Verbesserung (Forts.):

→ Auch die Komposition lässt sich direkt implementieren:

$$
\begin{array}{rcll}
(M_1 \circ M_2)\ x & = & b' \sqcup \bigsqcup_{y \in I'} y & \text{mit} \\
b' & = & b \sqcup \bigsqcup_{z \in I} b_z & \\
I' & = & \bigcup_{z \in I} I_z & \text{sofern} \\
M_1\ x & = & b \sqcup \bigsqcup_{y \in I} y & \\
M_2\ z & = & b_z \sqcup \bigsqcup_{y \in I_z} y &
\end{array}
$$

→ Die Effekte von Zuweisungen sehen dann so aus:

$$
[\![x = e;]\!]^\sharp \quad = \quad \begin{cases}
\mathsf{Id}_{Vars} \oplus \{x \mapsto c\} & \text{falls} \quad e = c \in \mathbb{Z} \\
\mathsf{Id}_{Vars} \oplus \{x \mapsto y\} & \text{falls} \quad e = y \in Vars \\
\mathsf{Id}_{Vars} \oplus \{x \mapsto \top\} & \text{sonst}
\end{cases}
$$

... im Beispiel:

$$
\begin{array}{lcl}
[\![t = 0;]\!]^\sharp & = & \{a_1 \mapsto a_1, \mathsf{ret} \mapsto \mathsf{ret}, \boxed{t \mapsto 0}\} \\
[\![a_1 = t;]\!]^\sharp & = & \{\boxed{a_1 \mapsto t}, \mathsf{ret} \mapsto \mathsf{ret}, t \mapsto t\}
\end{array}
$$

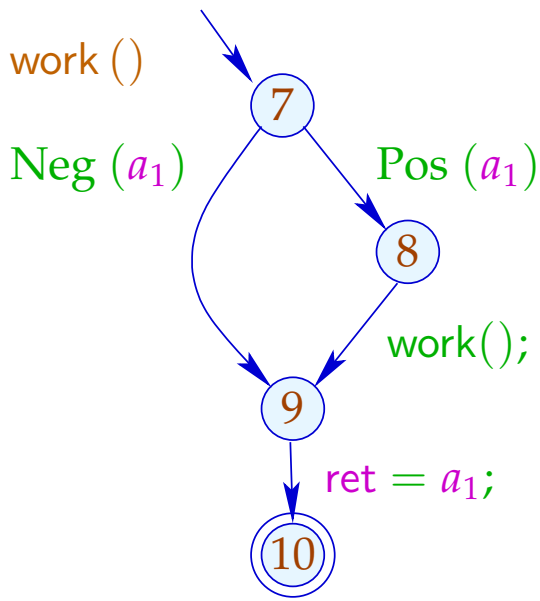
Um die Analyse zu implementieren, müssen wir nur noch den Effekt eines Aufrufs $k = (_, f\ ();, _)$ aus dem Effekt der Prozedur f ermitteln:

$$
\begin{array}{lcll}
[\![k]\!]^\sharp & = & H\ ([\![f]\!]^\sharp) & \text{wobei:} \\
H\ (M) & = & \mathsf{Id}|_{Locals} \oplus \{\mathsf{x} \mapsto (M \circ E_f)\ x \mid x \in Globals\} & \\
E_f\ x & = & \begin{cases} x & \text{falls} \quad x \in Globals \\ 0 & \text{sonst} \end{cases} &
\end{array}
$$

... im Beispiel:

