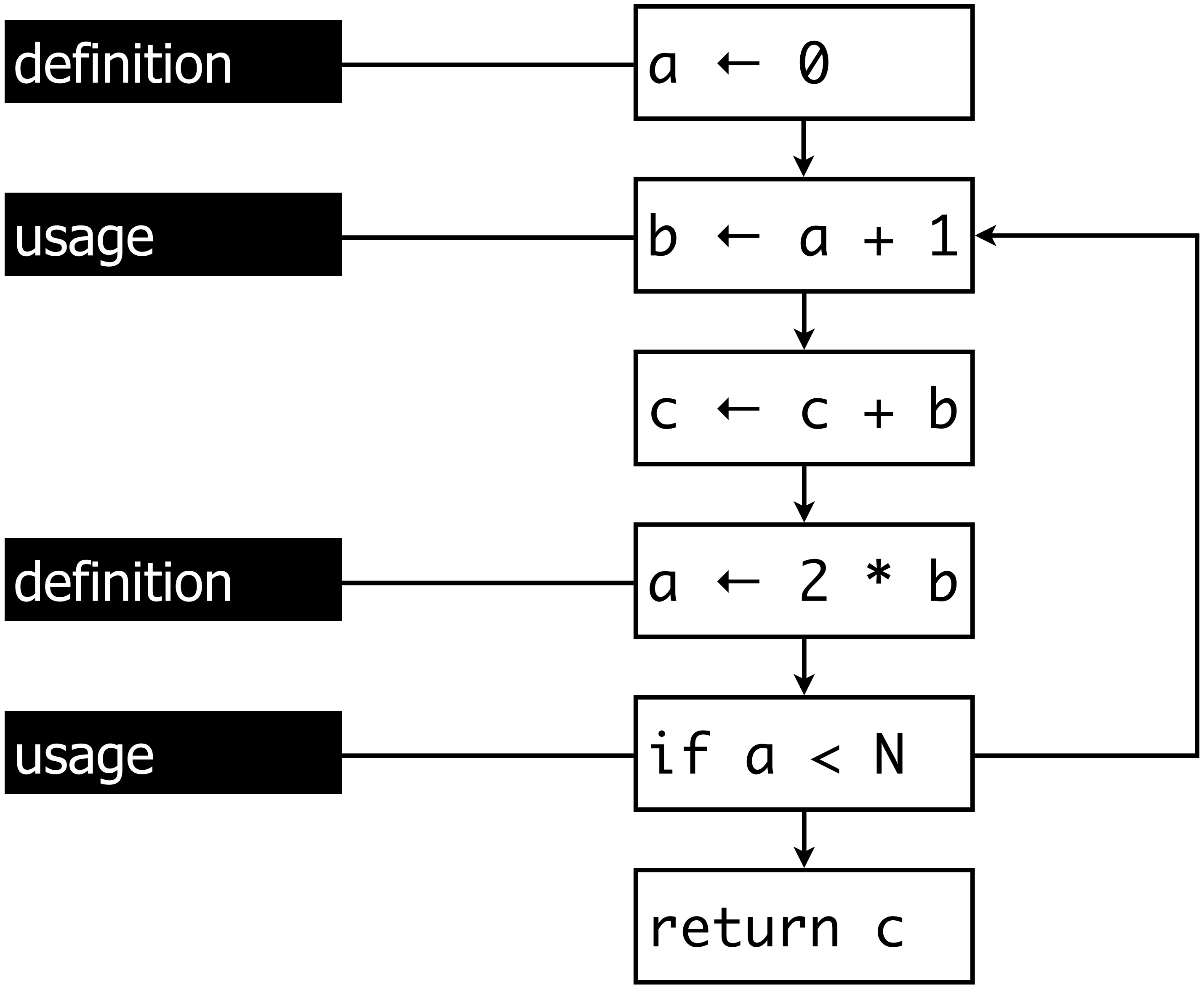


Liveness analysis
terminology



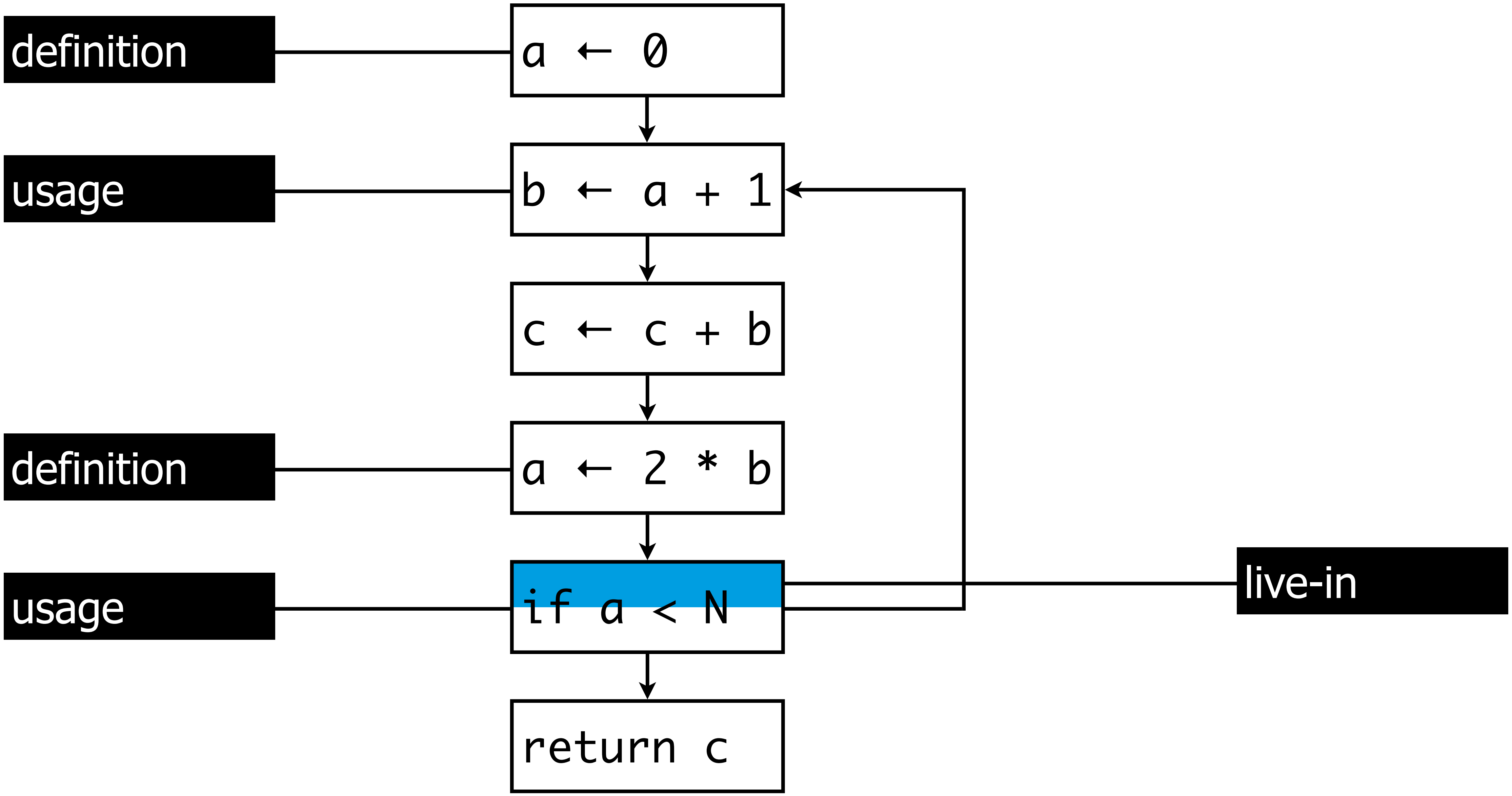
Liveness analysis

terminology



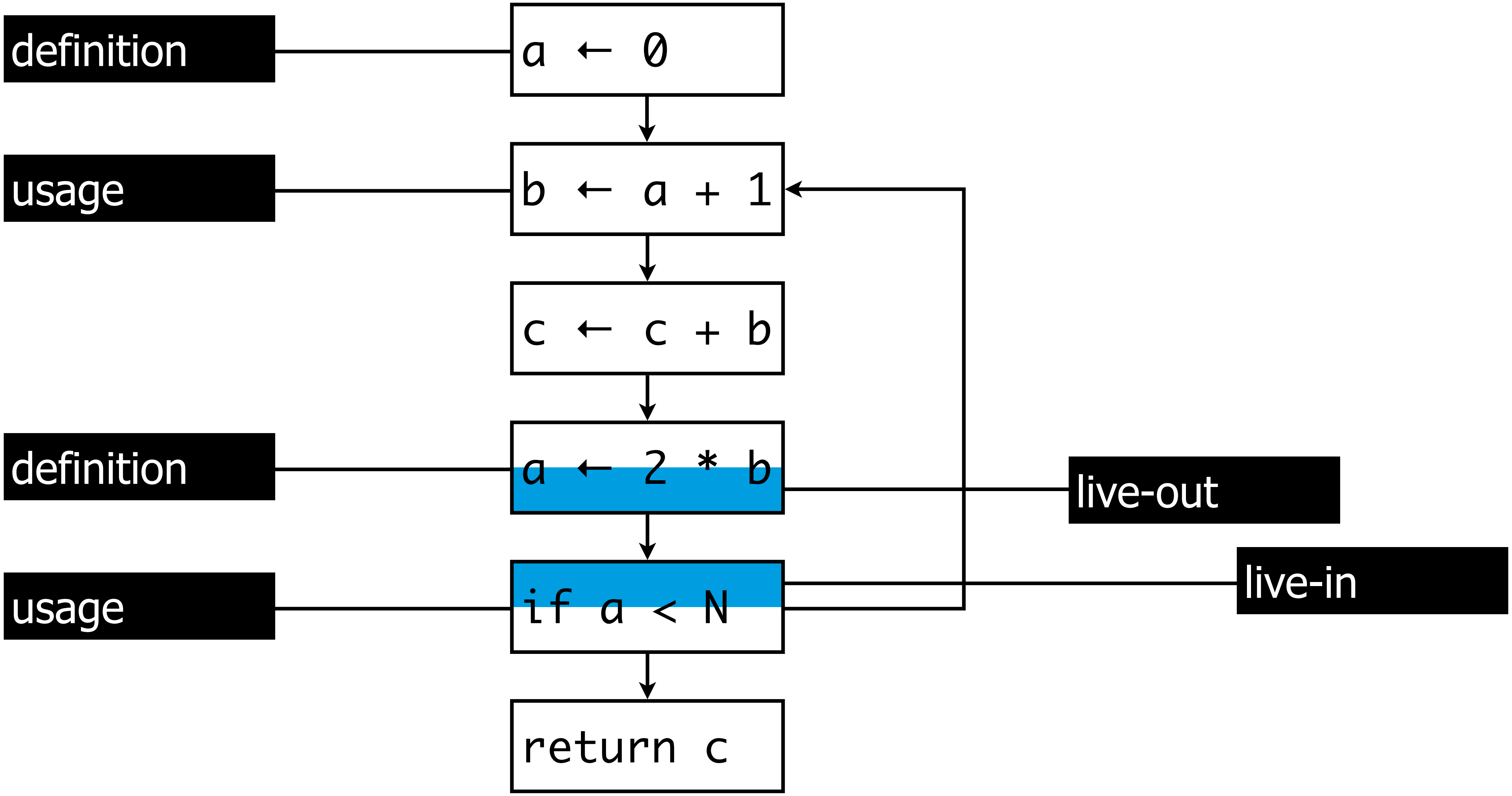
Liveness analysis

terminology



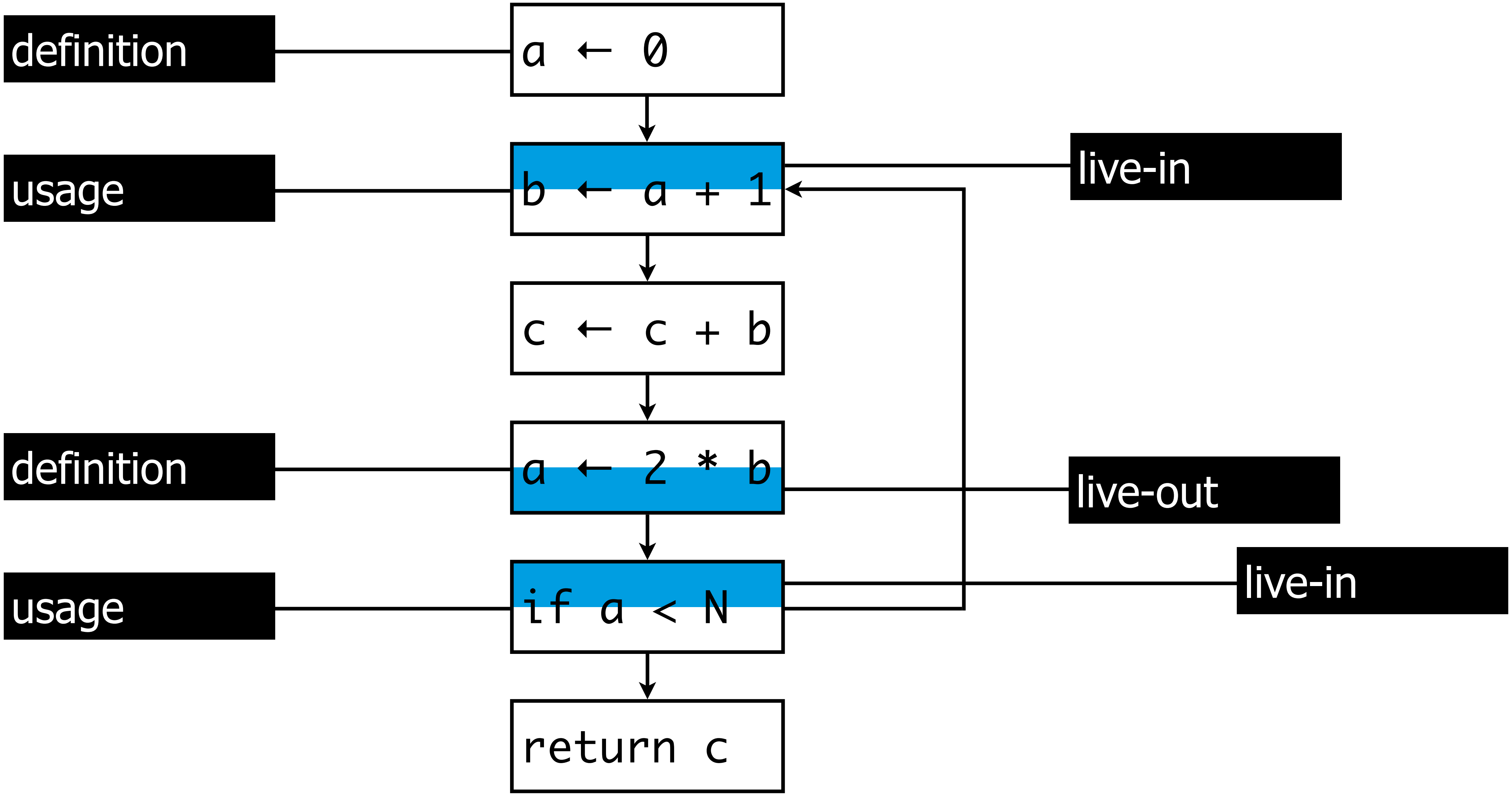
Liveness analysis

terminology

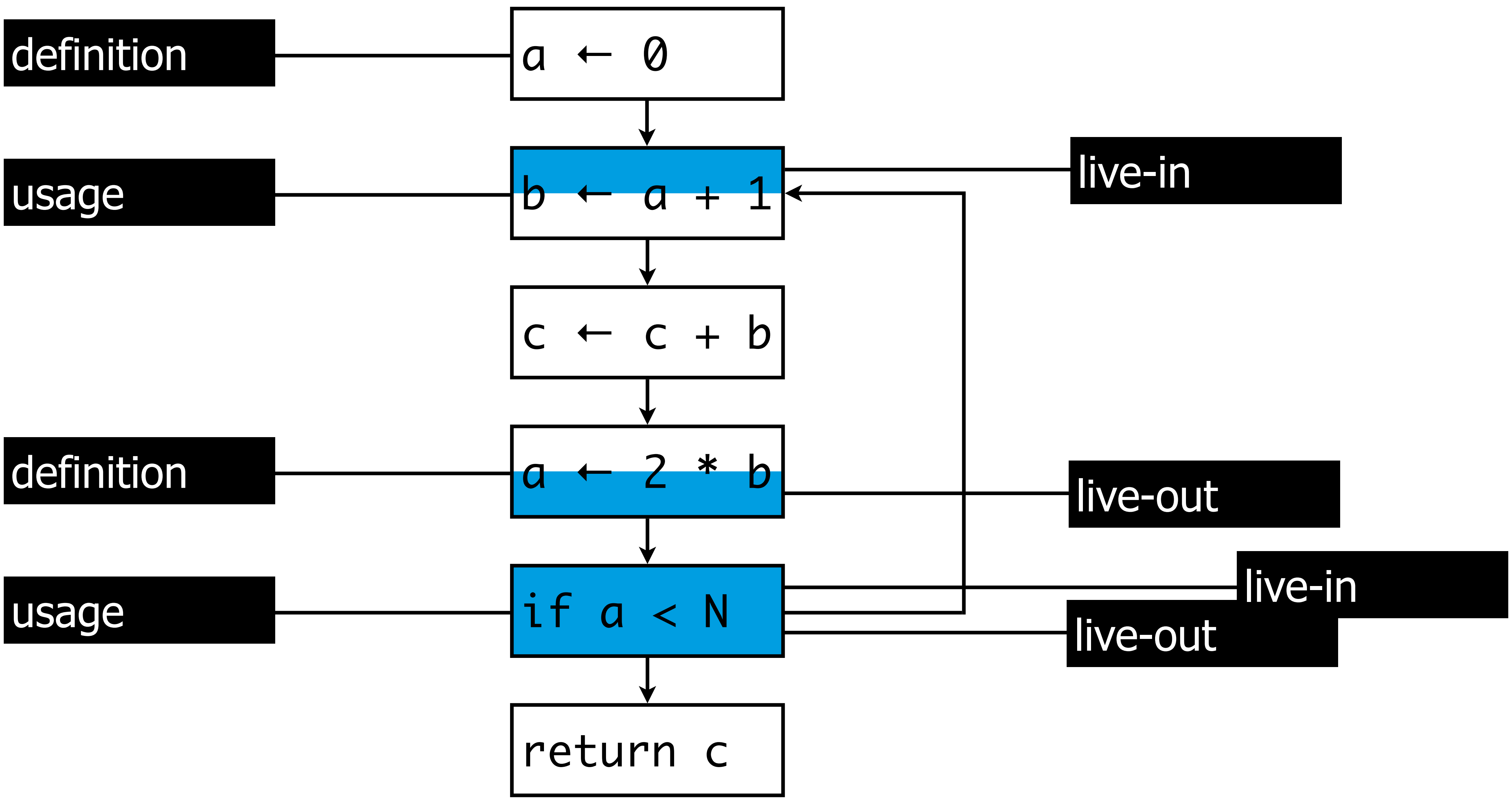


Liveness analysis

terminology

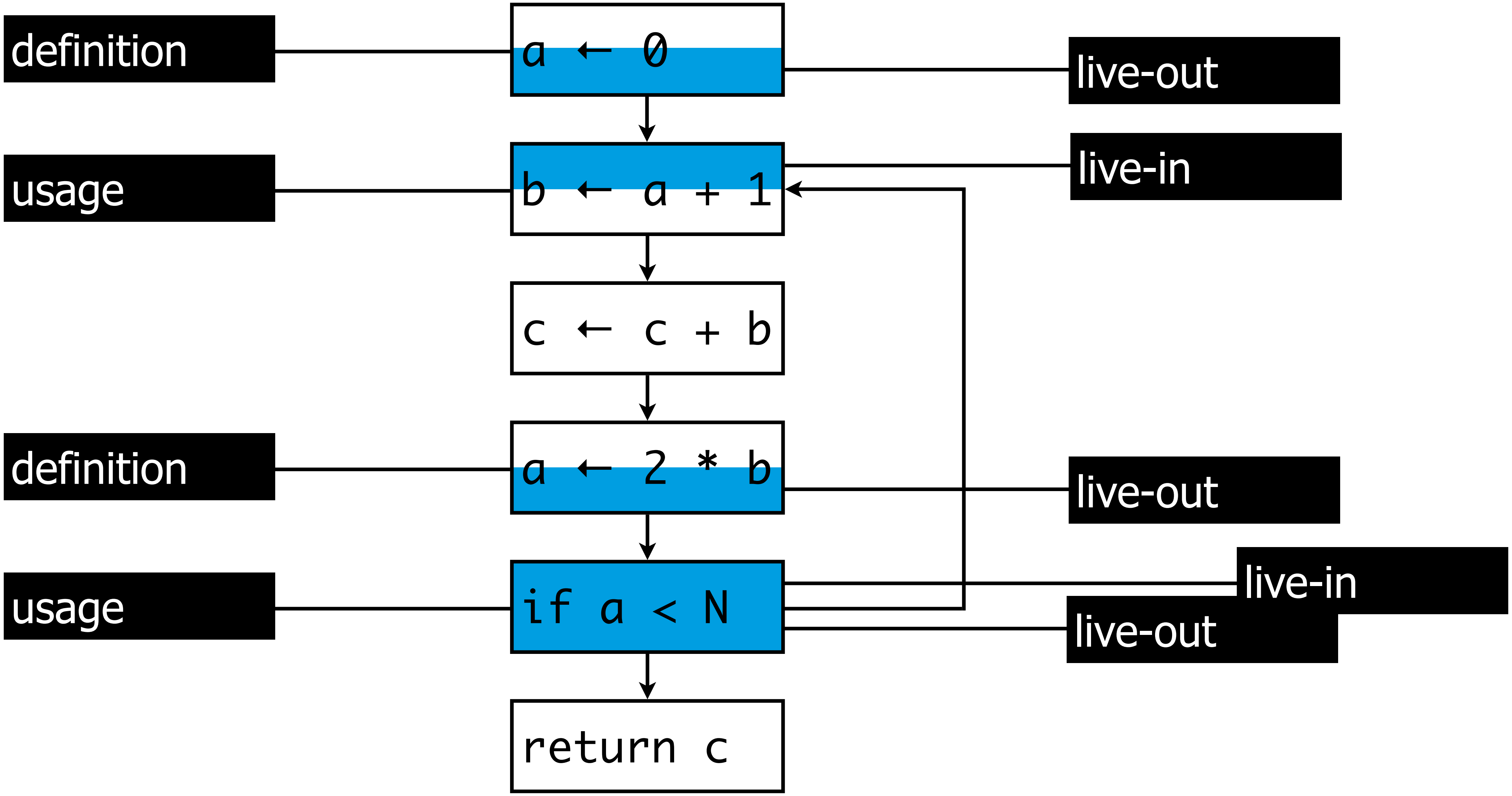


Liveness analysis
terminology



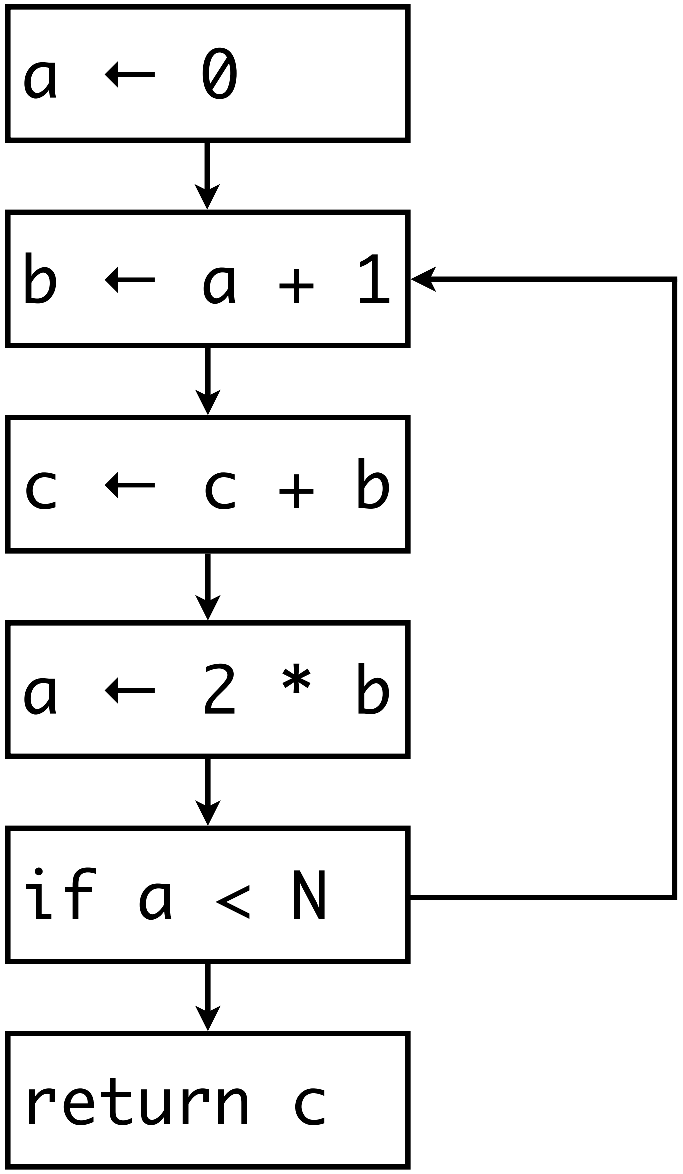
Liveness analysis

terminology



