

## Verbesserung (Forts.):

→ Auch die Komposition lässt sich direkt implementieren:

$$(M_1 \circ M_2) x = b' \sqcup \bigsqcup_{y \in I'} y \quad \text{mit}$$

$$b' = b \sqcup \bigsqcup_{z \in I} b_z$$

$$I' = \bigcup_{z \in I} I_z \quad \text{sofern}$$

$$M_1 x = b \sqcup \bigsqcup_{y \in I} y$$

$$M_2 z = b_z \sqcup \bigsqcup_{y \in I_z} y$$

→ Die Effekte von Zuweisungen sehen dann so aus:

$$\llbracket x = e; \rrbracket^\# = \begin{cases} \text{Id}_{Vars} \oplus \{x \mapsto c\} & \text{falls } e = c \in \mathbb{Z} \\ \text{Id}_{Vars} \oplus \{x \mapsto y\} & \text{falls } e = y \in Vars \\ \text{Id}_{Vars} \oplus \{x \mapsto \top\} & \text{sonst} \end{cases}$$

... im Beispiel:

$$\begin{aligned} \llbracket t = 0; \rrbracket^\# &= \{a_1 \mapsto a_1, \text{ret} \mapsto \text{ret}, t \mapsto 0\} \\ \llbracket a_1 = t; \rrbracket^\# &= \{a_1 \mapsto t, \text{ret} \mapsto \text{ret}, t \mapsto t\} \end{aligned}$$

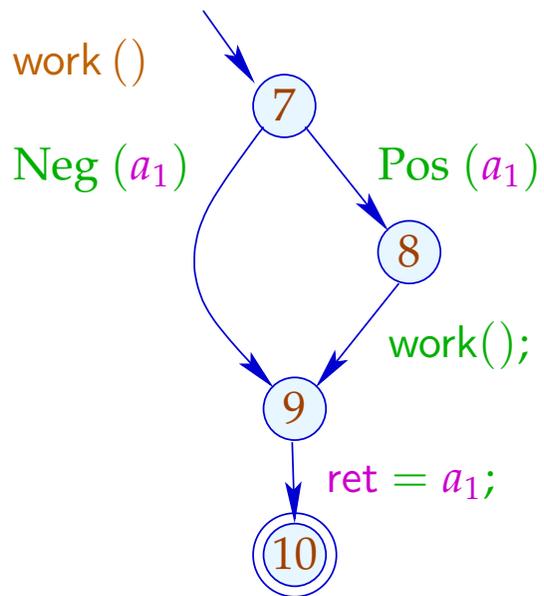
Um die Analyse zu implementieren, müssen wir nur noch den Effekt eines Aufrufs  $k = (\_, f (); \_)$  aus dem Effekt der Prozedur  $f$  ermitteln:

$$\begin{aligned} \llbracket k \rrbracket^\# &= H(\llbracket f \rrbracket^\#) \quad \text{wobei:} \\ H(M) &= \text{Id}|_{Locals} \oplus \{x \mapsto (M \circ E_f) x \mid x \in Globals\} \\ E_f x &= \begin{cases} x & \text{falls } x \in Globals \\ 0 & \text{sonst} \end{cases} \end{aligned}$$

... im Beispiel:

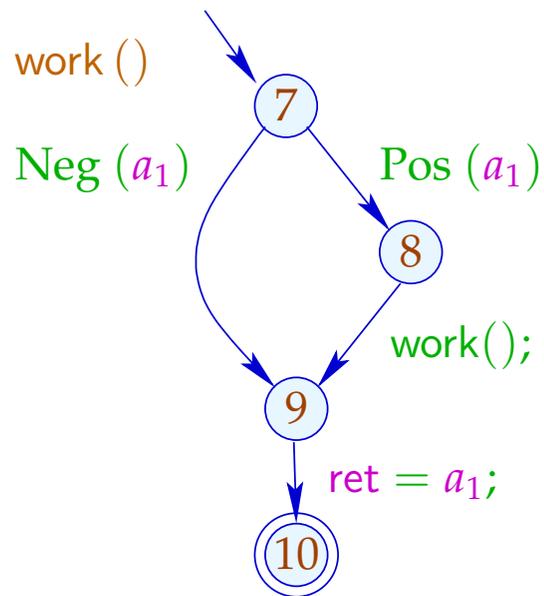
$$\begin{aligned} \text{Falls } \llbracket \text{work} \rrbracket^\# &= \{a_1 \mapsto a_1, \text{ret} \mapsto a_1, t \mapsto t\} \\ \text{dann } H \llbracket \text{work} \rrbracket^\# &= \text{Id} \oplus \{a_1 \mapsto a_1, \text{ret} \mapsto a_1\} \\ &= \{a_1 \mapsto a_1, \text{ret} \mapsto a_1, t \mapsto t\} \end{aligned}$$