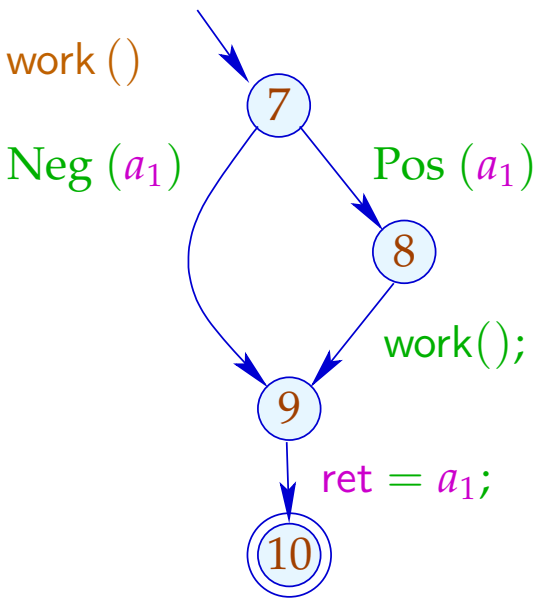
$$
\begin{array}{lrcl}
\text{Falls} & [\![\mathsf{work}]\!]^\sharp & = & \{a_1 \mapsto a_1, \mathsf{ret} \mapsto a_1, t \mapsto t\} \\
\text{dann} & H\,[\![\mathsf{work}]\!]^\sharp & = & \mathsf{Id} \oplus \{a_1 \mapsto a_1, \mathsf{ret} \mapsto a_1\} \\
 & & = & \{a_1 \mapsto a_1, \mathsf{ret} \mapsto a_1, t \mapsto t\}
\end{array}
$$

Damit können wir die Fixpunkt-Iteration durchführen :-)



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

$$
\begin{aligned}
[\![(8,\ldots,9)]\!]^\sharp \circ [\![8]\!]^\sharp &= \{a_1 \mapsto a_1, \mathsf{ret} \mapsto a_1, t \mapsto t\} \circ \\
&\quad \{a_1 \mapsto a_1, \mathsf{ret} \mapsto \mathsf{ret}, t \mapsto t\} \\
&= \{a_1 \mapsto a_1, \mathsf{ret} \mapsto a_1, t \mapsto t\}
\end{aligned}
$$



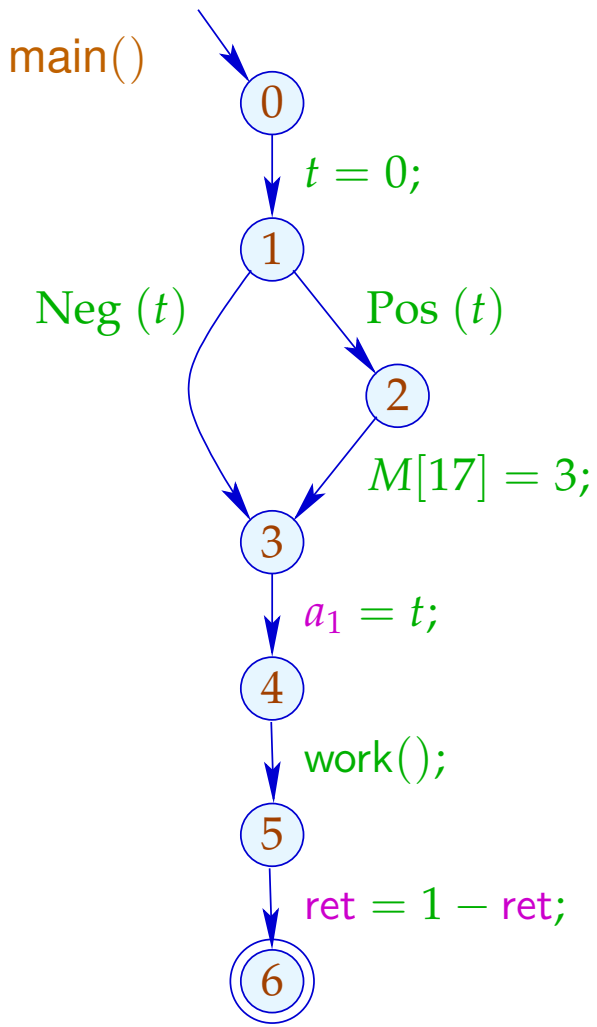
	2
7	$\{a_1 \mapsto a_1, \mathsf{ret} \mapsto \mathsf{ret}, t \mapsto t\}$
9	$\boxed{\{a_1 \mapsto a_1, \mathsf{ret} \mapsto a_1 \sqcup \mathsf{ret}, t \mapsto t\}}$
10	$\boxed{\{a_1 \mapsto a_1, \mathsf{ret} \mapsto a_1, t \mapsto t\}}$
8	$\{a_1 \mapsto a_1, \mathsf{ret} \mapsto \mathsf{ret}, t \mapsto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

$$
\begin{aligned}
[\![(8,\ldots,9)]\!]^\sharp \circ [\![8]\!]^\sharp &= \{a_1 \mapsto a_1, \mathsf{ret} \mapsto a_1, t \mapsto t\} \circ \\
&\quad \{a_1 \mapsto a_1, \mathsf{ret} \mapsto \mathsf{ret}, t \mapsto t\} \\
&= \{a_1 \mapsto a_1, \mathsf{ret} \mapsto a_1, t \mapsto t\}
\end{aligned}
$$