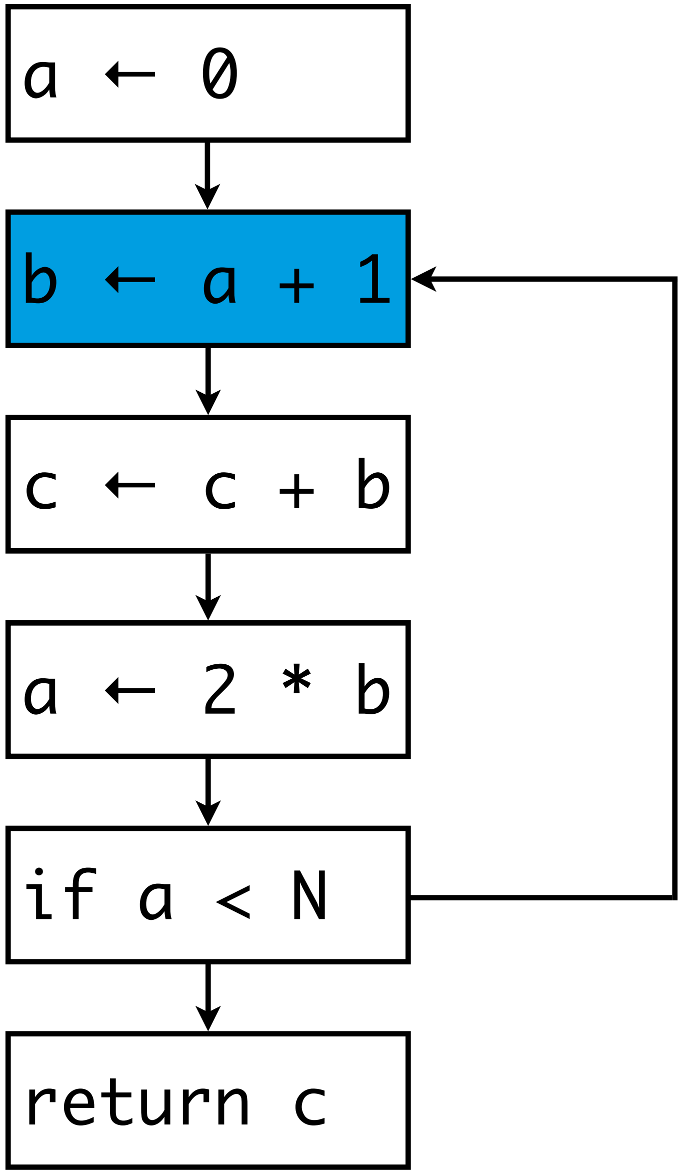
Liveness analysis

example



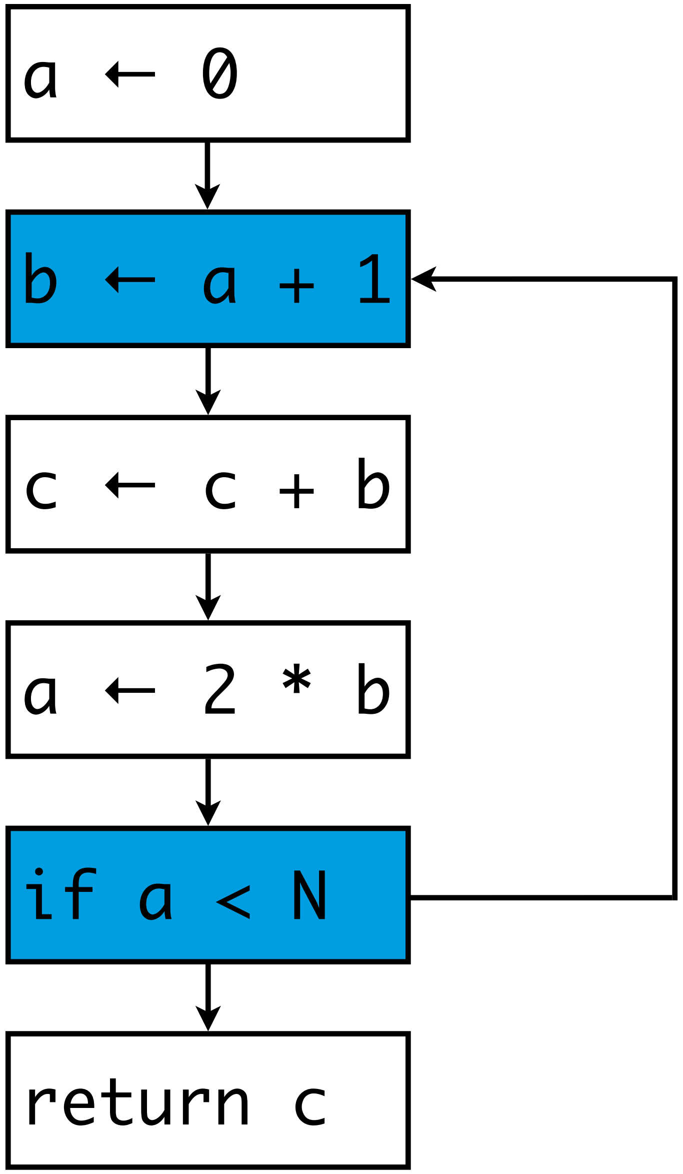
Liveness analysis

example



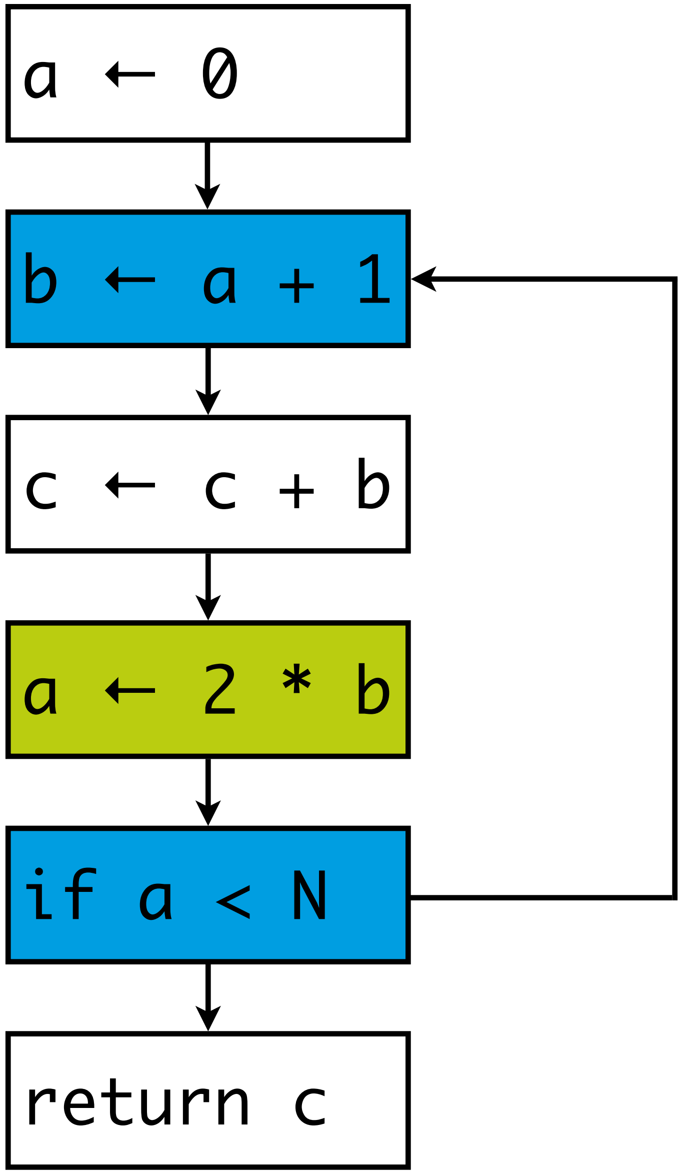
Liveness analysis

example



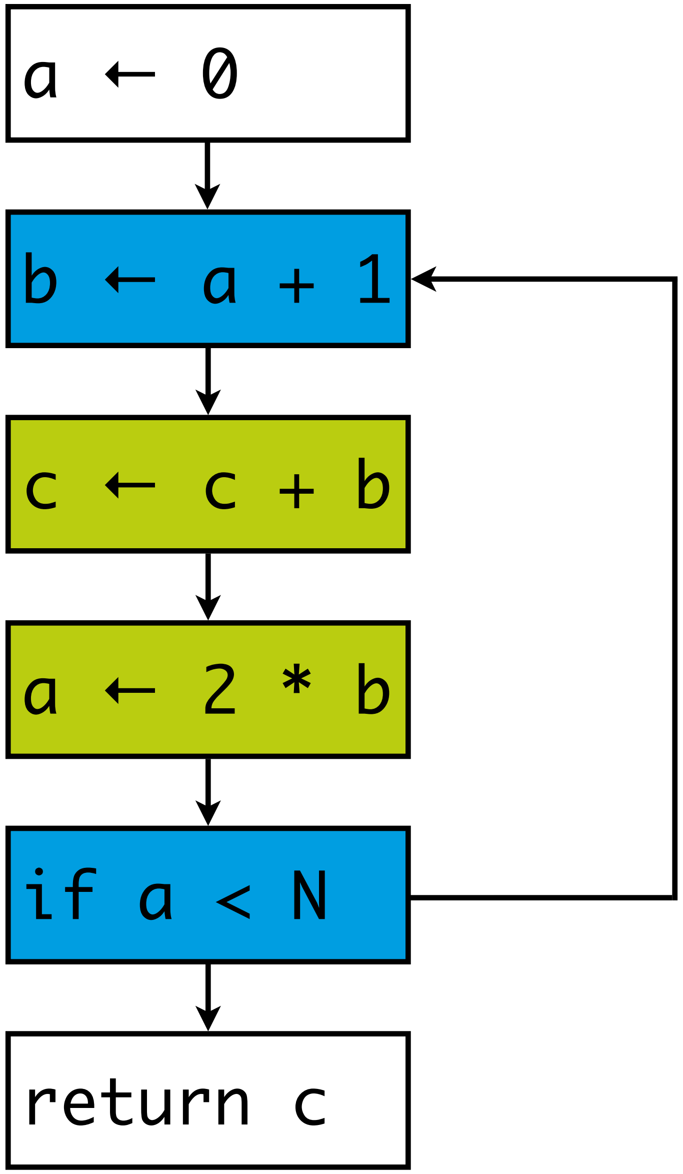
Liveness analysis

example

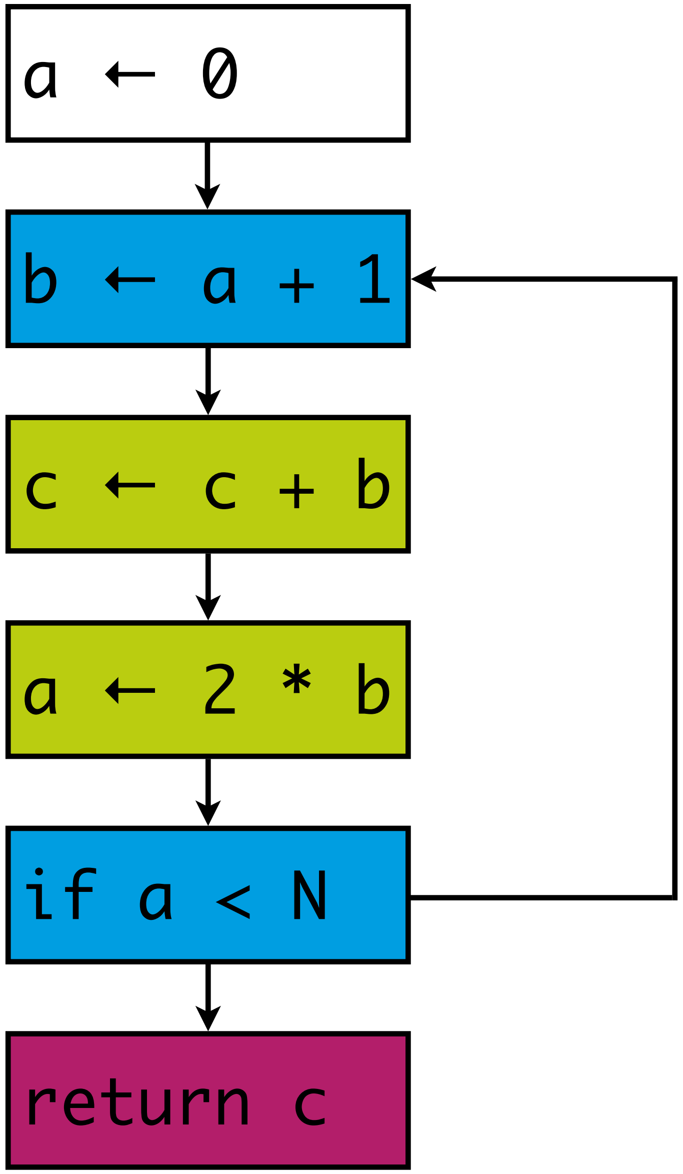


Liveness analysis

example

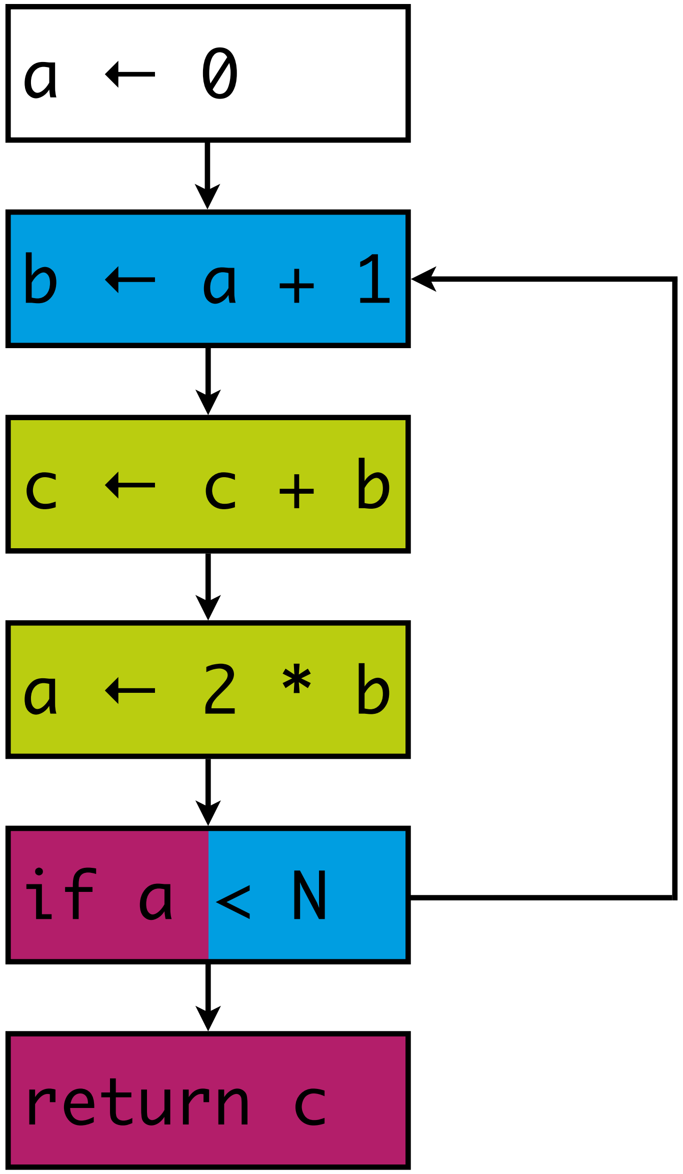


Liveness analysis
example



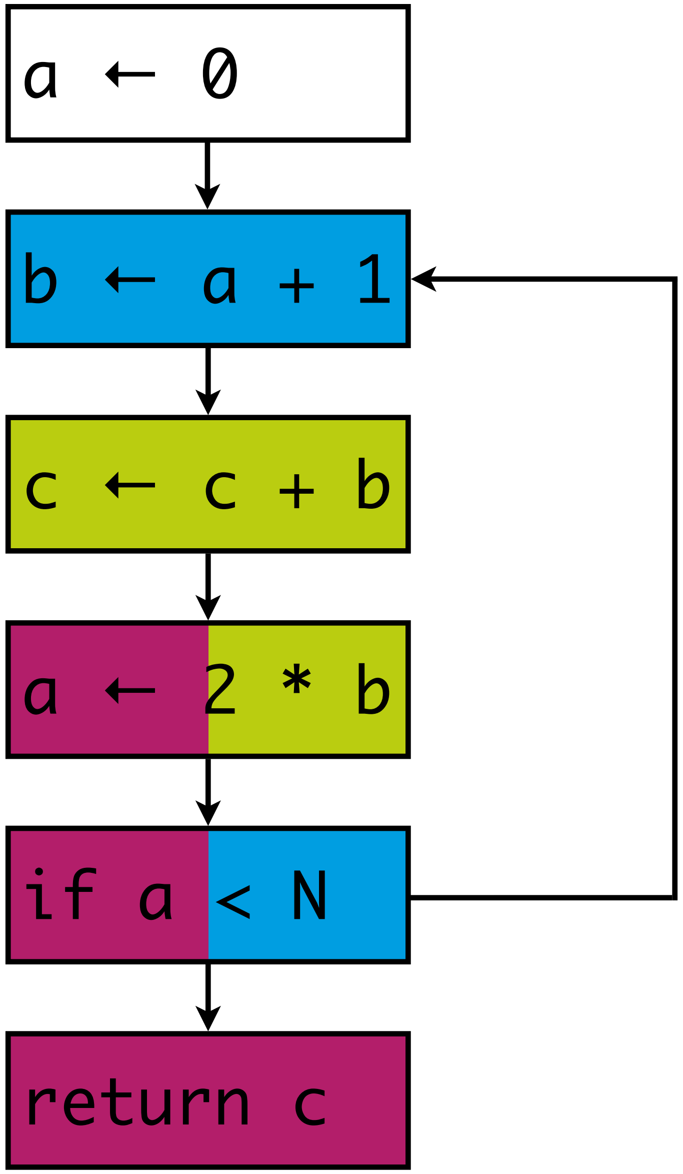
Liveness analysis

example

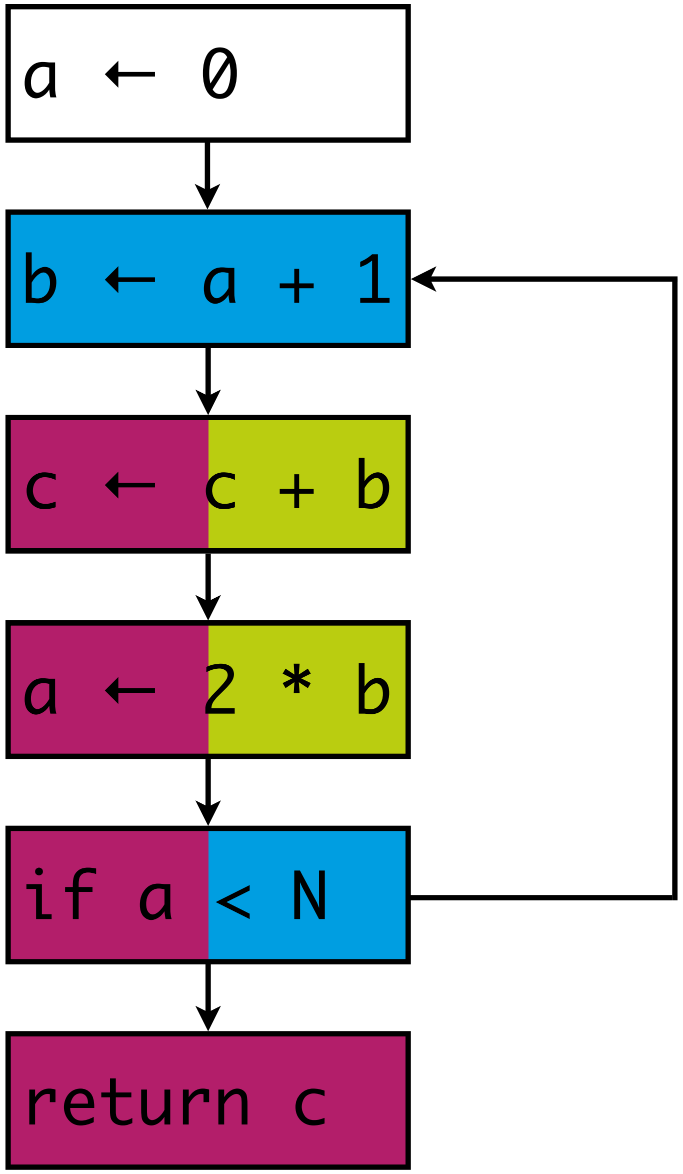


Liveness analysis