Damit können wir die Fixpunkt-Iteration durchführen :-)



	1
7	$\{a_1 \mapsto a_1, \text{ret} \mapsto \text{ret}, t \mapsto t\}$
9	$\{a_1 \mapsto a_1, \text{ret} \mapsto \text{ret}, t \mapsto t\}$
10	$\{a_1 \mapsto a_1, \text{ret} \mapsto a_1, t \mapsto t\}$
8	$\{a_1 \mapsto a_1, \text{ret} \mapsto a_1, t \mapsto t\}$

$$\begin{aligned}
 \llbracket (8, \dots, 9) \rrbracket^\# \circ \llbracket 8 \rrbracket^\# &= \{a_1 \mapsto a_1, \text{ret} \mapsto a_1, t \mapsto t\} \circ \\
 &\quad \{a_1 \mapsto a_1, \text{ret} \mapsto \text{ret}, t \mapsto t\} \\
 &= \{a_1 \mapsto a_1, \text{ret} \mapsto a_1, t \mapsto t\}
 \end{aligned}$$



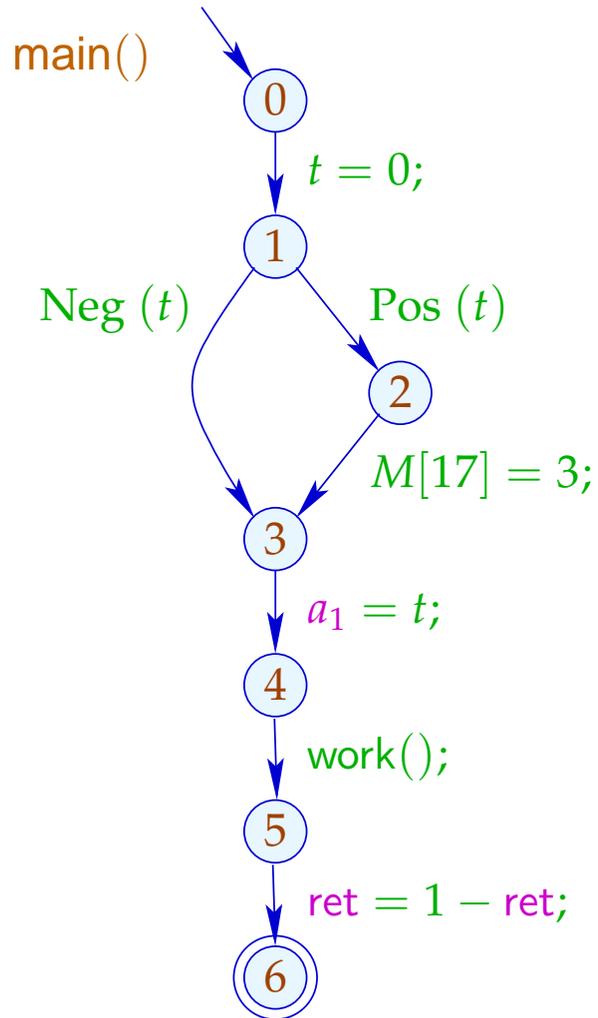
	2
7	$\{a_1 \mapsto a_1, \text{ret} \mapsto \text{ret}, t \mapsto t\}$
9	$\{a_1 \mapsto a_1, \text{ret} \mapsto a_1 \sqcup \text{ret}, t \mapsto t\}$
10	$\{a_1 \mapsto a_1, \text{ret} \mapsto a_1, t \mapsto t\}$
8	$\{a_1 \mapsto a_1, \text{ret} \mapsto \text{ret}, t \mapsto t\}$

$$\begin{aligned}
 \llbracket (8, \dots, 9) \rrbracket^\# \circ \llbracket 8 \rrbracket^\# &= \{a_1 \mapsto a_1, \text{ret} \mapsto a_1, t \mapsto t\} \circ \\
 &\quad \{a_1 \mapsto a_1, \text{ret} \mapsto \text{ret}, t \mapsto t\} \\
 &= \{a_1 \mapsto a_1, \text{ret} \mapsto a_1, t \mapsto t\}
 \end{aligned}$$