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

$$
\begin{array}{lll}
\mathcal{R}[\mathsf{main}] & \sqsupseteq & \mathsf{enter}^\sharp\ d_0 \\
\mathcal{R}[f] & \sqsupseteq & \mathsf{enter}^\sharp\ (\mathcal{R}[u]) \qquad k = (u, f\ ();, _) \quad \text{Aufruf} \\
\mathcal{R}[v] & \sqsupseteq & \mathcal{R}[f] \qquad v \quad \text{Anfangspunkt von} \quad f \\
\mathcal{R}[v] & \sqsupseteq & [\![k]\!]^\sharp\ (\mathcal{R}[u]) \qquad k = (u, _, v) \quad \text{Kante}
\end{array}
$$

... im Beispiel:



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

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 $\mathbb{M} \subseteq \mathbb{D} \to \mathbb{D}$ 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:

$$
\begin{array}{lll}
[\![v,a]\!]^\sharp & \sqsupseteq & a \qquad\qquad\qquad v \quad \text{Eintrittspunkt} \\
[\![v,a]\!]^\sharp & \sqsupseteq & \mathsf{combine}^\sharp\,([\![u,a]\!], [\![f, \mathsf{enter}^\sharp\,[\![u,a]\!]^\sharp]\!]^\sharp) \\
 & & \qquad\qquad\qquad (u, f\,();, v) \quad \text{Aufruf} \\
[\![v,a]\!]^\sharp & \sqsupseteq & [\![lab]\!]^\sharp\,[\![u,a]\!]^\sharp \quad k = (u, lab, v) \quad \text{Kante} \\
[\![f,a]\!]^\sharp & \sqsupseteq & [\![stop_f, a]\!]^\sharp \qquad stop_f \quad \text{Endpunkt von} \quad f
\end{array}
$$

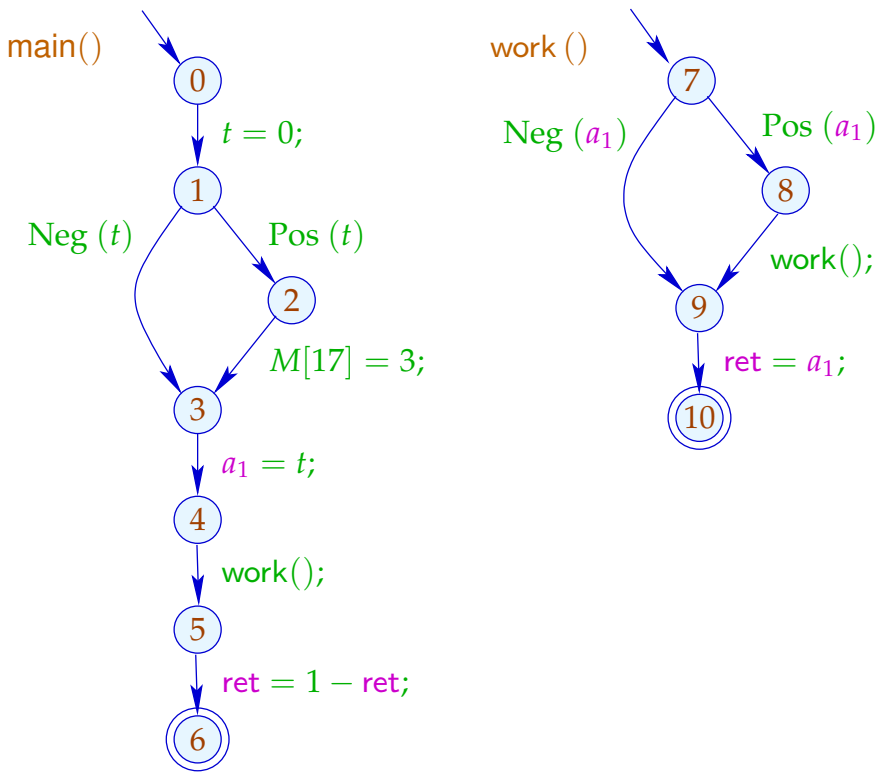
// $[\![v,a]\!]^\sharp$ == Wert des Effekts für das Argument a .