example

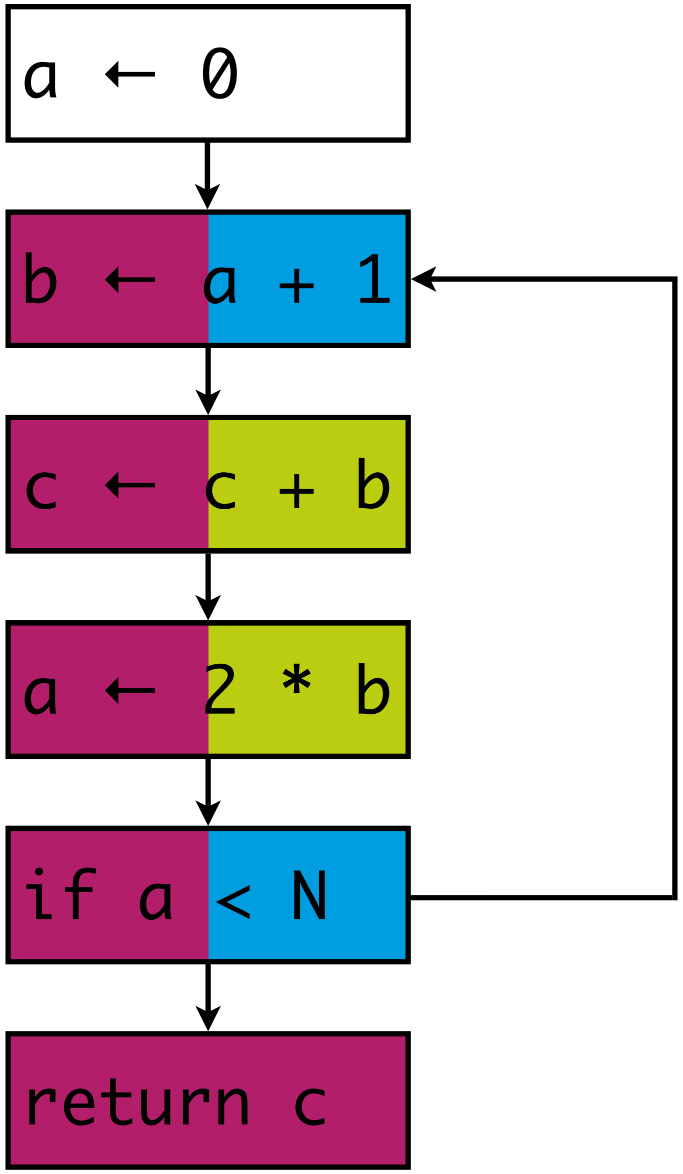


Liveness analysis
example



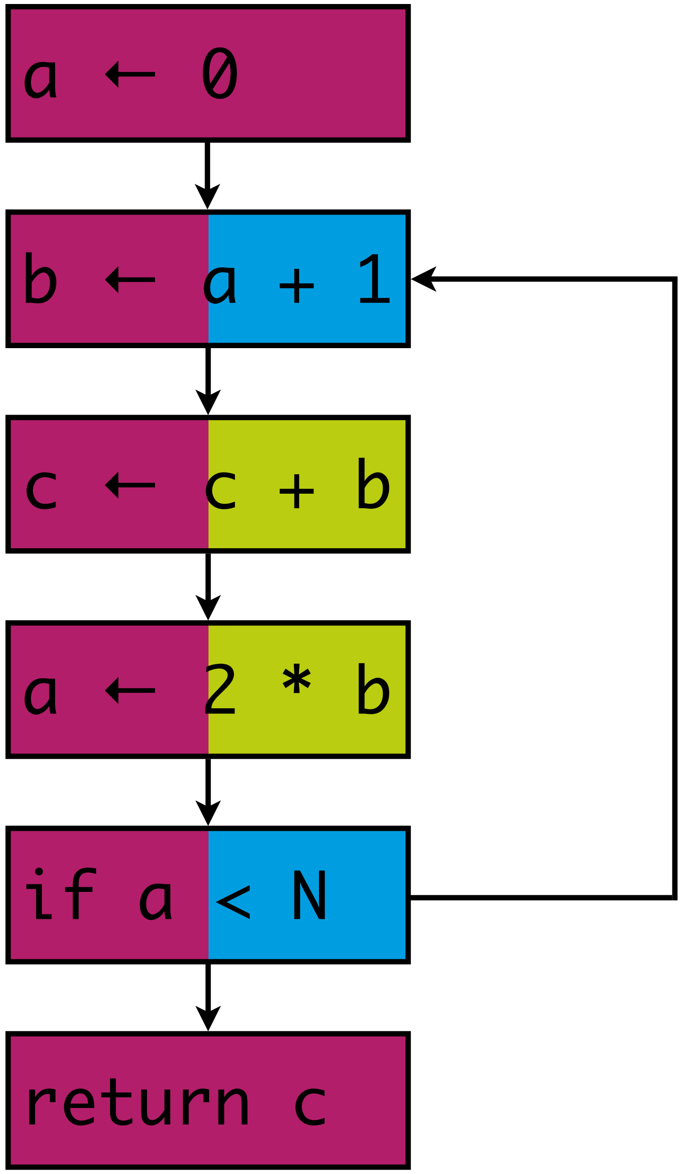
Liveness analysis

example



Liveness analysis

example



Liveness analysis
formalisation

$$in[n] = use[n] \cup \left(out[n] - def[n]\right)$$

$$out[n] = \bigcup_{s \in succ[n]} in[s]$$

1. If a variable is used at node n, then it is live-in at n.
2. If a variable is live-out but not defined at node n, it is live-in at n.
3. If a variable is live-in at node n, then it is live-out at its predecessors.