Wenn wir die Effekte von Funktionsaufrufen kennen, können wir ein Ungleichungssystem aufstellen, um den abstrakten Zustand bei Erreichen eines Punkts ermitteln:

$$\begin{array}{lll}
 \mathcal{R}[\text{main}] & \supseteq & \text{enter}^\# d_0 \\
 \mathcal{R}[f] & \supseteq & \text{enter}^\# (\mathcal{R}[u]) \quad k = (u, f(), \_) \quad \text{Aufruf} \\
 \mathcal{R}[v] & \supseteq & \mathcal{R}[f] \quad v \quad \text{Anfangspunkt von } f \\
 \mathcal{R}[v] & \supseteq & \llbracket k \rrbracket^\# (\mathcal{R}[u]) \quad k = (u, \_, v) \quad \text{Kante}
 \end{array}$$

... im Beispiel:



0	$\{a_1 \mapsto \top, \text{ret} \mapsto \top, t \mapsto 0\}$
1	$\{a_1 \mapsto \top, \text{ret} \mapsto \top, t \mapsto 0\}$
2	$\{a_1 \mapsto \top, \text{ret} \mapsto \top, t \mapsto 0\}$
3	$\{a_1 \mapsto \top, \text{ret} \mapsto \top, t \mapsto 0\}$
4	$\{a_1 \mapsto 0, \text{ret} \mapsto \top, t \mapsto 0\}$
5	$\{a_1 \mapsto 0, \text{ret} \mapsto 0, t \mapsto 0\}$
6	$\{a_1 \mapsto 0, \text{ret} \mapsto \top, t \mapsto 0\}$

## Diskussion:

- Zumindest **Kopier-Konstanten** lassen sich interprozedural ermitteln.
- Dazu mussten wir Bedingungen und kompliziertere Zuweisungen ignorieren :-)
- In der zweiten Phase hätten wir allerdings exakter rechnen können :-)
- Die weitere Abstrahierung war aus zwei Gründen notwendig:
  - (1) Die Menge der auftretenden Transformer  $\mathbb{M} \subseteq \mathbb{D} \rightarrow \mathbb{D}$  muss **endlich** sein;
  - (2) Die Funktionen  $M \in \mathbb{M}$  müssen **effizient** implementierbar sein :-)
- Auf die zweite Bedingung kann evt. verzichtet werden ...

## Beobachtung:

Sharir/Pnueli, Cousot

- Oft werden Prozeduren nur mit **wenigen** verschiedenen abstrakten Argumenten aufgerufen.
- Man könnte dann doch jede Prozedur für nur genau diese Aufrufe analysieren :-)
- Stelle das folgende Ungleichungssystem auf:

$$\llbracket v, a \rrbracket^\# \sqsupseteq a \quad v \text{ Eintrittspunkt}$$

$$\llbracket v, a \rrbracket^\# \sqsupseteq \text{combine}^\# (\llbracket u, a \rrbracket^\#, \llbracket f, \text{enter}^\# \llbracket u, a \rrbracket^\# \rrbracket^\#)$$

$(u, f ();, v)$  Aufruf

$$\llbracket v, a \rrbracket^\# \sqsupseteq \llbracket lab \rrbracket^\# \llbracket u, a \rrbracket^\# \quad k = (u, lab, v) \text{ Kante}$$

$$\llbracket f, a \rrbracket^\# \sqsupseteq \llbracket stop_f, a \rrbracket^\# \quad stop_f \text{ Endpunkt von } f$$