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 $[\![\mathsf{main}(), a_0]\!]^\sharp$ benötigt werden $\Longrightarrow$ 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	$\top$	$\top$	$\top$	$\top$
1	$\top$	$\top$	$\top$	$\top$
2	$\top$	$\top$	$\bot$	
3	$\top$	$\top$	$\top$	$\top$
4	$\top$	$\top$	0	$\top$
7	0	$\top$	0	$\top$
8	0	$\top$	$\bot$	
9	0	$\top$	0	$\top$
10	0	$\top$	0	0
5	$\top$	$\top$	0	0
main()	$\top$	$\top$	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:

$\rightarrow$ Berechne die Menge aller erreichbaren Aufrufkeller!

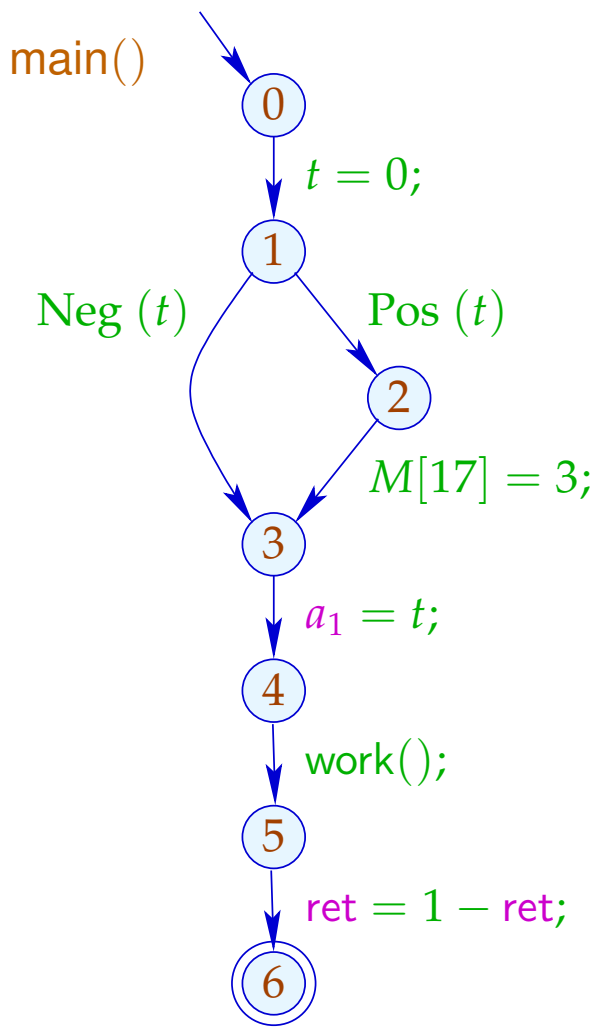
$\rightarrow$ Diese ist i.a. unendlich :-(

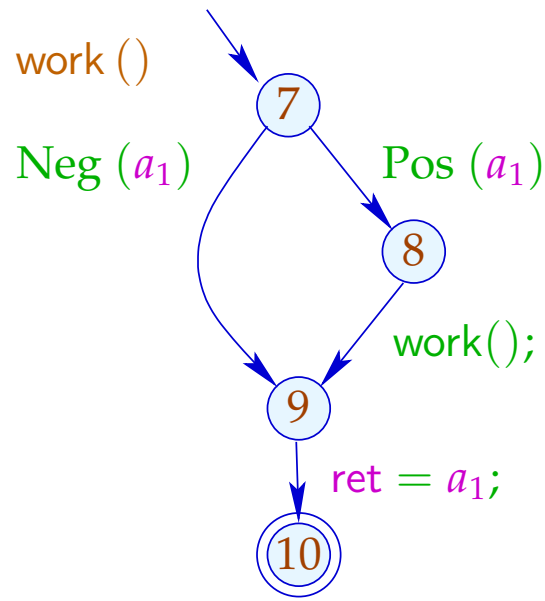
$\rightarrow$ Behandle Keller bis zu einer festen Tiefe d exakt! Behalte von längeren Kellern nur das obere Ende der Länge d :-)

$\rightarrow$ Wichtiger Spezialfall: $d = 0$.