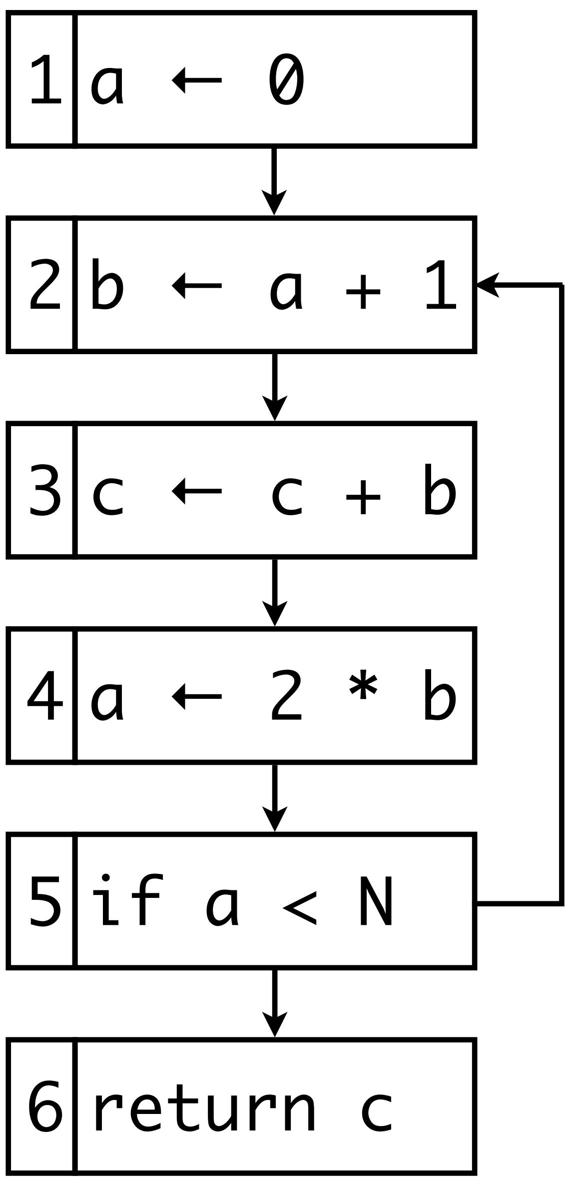
Liveness analysis

algorithm

```
for each n
    in[n] ← {} ; out[n] ← {}
repeat
    for each n
        in'[n] = in[n] ; out'[n] = out[n]
        in[n] = use[n] ∪ (out[n] - def[n])
        for each s in succ[n]
            out[n] = out[n] ∪ in[s]
until
    for all n: in[n] = in'[n] and out[n] = out'[n]
```

Liveness analysis

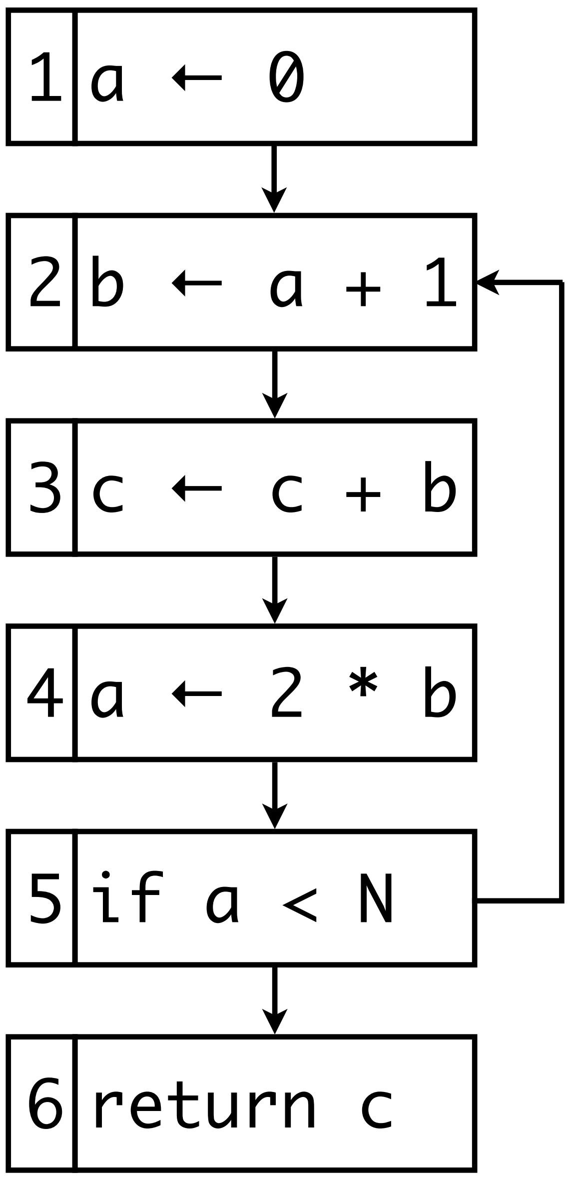
example



	u
1	
2	a
3	bc
4	b
5	a
6	c

Liveness analysis

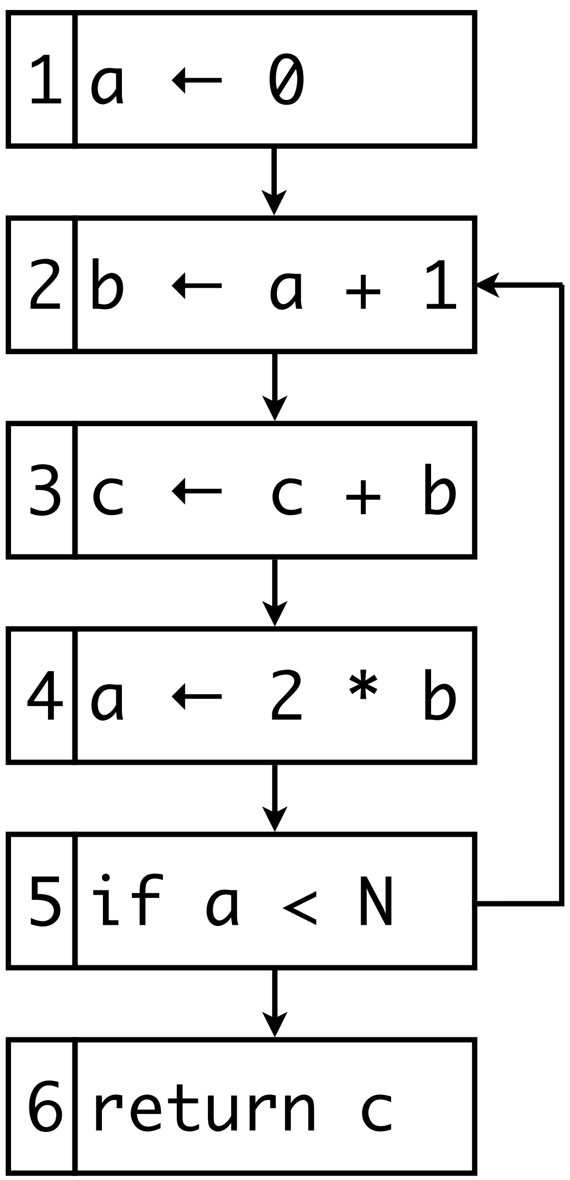
example



	u	d
1		a
2	a	b
3	bc	c
4	b	a
5	a	
6	c	

Liveness analysis

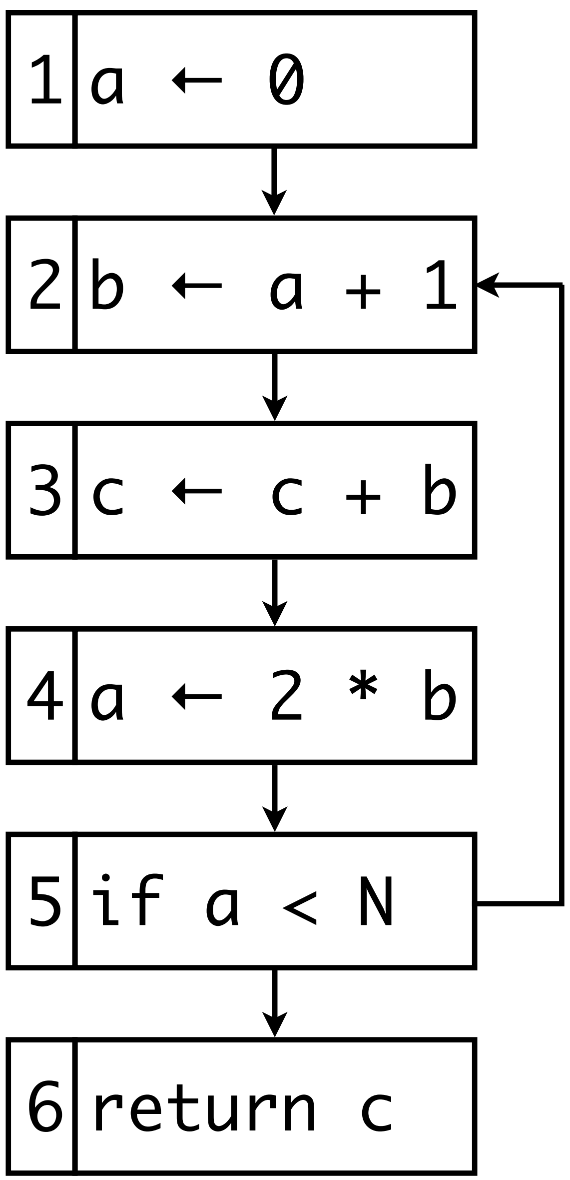
example



			1	
	u	d	i	o
1		a		
2	a	b	a	
3	bc	c	bc	
4	b	a	b	
5	a		a	a
6	c		c	

Liveness analysis

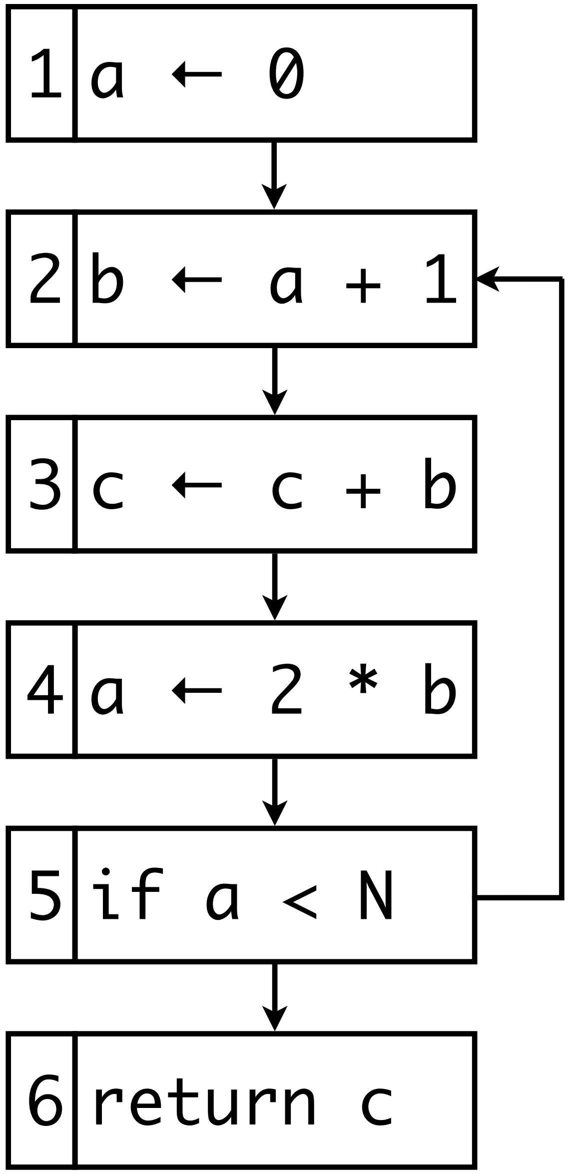
example



			1		2	
	u	d	i	o	i	o
1		a				a
2	a	b	a		a	bc
3	bc	c	bc		bc	b
4	b	a	b		b	a
5	a		a	a	a	ac
6	c		c		c	

Liveness analysis

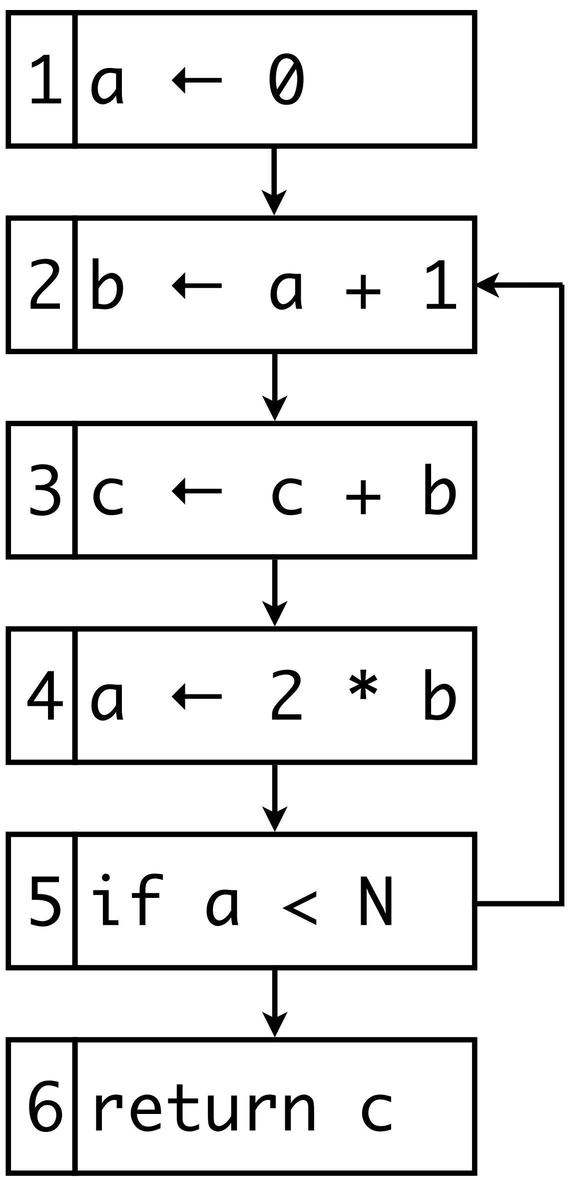
example



			1		2		3									
	u	d	i	o	i	o	i	o								
1		a				a		a								
2	a	b	a		a	bc	ac	bc								
3	bc	c	bc		bc	b	bc	b								
4	b	a	b		b	a	b	a								
5	a		a	a	a	ac	ac	ac								
6	c		c		c		c									