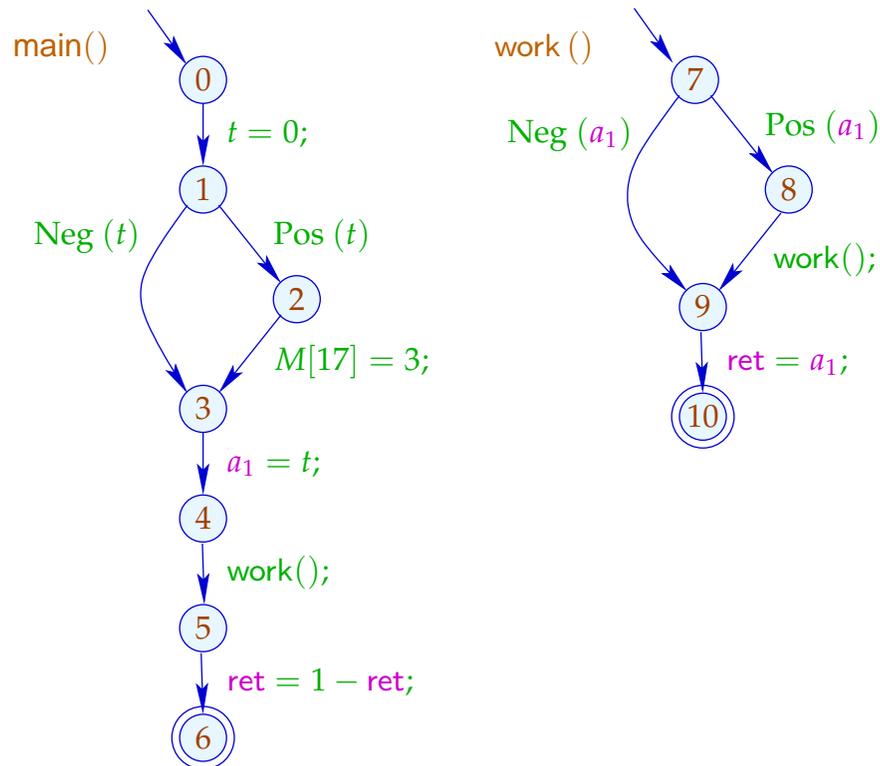
$$// \llbracket v, a \rrbracket^\# = \text{Wert des Effekts für das Argument } a .$$

## Diskussion:

- Dieses Ungleichungssystem ist i.a. riesengroß :-)
- Wir wollen es aber gar nicht komplett lösen !!!
- Uns reicht es, die korrekten Werte für alle Aufrufe zu ermitteln, die vorkommen, d.h. für die Berechnung des Werts  $\llbracket \text{main}(), a_0 \rrbracket^\#$  benötigt werden  $\implies$  wir verwenden unseren lokalen Fixpunkt-Algorithmus :-))
- Der Fixpunkt-Algo liefert uns sogar noch die Menge der aktuellen Parameter  $a \in \mathbb{D}$ , für die eine Funktion (möglicherweise) aufgerufen wird sowie die Werte an allen ihren Programm-Punkten für jeden dieser Aufrufe :-)

... im Beispiel:

Versuchen wir einfach einmal eine **volle** Konstanten-Propagation ...



	$a_1$	ret	$a_1$	ret
0	T	T	T	T
1	T	T	T	T
2	T	T	⊥	
3	T	T	T	T
4	T	T	0	T
7	0	T	0	T
8	0	T	⊥	
9	0	T	0	T
10	0	T	0	0
5	T	T	0	0
main()	T	T	0	1