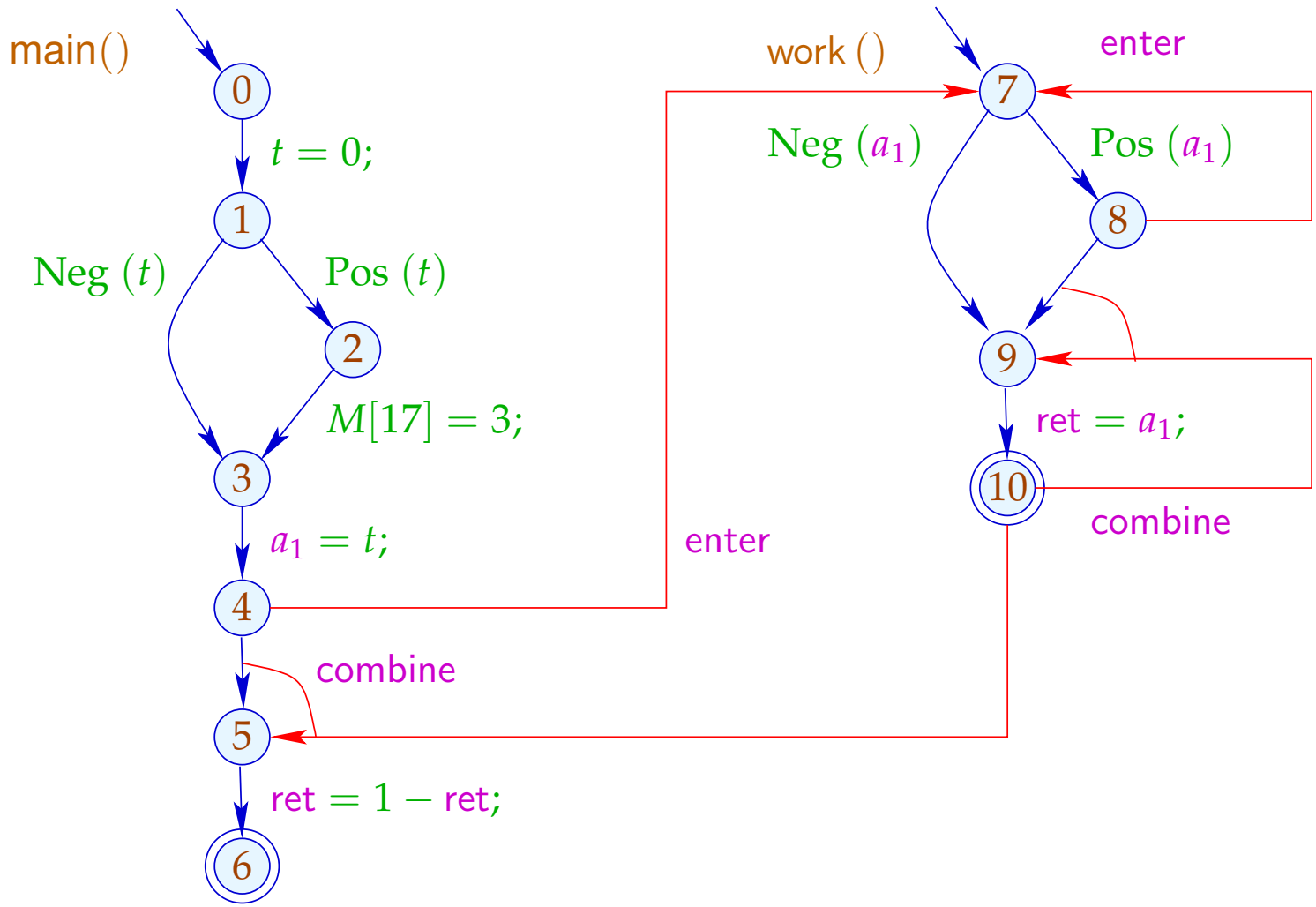
$\Longrightarrow$ Betrachte nur die obersten Kellerrahmen ...