Liveness analysis

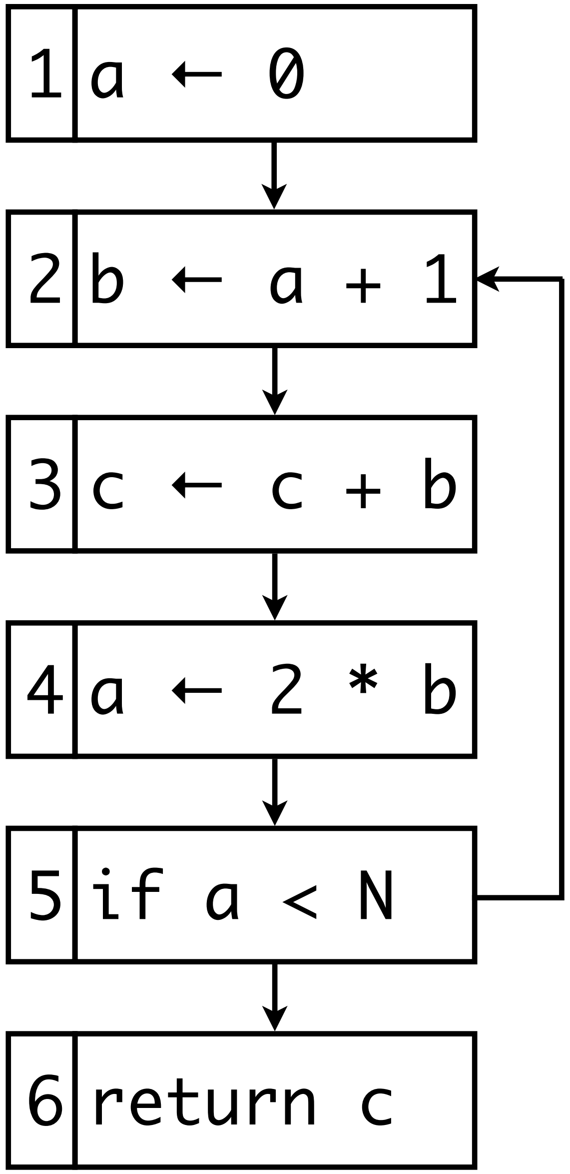
example



			1		2		3		4							
	u	d	i	o	i	o	i	o	i	o						
1		a				a		a		ac						
2	a	b	a		a	bc	ac	bc	ac	bc						
3	bc	c	bc		bc	b	bc	b	bc	b						
4	b	a	b		b	a	b	a	b	ac						
5	a		a	a	a	ac	ac	ac	ac	ac						
6	c		c		c		c		c							

Liveness analysis

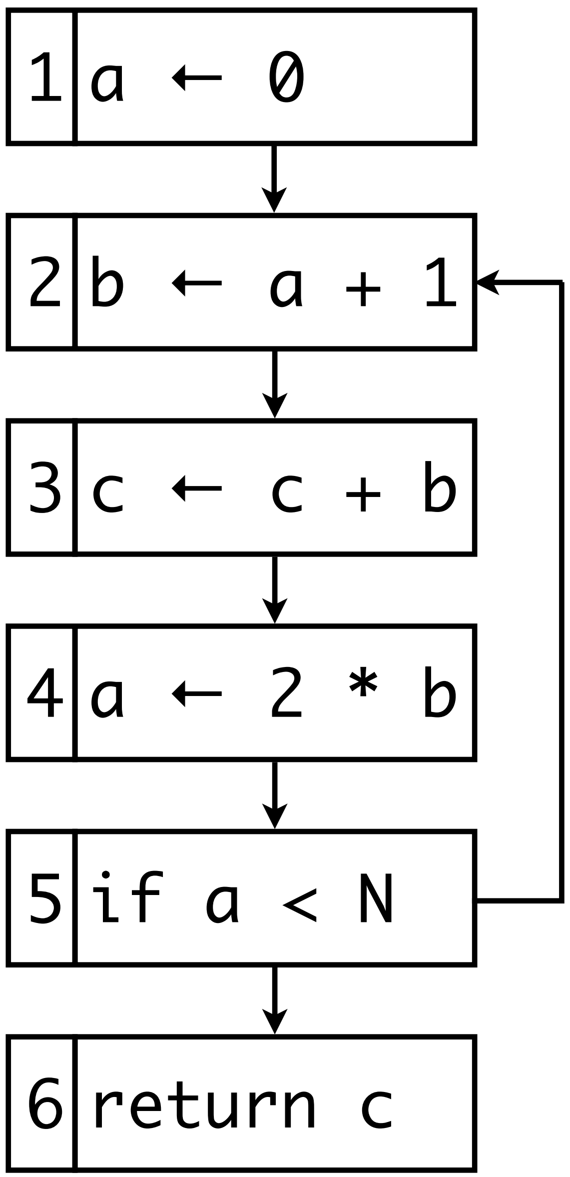
example



			1		2		3		4		5	
	u	d	i	o	i	o	i	o	i	o	i	o
1		a				a		a		ac	c	ac
2	a	b	a		a	bc	ac	bc	ac	bc	ac	bc
3	bc	c	bc		bc	b	bc	b	bc	b	bc	b
4	b	a	b		b	a	b	a	b	ac	bc	ac
5	a		a	a	a	ac	ac	ac	ac	ac	ac	ac
6	c		c		c		c		c		c	

Liveness analysis

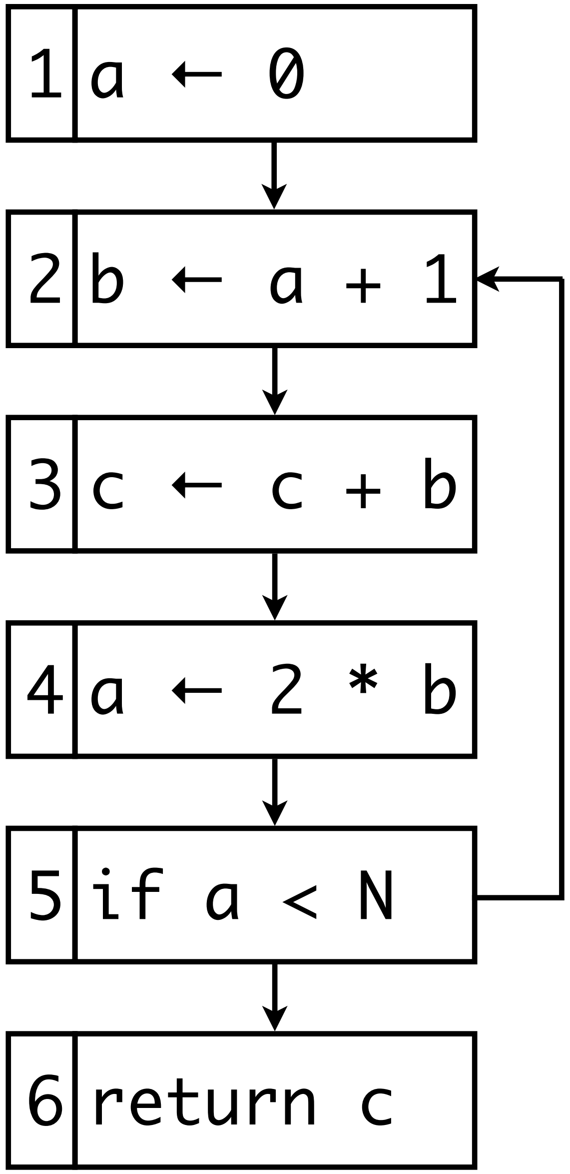
example



			1		2		3		4		5		6	
	u	d	i	o	i	o	i	o	i	o	i	o	i	o
1		a				a		a		ac	c	ac	c	ac
2	a	b	a		a	bc	ac	bc	ac	bc	ac	bc	ac	bc
3	bc	c	bc		bc	b	bc	b	bc	b	bc	b	bc	bc
4	b	a	b		b	a	b	a	b	ac	bc	ac	bc	ac
5	a		a	a	a	ac	ac	ac	ac	ac	ac	ac	ac	ac
6	c		c		c		c		c		c		c	

Liveness analysis

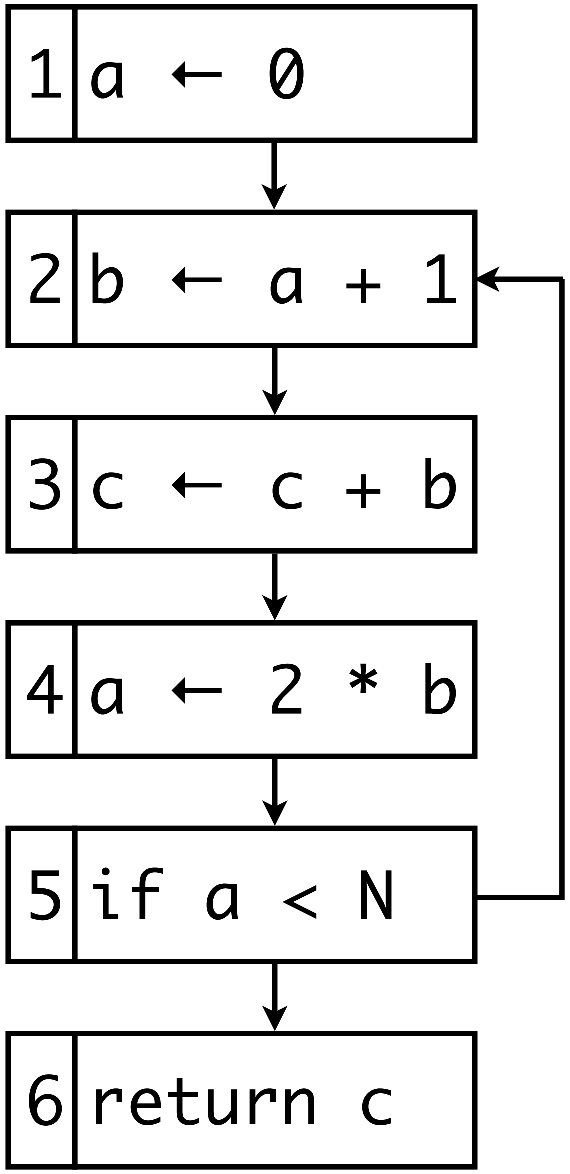
example



			1		2		3		4		5		6		7	
	u	d	i	o	i	o	i	o	i	o	i	o	i	o	i	o
1		a				a		a		ac	c	ac	c	ac	c	ac
2	a	b	a		a	bc	ac	bc	ac	bc	ac	bc	ac	bc	ac	bc
3	bc	c	bc		bc	b	bc	b	bc	b	bc	b	bc	bc	bc	bc
4	b	a	b		b	a	b	a	b	ac	bc	ac	bc	ac	bc	ac
5	a		a	a	a	ac	ac	ac	ac	ac	ac	ac	ac	ac	ac	ac
6	c		c		c		c		c		c		c		c	

Liveness analysis

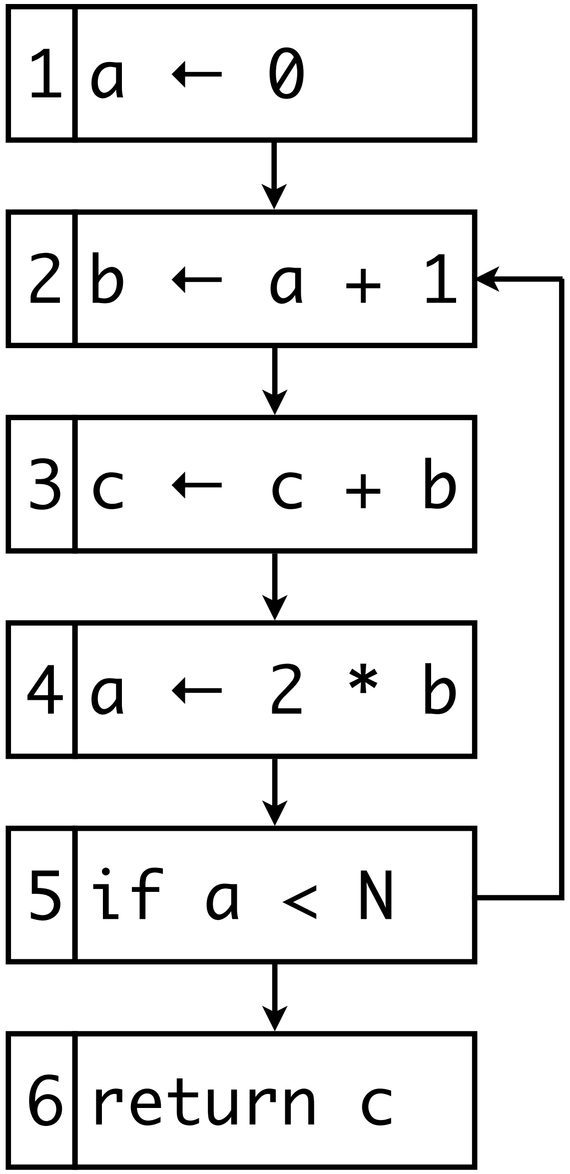
optimisation



	u							
6	c							
5	a							
4	b							
3	bc							
2	a							
1								

Liveness analysis

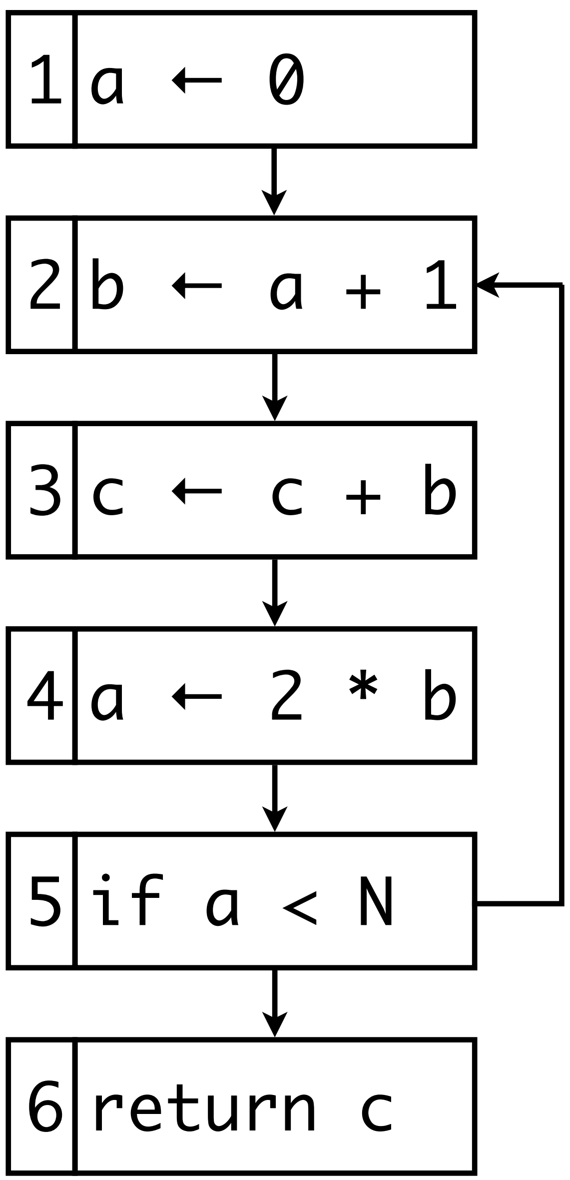
optimisation



	u	d						
6	c							
5	a							
4	b	a						
3	bc	c						
2	a	b						
1		a						

Liveness analysis

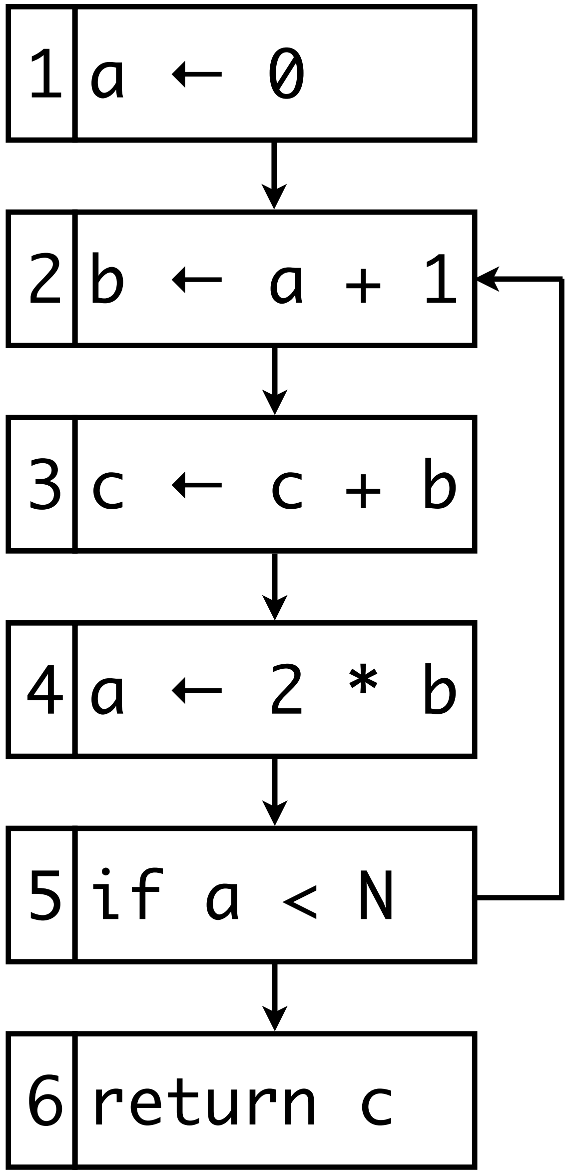
optimisation



			1					
	u	d	o	i				
6	c			c				
5	a		c	ac				
4	b	a	ac	bc				
3	bc	c	bc	bc				
2	a	b	bc	ac				
1		a	ac	a				