## Diskussion:

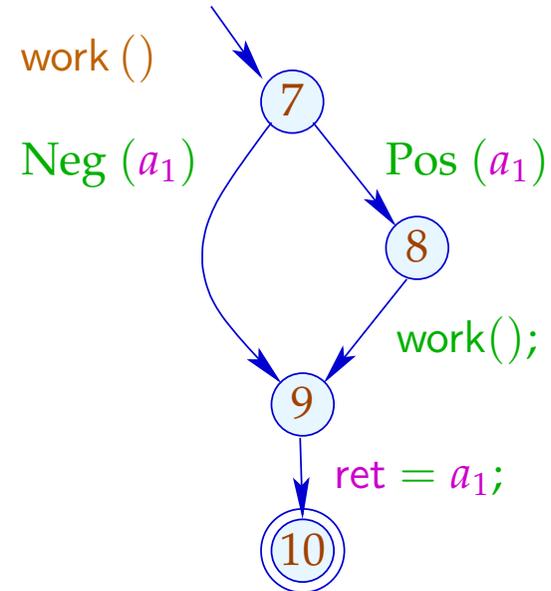
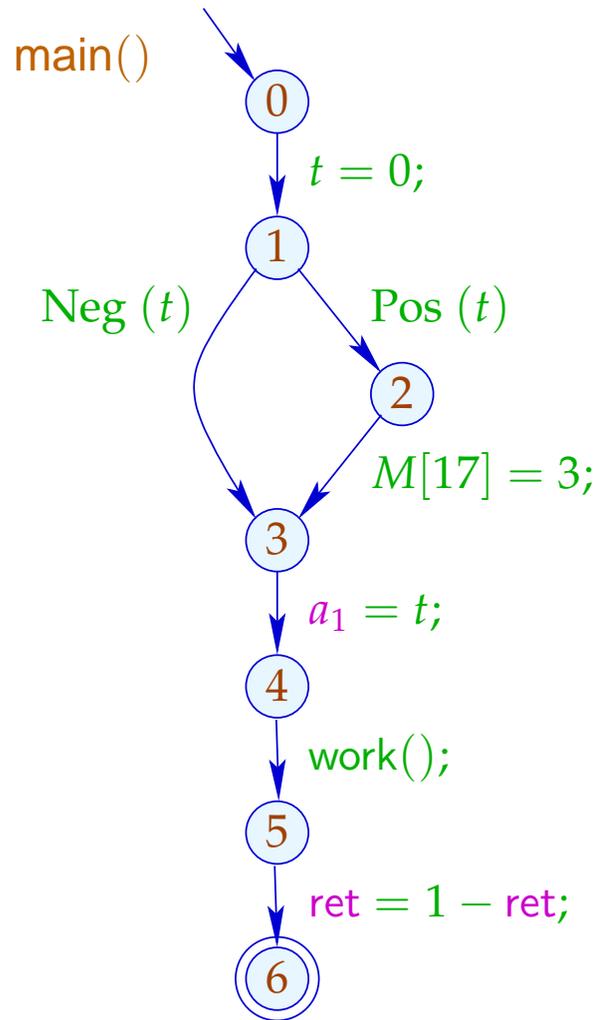
- Im Beispiel terminiert die Analyse **schnell** :-)
- Falls  $\mathbb{D}$  endliche Höhe hat, terminiert die Analyse, sofern nur jede Funktion während der Iteration nur mit **endlich vielen** verschiedenen Argumenten aufgerufen wird :-))
- Analoge Analyse-Algorithmen erwiesen sich bei der Analyse von **Prolog** als äußerst effizient und präzise :-)
- Zusammen mit einer Points-To-Analyse und Propagation selbst von negativer Konstanten-Information haben wir diesen Algorithmus äußerst erfolgreich zur Fehlersuche in **C** mit **Posix**-Threads eingesetzt :-)