... im Beispiel:


main()
0
$t = 0;$
1
Neg (t)
Pos (t)
2
$M[17] = 3;$
3
$a_1 = t;$
4
work();
5
ret $= 1 -$ ret;
6


work ()
7
Neg (a_1)
Pos (a_1)
8
work();
9
ret $= a_1;$
10

... im Beispiel:


main()
0
$t = 0;$
1
Neg (t)
Pos (t)
2
$M[17] = 3;$
3
$a_1 = t;$
4
combine
5
ret $= 1 -$ ret;
6
work ()
enter
7
Neg (a_1)
Pos (a_1)
8
9
ret $= a_1;$
10
combine
enter

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

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

$$\mathcal{R}[7] \sqsupseteq \mathsf{enter}^\sharp\,(\mathcal{R}[4])$$

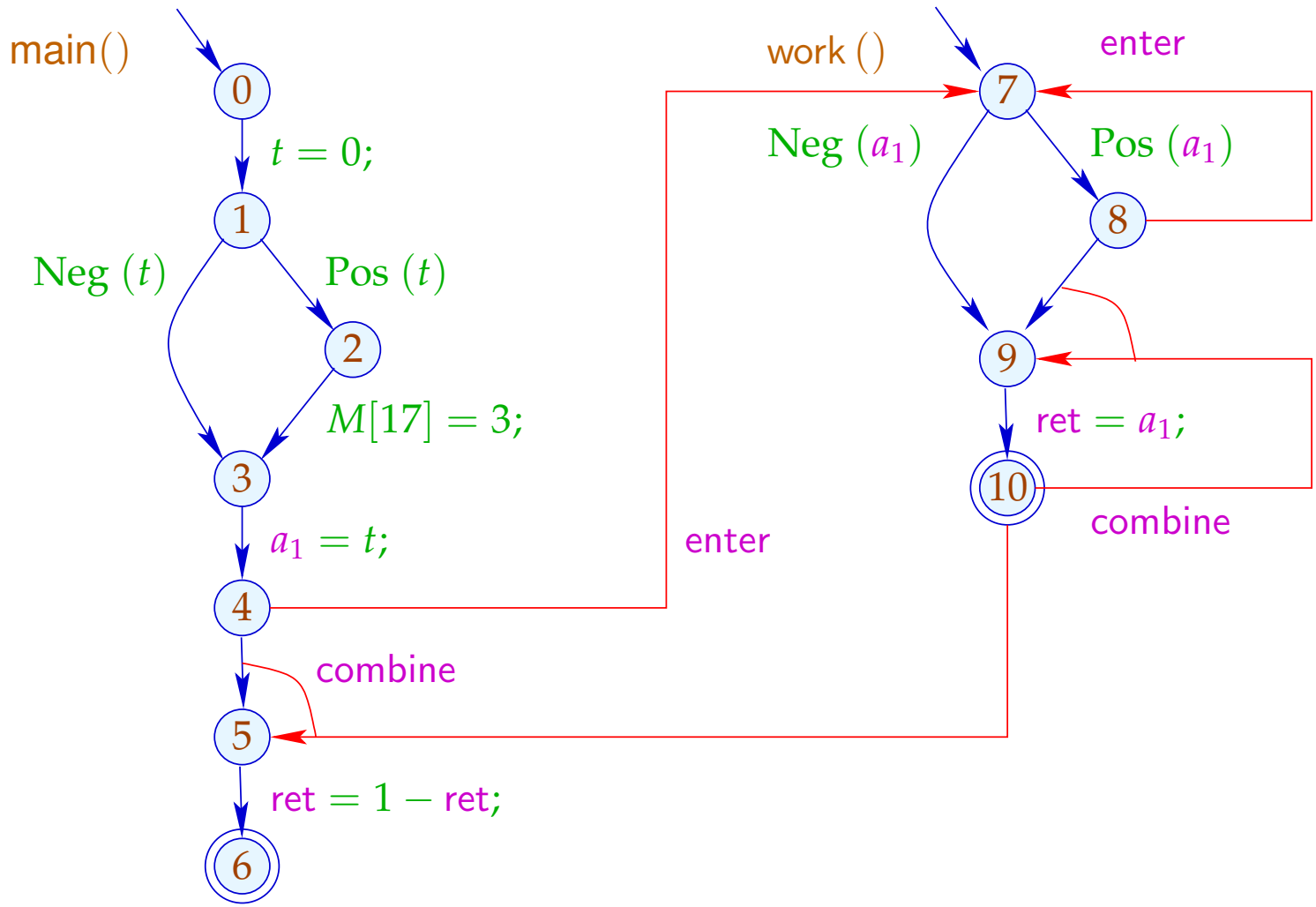
$$\mathcal{R}[7] \sqsupseteq \mathsf{enter}^\sharp\,(\mathcal{R}[8])$$