Liveness analysis

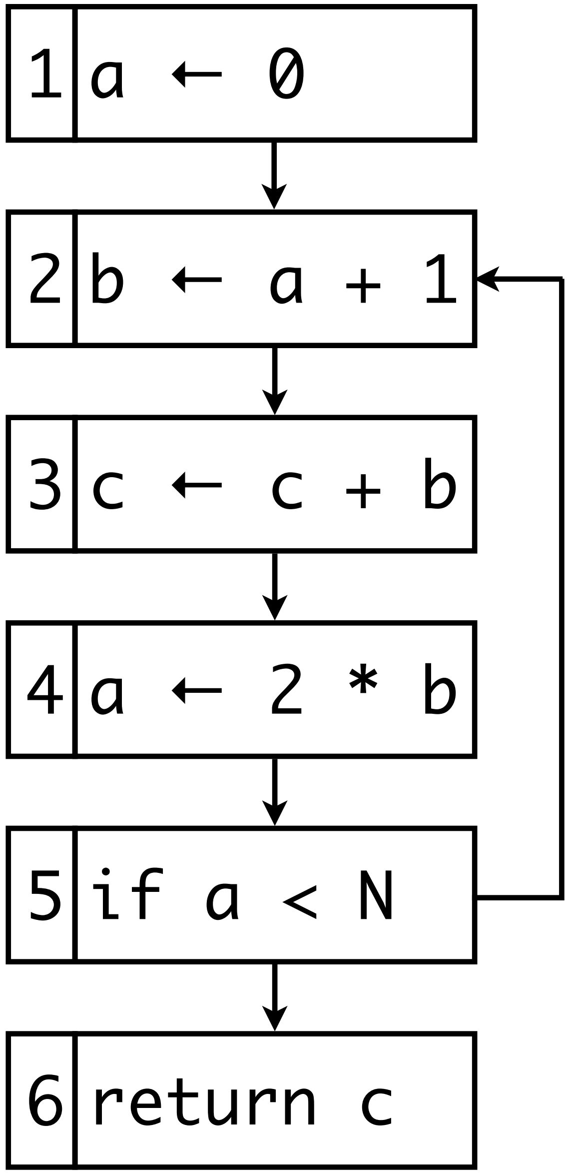
optimisation



			1		2	
	u	d	o	i	o	i
6	c			c		c
5	a		c	ac	ac	ac
4	b	a	ac	bc	ac	bc
3	bc	c	bc	bc	bc	bc
2	a	b	bc	ac	bc	ac
1		a	ac	a	ac	c

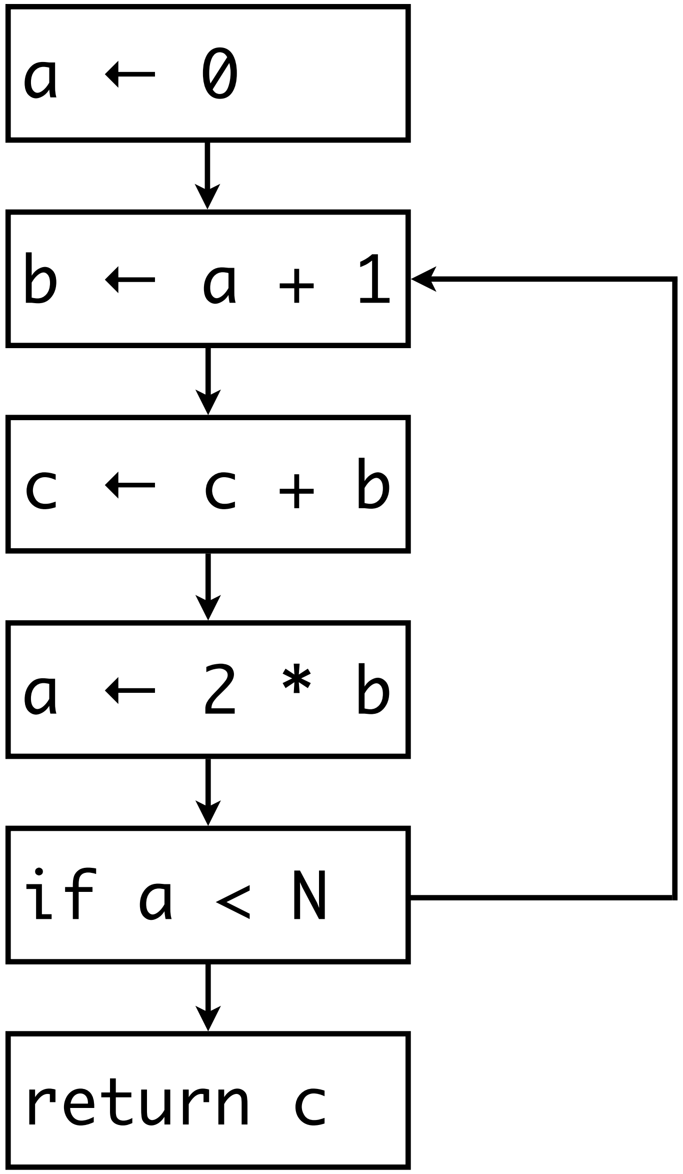
Liveness analysis

optimisation



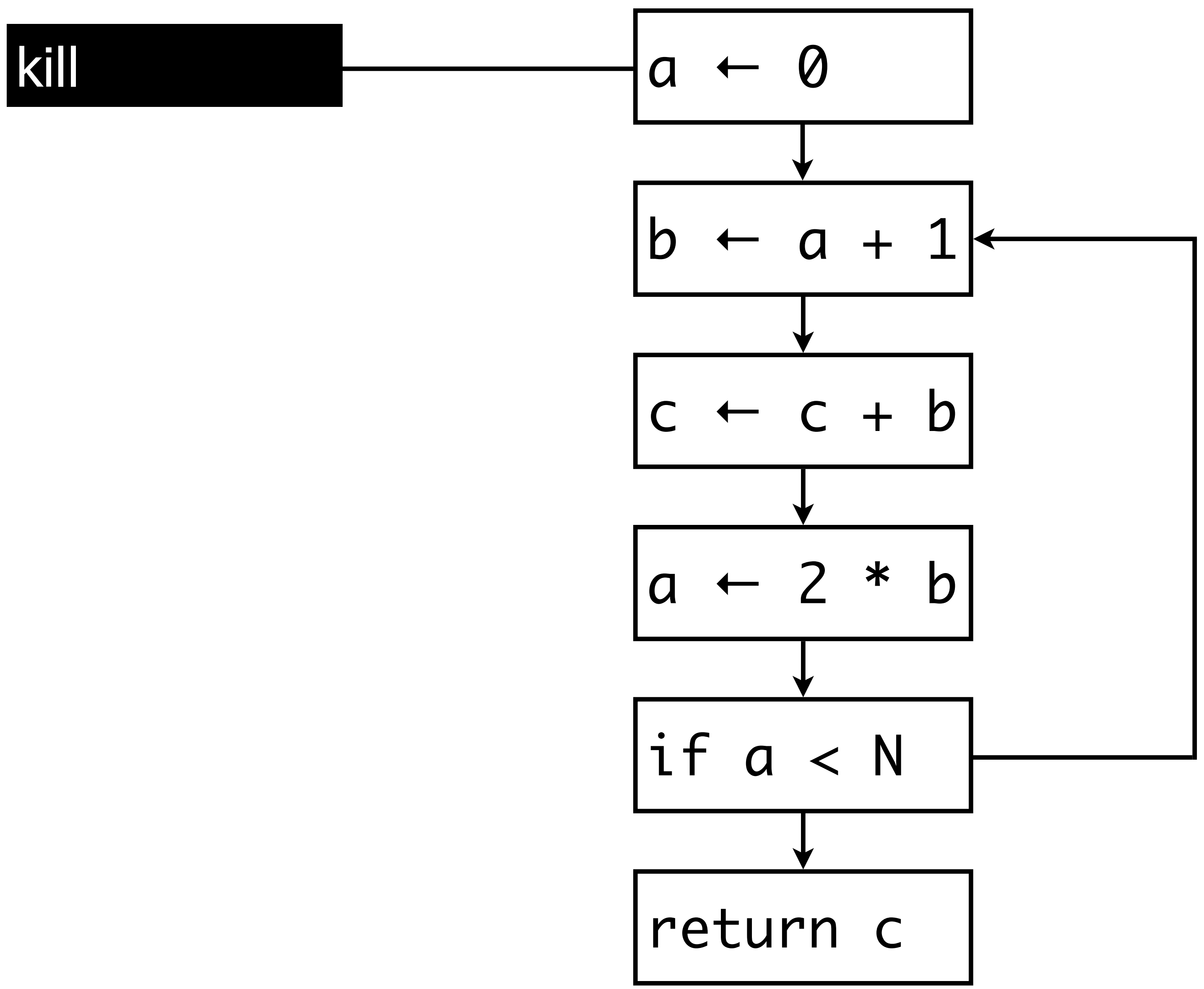
			1		2		3	
	u	d	o	i	o	i	o	i
6	c			c		c		c
5	a		c	ac	ac	ac	ac	ac
4	b	a	ac	bc	ac	bc	ac	bc
3	bc	c	bc	bc	bc	bc	bc	bc
2	a	b	bc	ac	bc	ac	bc	ac
1		a	ac	a	ac	c	ac	c