## (2) Der Call-String-Ansatz:

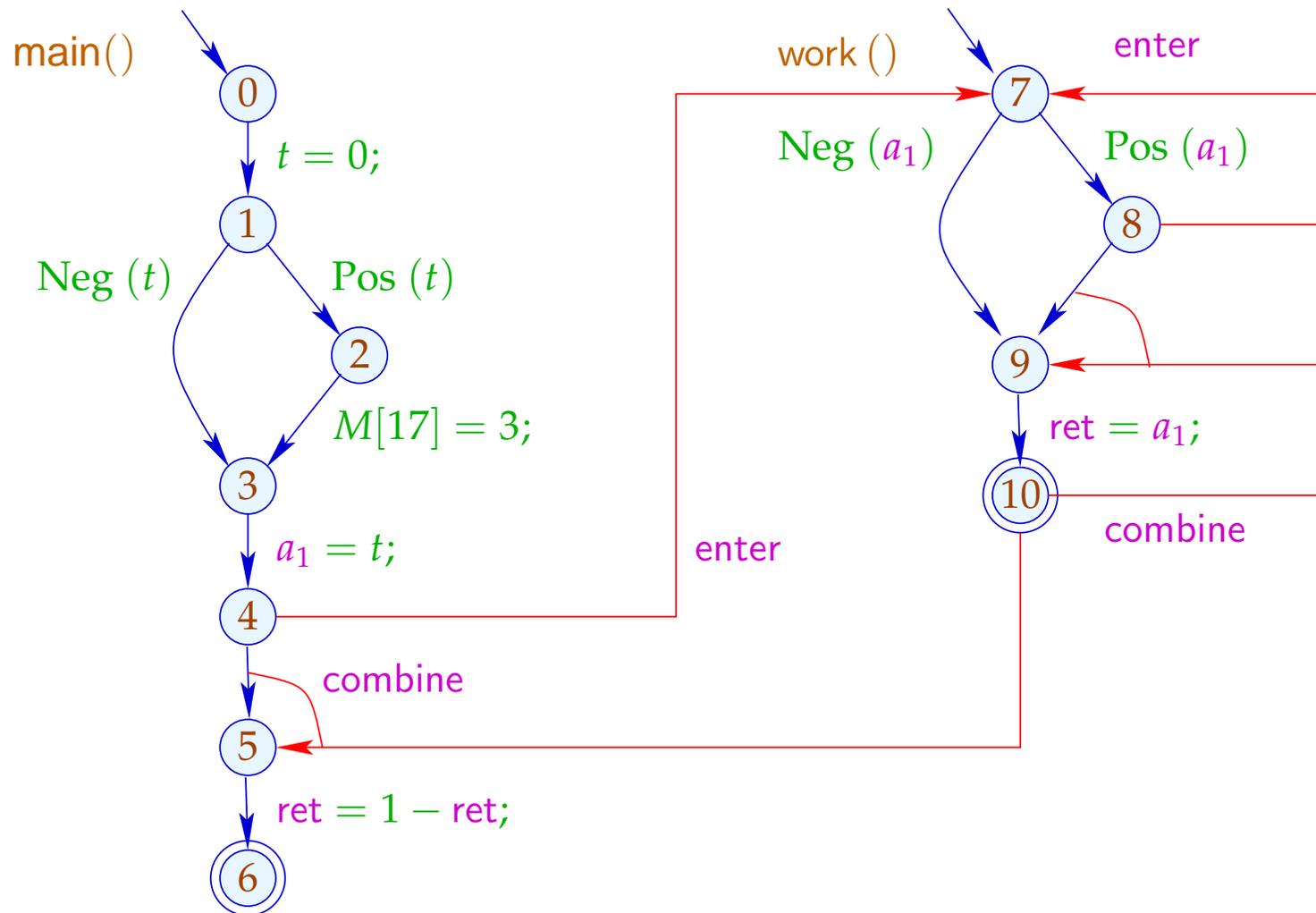
### Idee:

- Berechne die Menge aller erreichbaren Aufrufkeller!
- Diese ist i.a. unendlich :-)
- Handle Keller bis zu einer festen Tiefe  $d$  exakt! Behalte von längeren Kellern nur das obere Ende der Länge  $d$  :-)
- Wichtiger Spezialfall:  $d = 0$ .
  - ⇒ Betrachte nur die obersten Kellerrahmen ...

... im Beispiel:



... im Beispiel:



Die Bedingungen für  $5, 7, 10$  sind dann etwa:

$$\mathcal{R}[5] \sqsupseteq \text{combine}^\#(\mathcal{R}[4], \mathcal{R}[10])$$