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

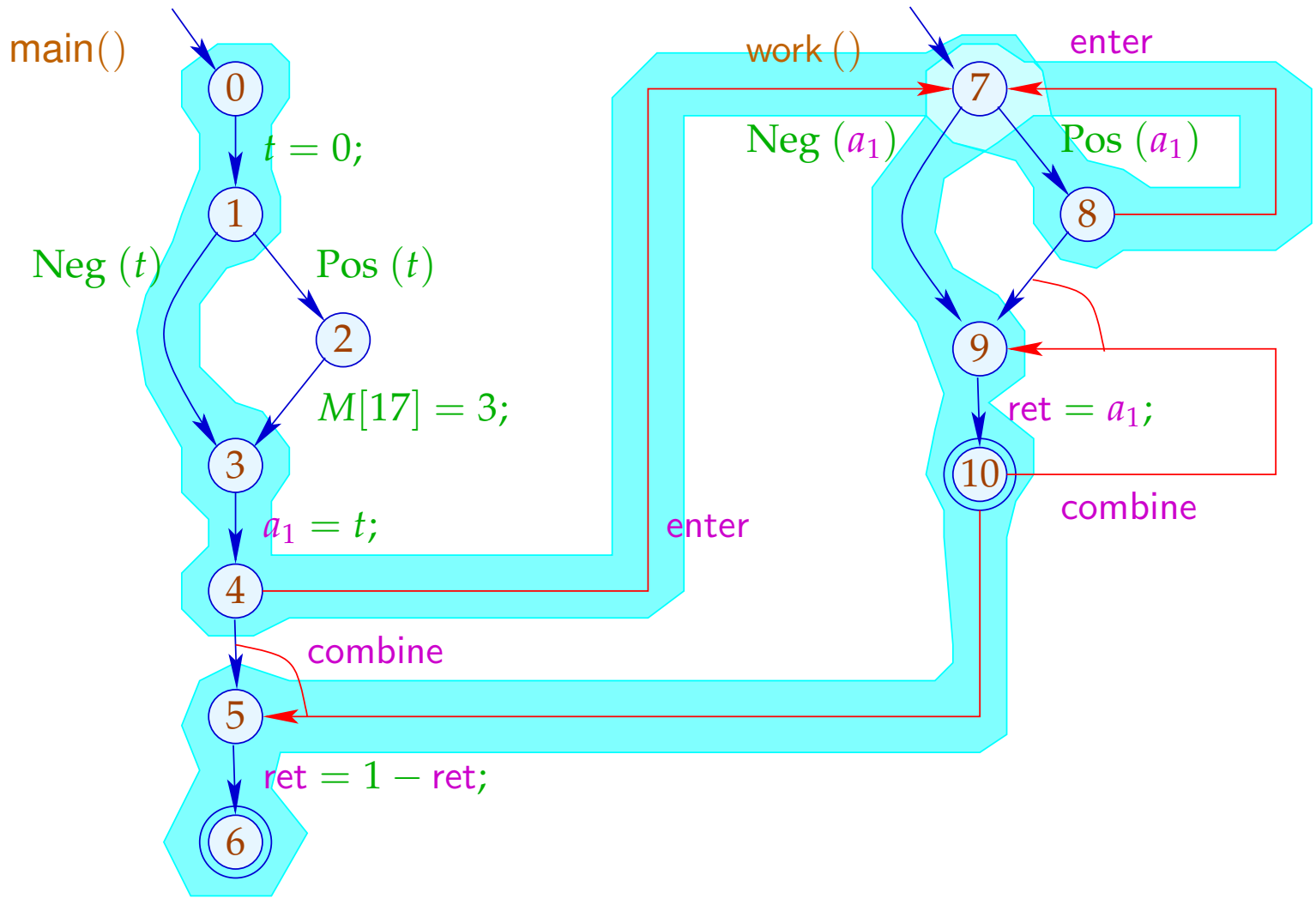
Achtung:

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 Ergebnissse 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 :-)