Liveness analysis
generalisation



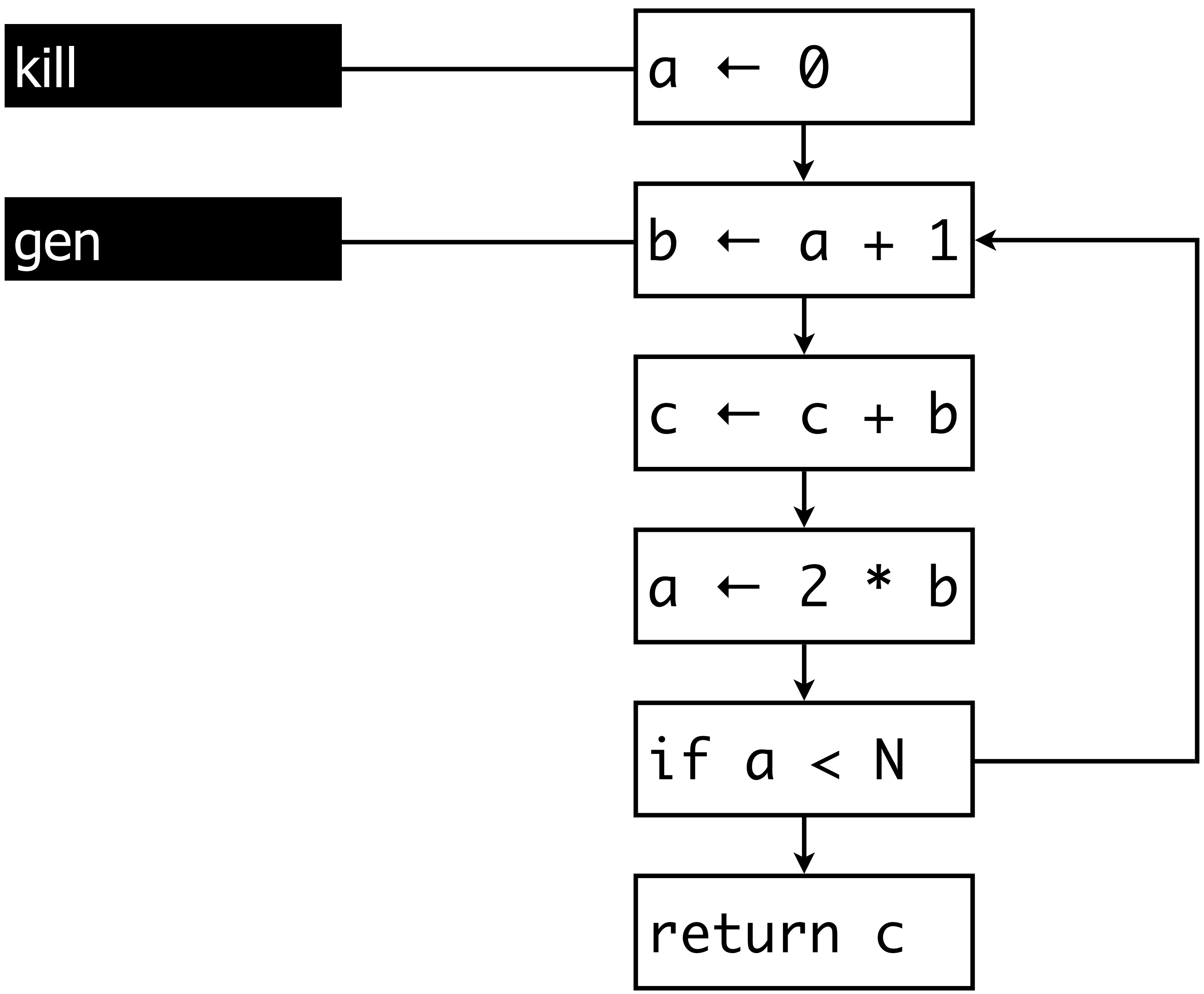
Liveness analysis

generalisation



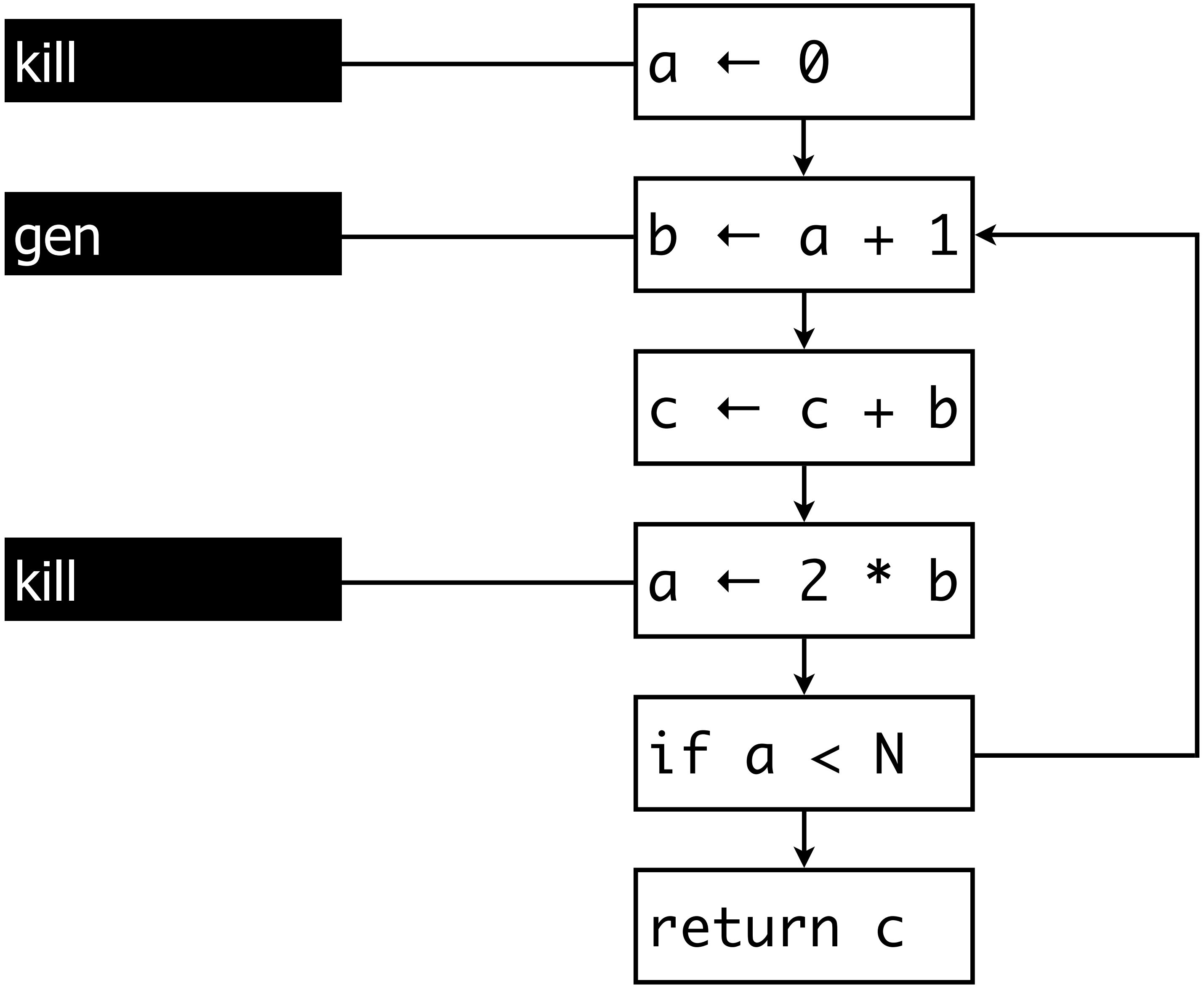
Liveness analysis

generalisation



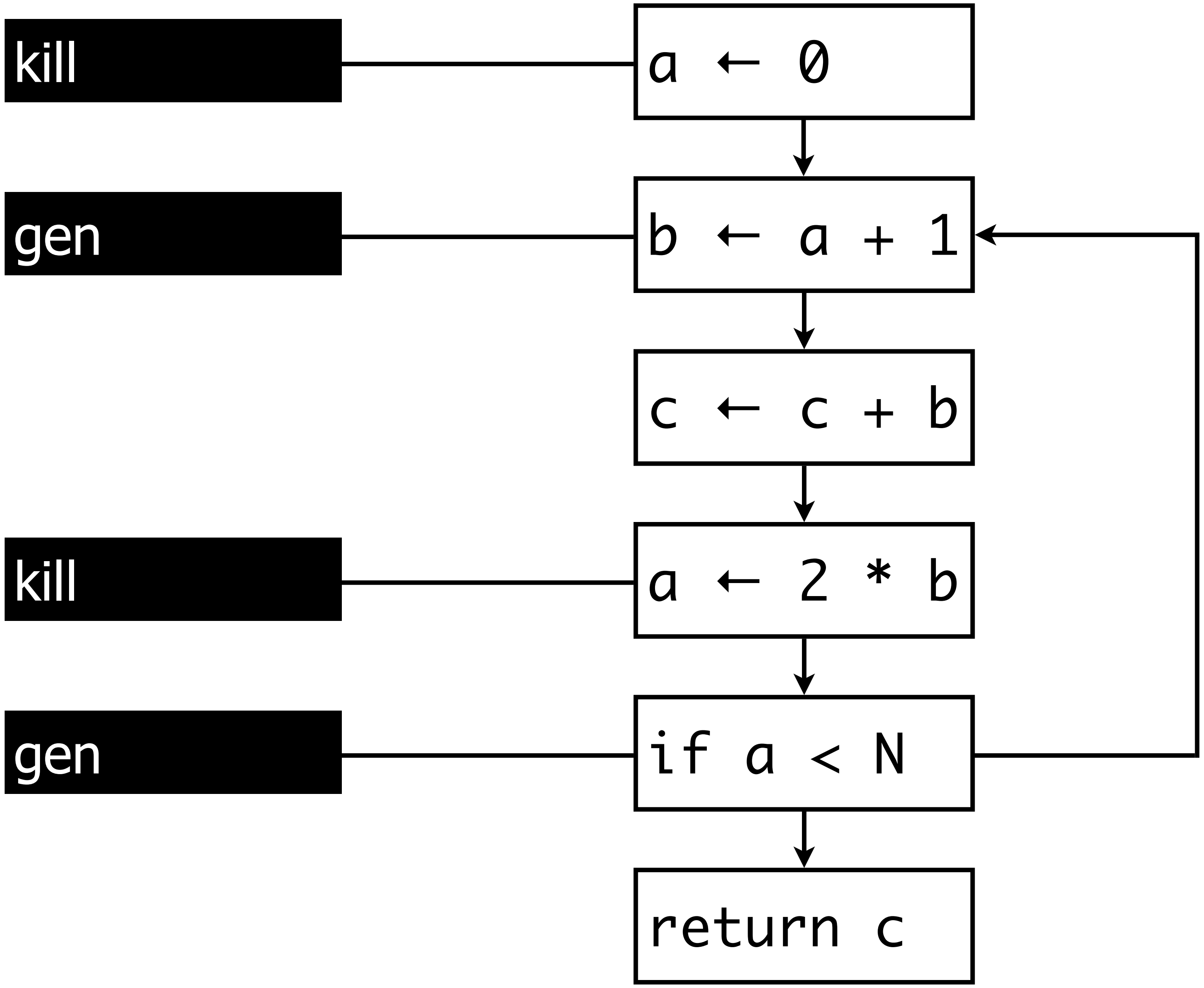
Liveness analysis

generalisation



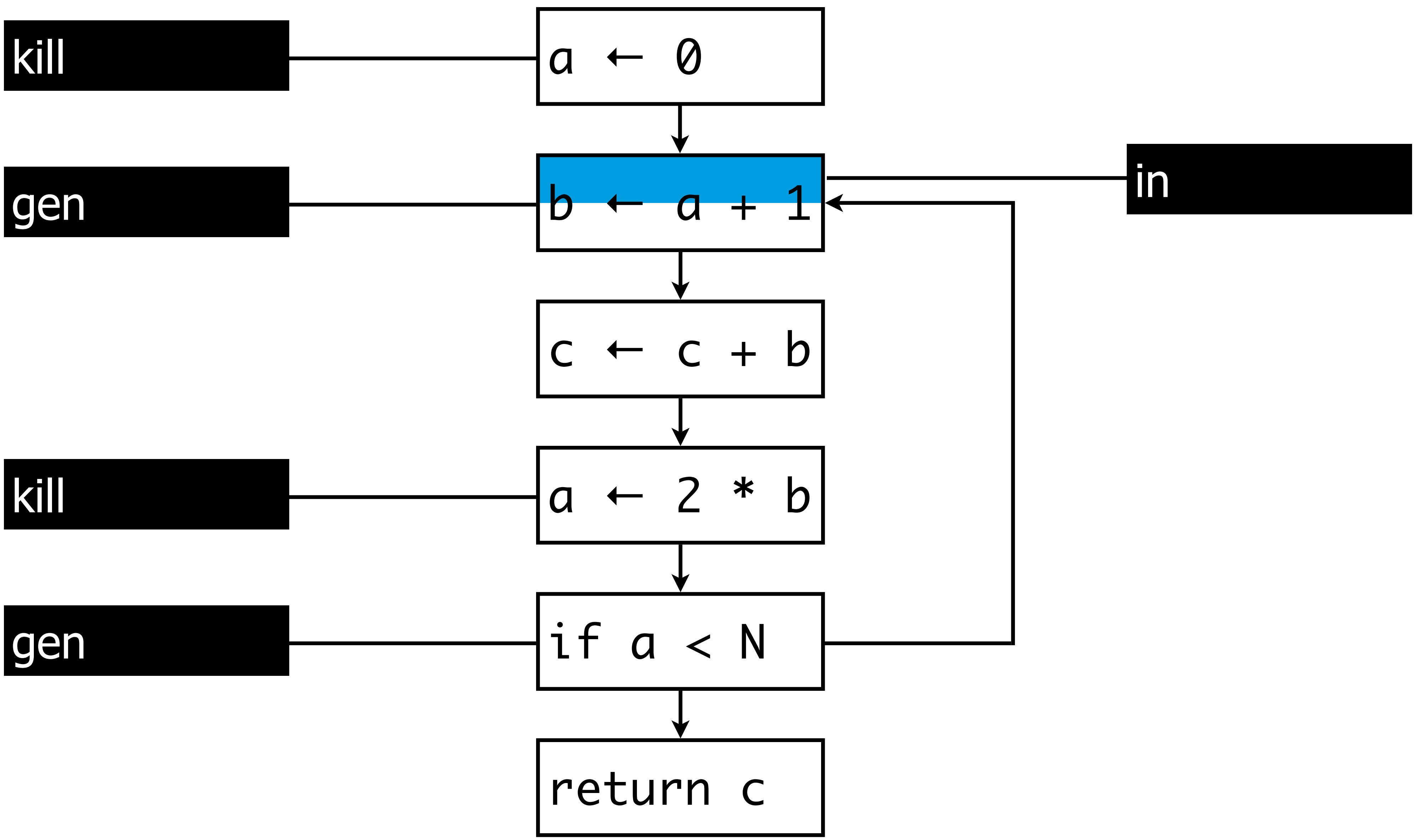
Liveness analysis

generalisation



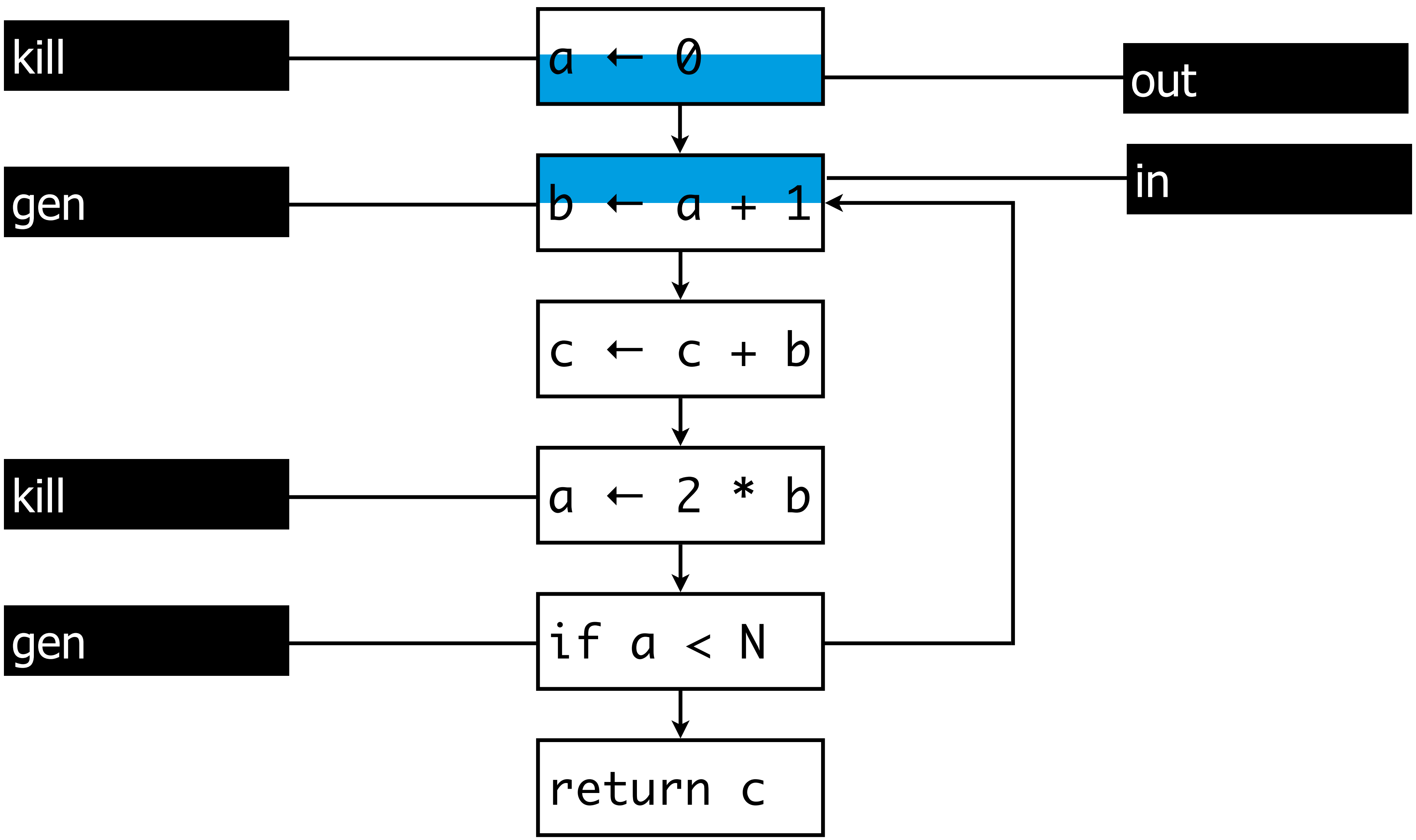
Liveness analysis

generalisation



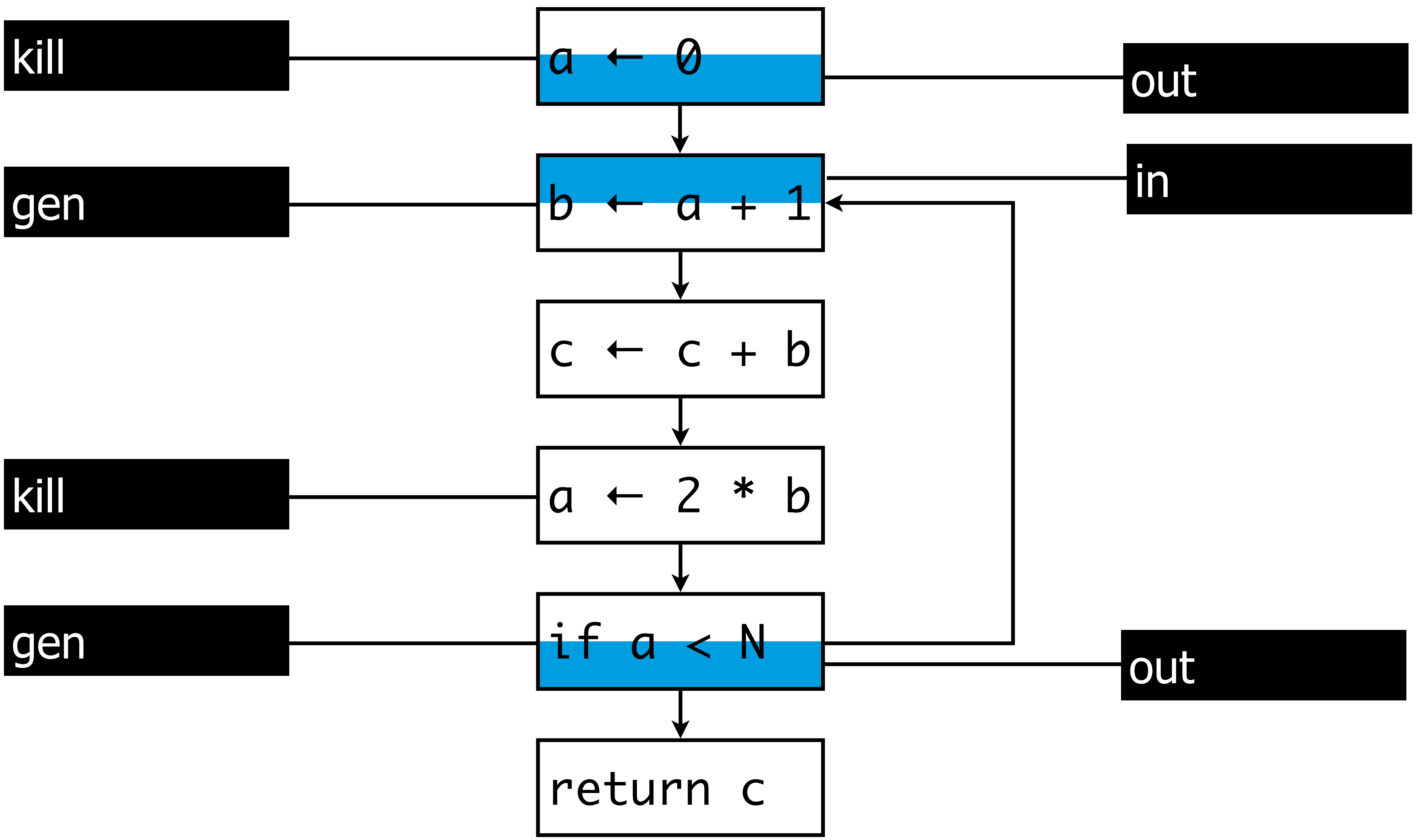
Liveness analysis

generalisation

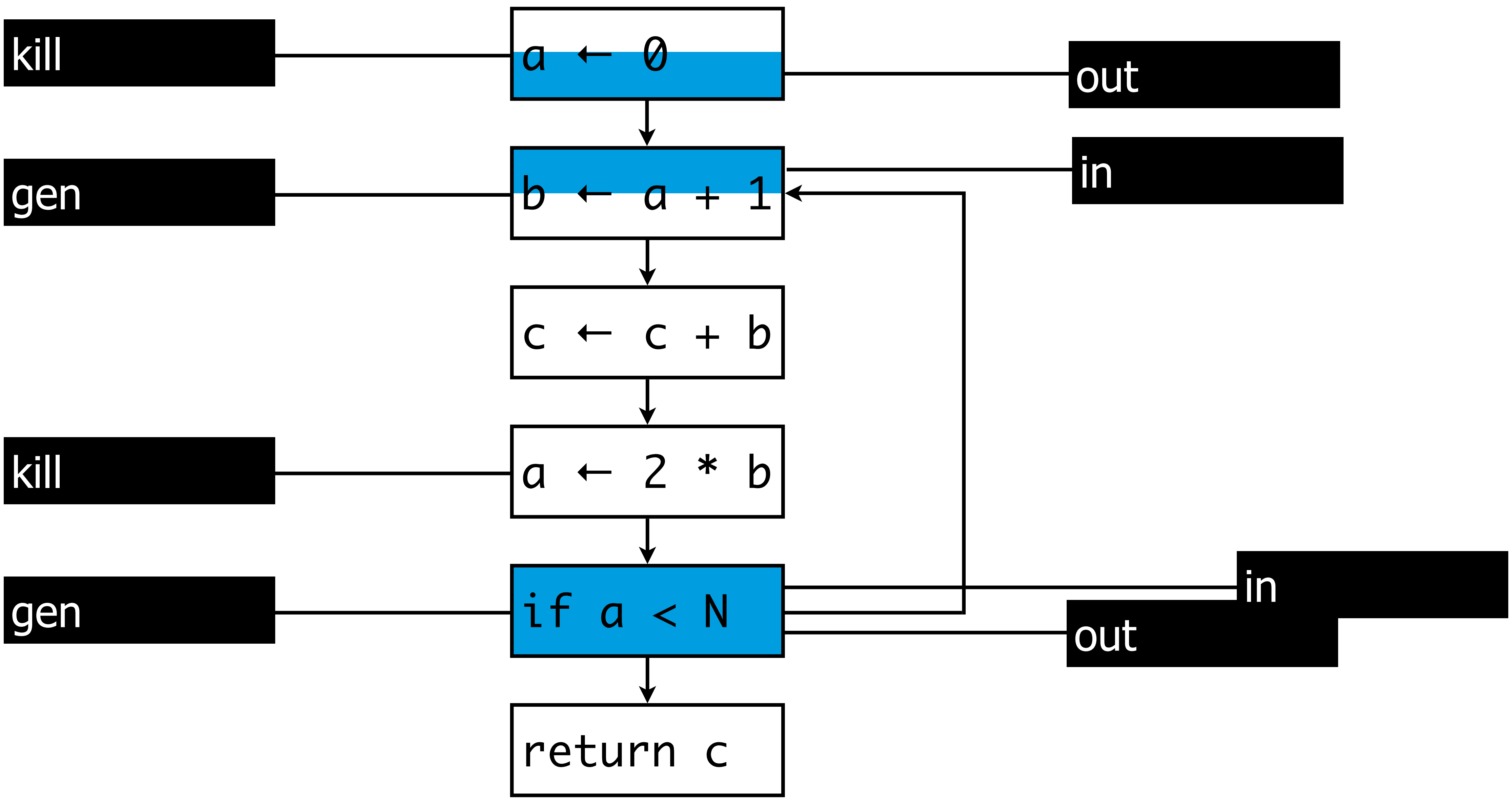


Liveness analysis

generalisation

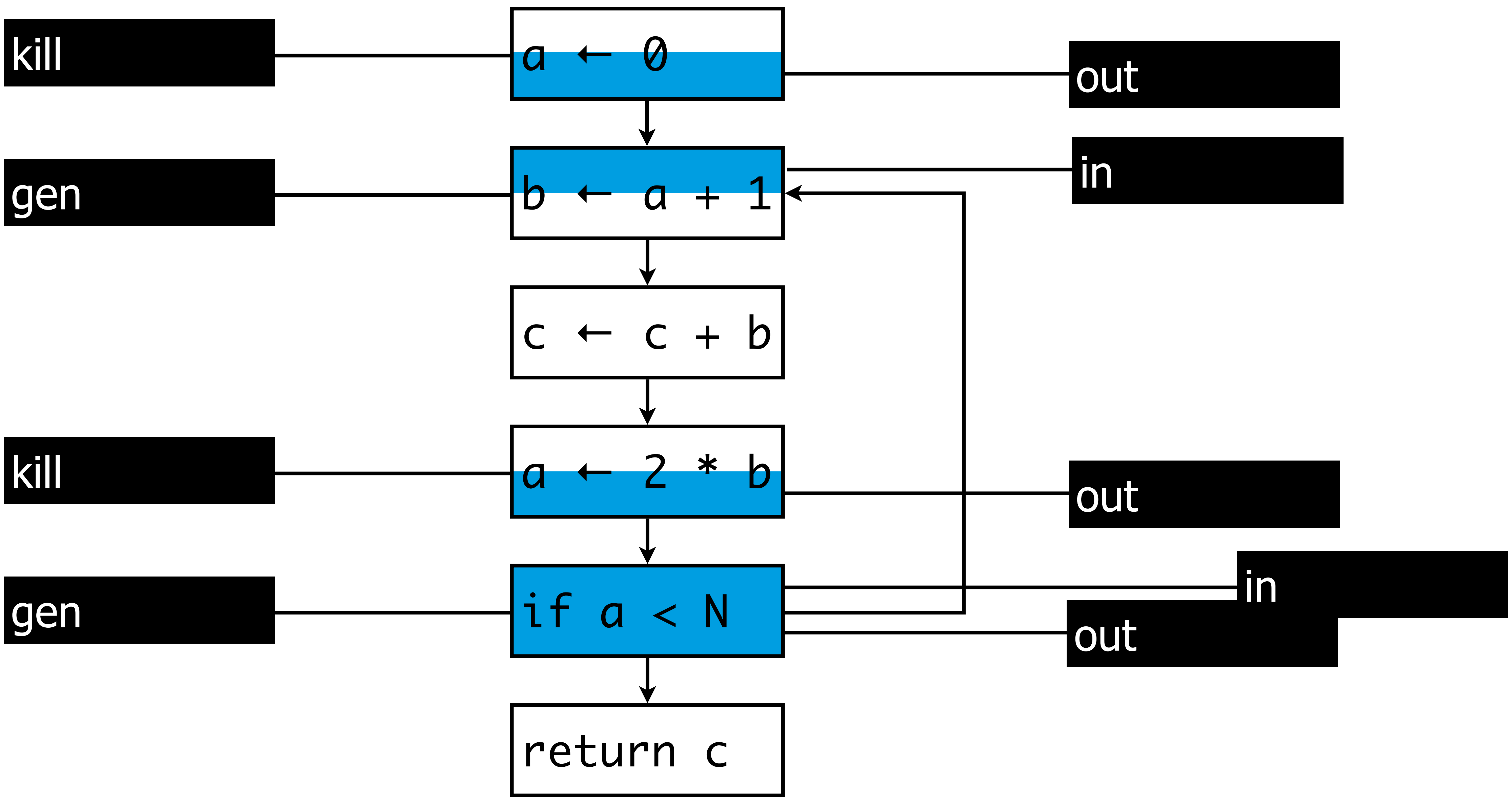


Liveness analysis
generalisation



Liveness analysis

generalisation



Liveness analysis

gen & kill

	gen	
$a \leftarrow b \oplus c$	{b,c}	
$a \leftarrow b$	{b}	
$a \leftarrow M[b]$	{b}	
$M[a] \leftarrow b$	{a,b}	
$f(a_1, \ldots, a_n)$	$\{a_1, \ldots, a_n\}$	
$a \leftarrow f(a_1, \ldots, a_n)$	$\{a_1, \ldots, a_n\}$	
goto L		
if $a \otimes b$	{a,b}	

Liveness analysis

gen & kill

	gen	kill
$a \leftarrow b \oplus c$	{b,c}	{a}
$a \leftarrow b$	{b}	{a}
$a \leftarrow M[b]$	{b}	{a}
$M[a] \leftarrow b$	{a,b}	
$f(a_1, \ldots, a_n)$	$\{a_1, \ldots, a_n\}$	
$a \leftarrow f(a_1, \ldots, a_n)$	$\{a_1, \ldots, a_n\}$	{a}
goto L		
if $a \otimes b$	{a,b}	

Liveness analysis
generalisation

$$in[n] = gen[n] \cup (out[n] - kill[n])$$

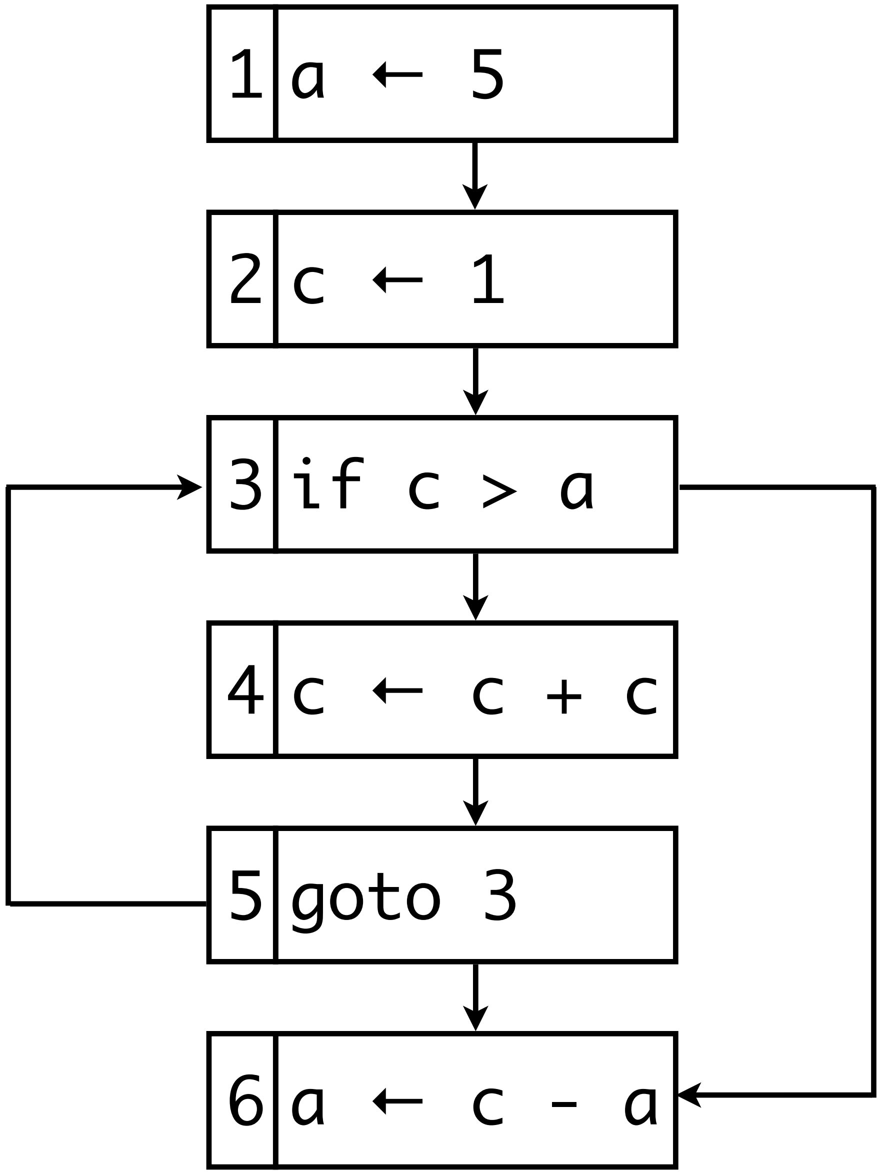
$$out[n] = \bigcup_{s \in succ[n]} in[s]$$

coffee break

III

more analyses

More analyses
reaching definitions



Reaching definitions

gen & kill

	gen	
d: a ← b ⊕ c	{d}	
d: a ← b	{d}	
d: a ← M[b]	{d}	
M[a] ← b		
f(a_1, …, a_n)		
d: a ← f(a_1, …, a_n)	{d}	
goto L		
if a ⊗ b		

Reaching definitions

gen & kill

	gen	kill
d: a ← b ⊕ c	{d}	defs(a)-{d}
d: a ← b	{d}	defs(a)-{d}
d: a ← M[b]	{d}	defs(a)-{d}
M[a] ← b		
f(a_1, …, a_n)		
d: a ← f(a_1, …, a_n)	{d}	defs(a)-{d}
goto L		
if a ⊗ b		

Reaching definitions