$$\mathcal{R}[7] \sqsupseteq \text{enter}^\#(\mathcal{R}[4])$$

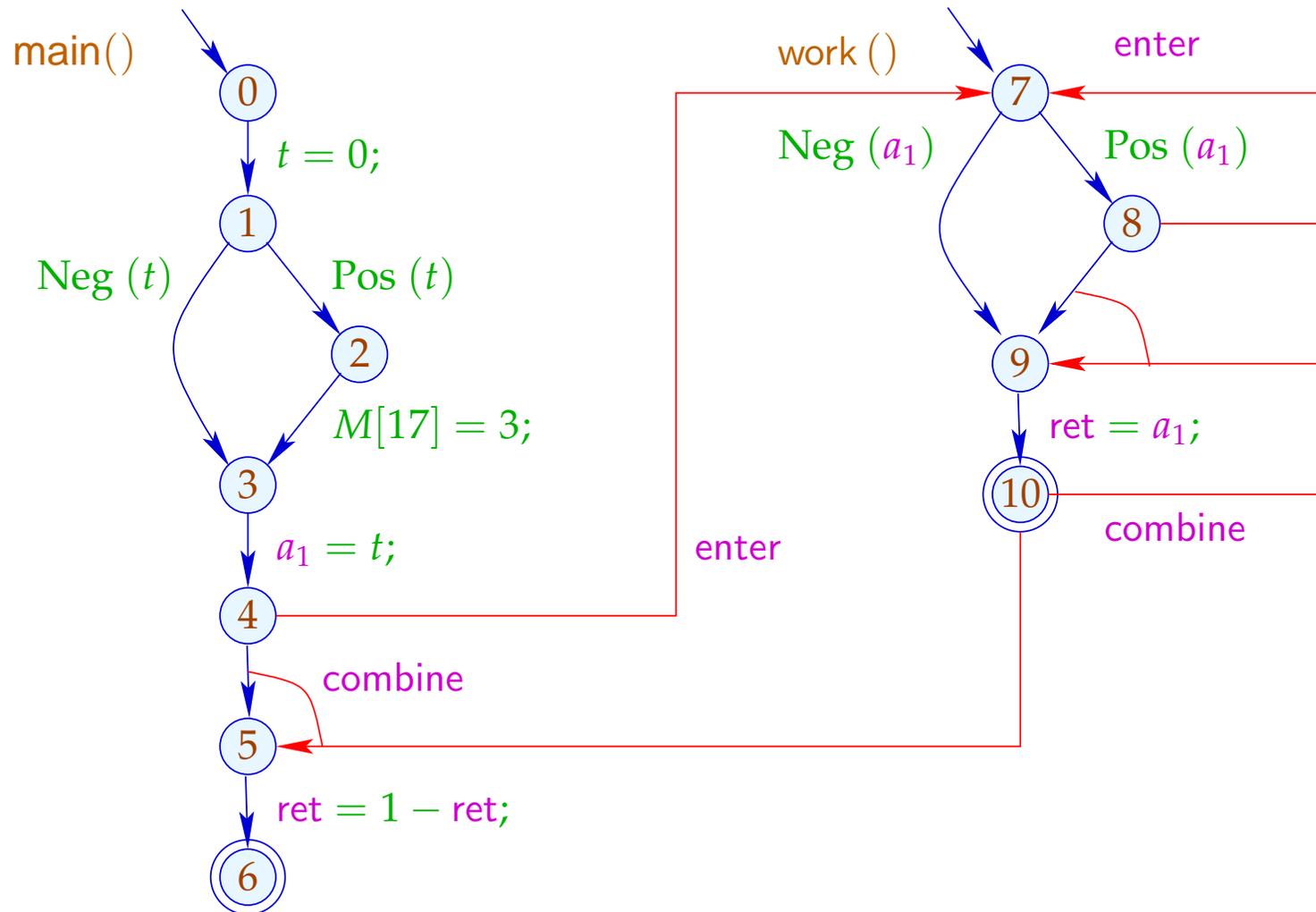
$$\mathcal{R}[7] \sqsupseteq \text{enter}^\#(\mathcal{R}[8])$$

$$\mathcal{R}[9] \sqsupseteq \text{combine}^\#(\mathcal{R}[8], \mathcal{R}[10])$$