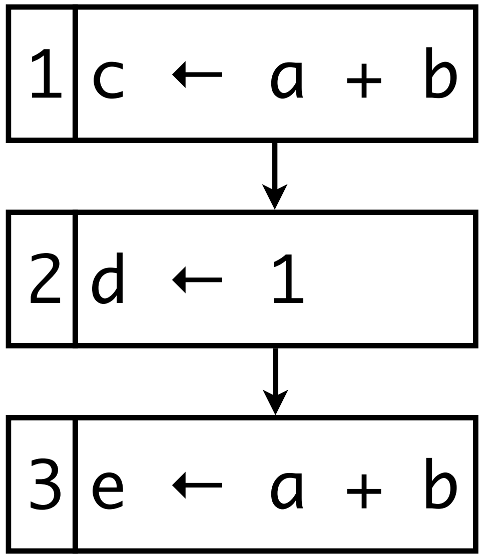
formalisation

$$in[n] = \bigcup_{p \in pred[n]} out[p]$$

$$out[n] = gen[n] \cup (in[n] - kill[n])$$

More analyses

available expressions



Available expressions

gen & kill

	gen	
d: a ← b ⊕ c	{b⊕c}-kill	
d: a ← b		
d: a ← M[b]	{M[b]}-kill	
M[a] ← b		
f(a_1, …, a_n)		
d: a ← f(a_1, …, a_n)		
goto L		
if a ⊗ b		

Available expressions

gen & kill

	gen	kill
d: a ← b ⊕ c	{b⊕c}-kill	exps(a)
d: a ← b		
d: a ← M[b]	{M[b]}-kill	exps(a)
M[a] ← b		exps(M[_])
f(a_1, …, a_n)		exps(M[_])
d: a ← f(a_1, …, a_n)		exps(M[_]) ∪ exps(a)
goto L		
if a ⊗ b		

Available expressions

formalisation

$$in[n] = \bigcap_{p \in pred[n]} out[p]$$

$$out[n] = gen[n] \cup (in[n] - kill[n])$$

IV

optimisations

Dead code elimination

example

```
a ← 0
b ← a + 1
c ← c + b
a ← 2 * b
return c
```

```
a ← 0
b ← a + 1
c ← c + b
return c
```

Constant propagation

example

```
a ← 0
b ← a + 1
c ← c + b
a ← 2 * b
return c
```

```
a ← 0
b ← 0 + 1
c ← c + b
a ← 2 * b
return c
```

Copy propagation
example

```
a ← e
b ← a + 1
c ← c + b
a ← 2 * b
return c
```

```
a ← e
b ← e + 1
c ← c + b
a ← 2 * b
return c
```

Common subexpression elimination

example

$c \leftarrow a + b$

$d \leftarrow 1$

$e \leftarrow a + b$

$x \leftarrow a + b$

$c \leftarrow x$

$d \leftarrow 1$

$e \leftarrow x$

V

summary

Summary
lessons learned

liveness analysis

- intermediate language
- control-flow graphs
- definition & algorithm

more dataflow analyses

- reaching definitions
- available expressions

optimisations

Literature

learn more

Andrew W. Appel, Jens Palsberg: Modern Compiler Implementation in Java, 2nd edition. 2002

copyrights & credits

Pictures
copyrights

Slide 1:
   ink swirl by Graham Richardson, some rights reserved

Slide 25:
   Seattle's Best Coffee by Dominica Williamson, some rights reserved

Slide 32:
   Animal Ark Reno by Joel Riley, some rights reserved