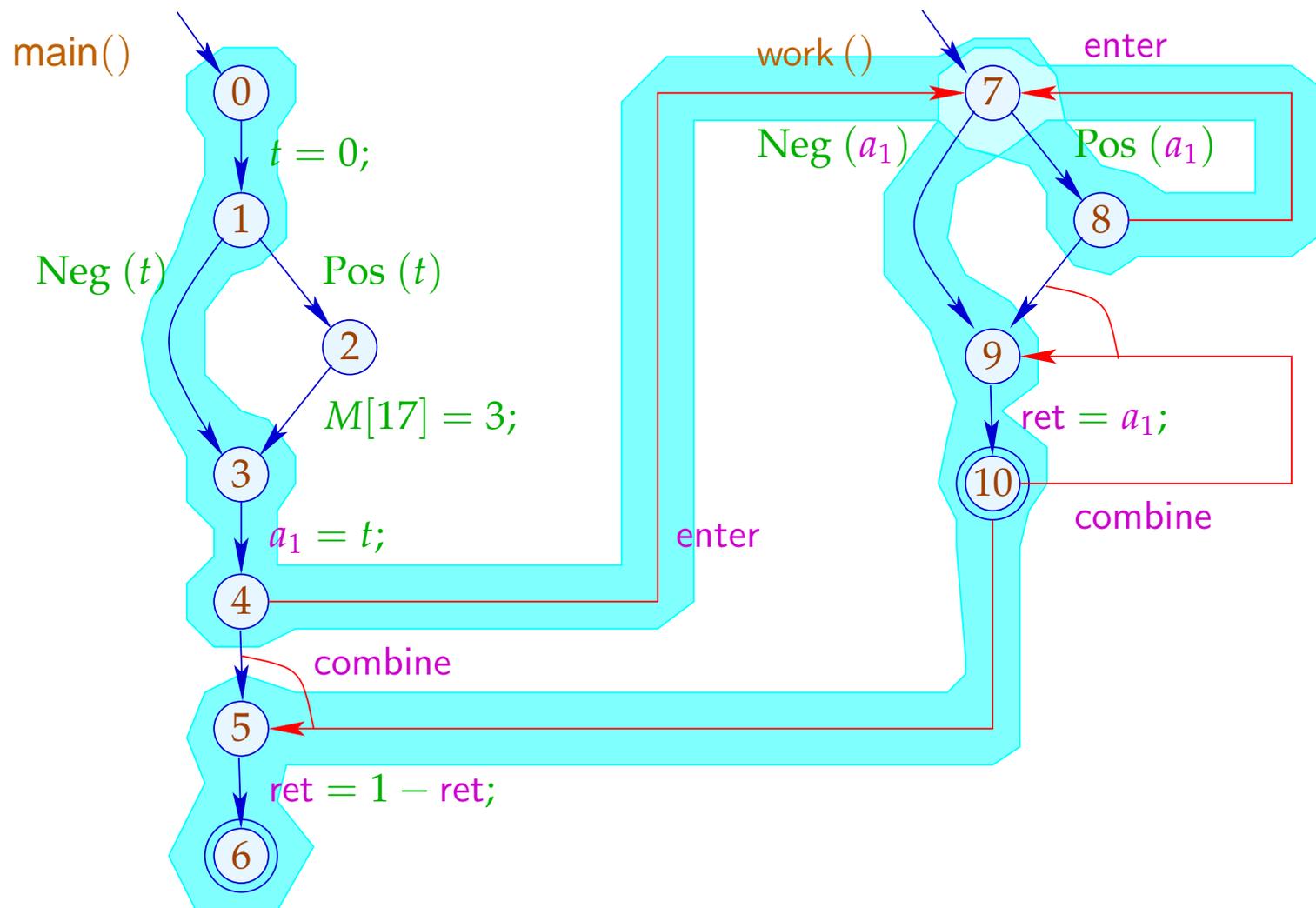
**Achtung:**

Der resultierende Supergraph enthält offensichtlich **unmögliche Pfade ...**

... im Beispiel ist das etwa:



... im Beispiel ist das etwa:



## Beachte:

- Im Beispiel finden wir zwar die gleichen Ergebnisse:  
Mehr Pfade machen die Ergebnisse evt. **weniger präzise**.  
Insbesondere analysieren wir jede Funktion nur für **ein** (evt. sehr nichtssagendes) Argument-Tupel **:-)**
- Die Analyse terminiert — sofern nur  $\mathbb{D}$  keine unendlichen echt aufsteigenden Ketten besitzt **:-)**
- Die Korrektheit zeigt man relativ zur operationellen Semantik mit den Stacks.
- Für die Korrektheit des funktionalen Ansatzes ist die Semantik über Berechnungswälder besser geeignet **:-)**