2. Idee: Beseitigung von Endrekursion

$$
\begin{array}{l}
f\,()\ \{\quad \mathsf{int}\ \ b;\\
\qquad \mathsf{if}\ \ (a_2 \leq 1)\ \ \{\ \ \mathsf{ret} = a_1;\ \mathsf{goto}\ \ \_exit;\ \ \ \}\\
\qquad b = a_1 \cdot a_2;\\
\qquad a_2 = a_2 - 1;\\
\qquad a_1 = b;\\
\qquad f\,();\\
\_exit\ \ :\\
\}
\end{array}
$$

Nach dem Prozeduraufruf gibt es im Rumpf nichts mehr zu tun.

$\Longrightarrow$ Wir könnten ihn direkt anspringen :-)

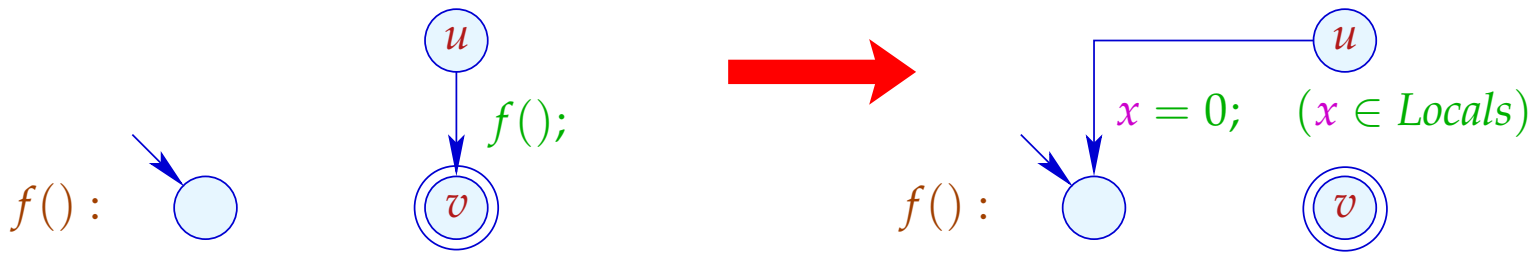
... nachdem wir die lokalen Variablen auf 0 gesetzt haben.

... das liefert im Beispiel:

```
f () {   int b;
 _f :    if (a2 ≤ 1) { ret = a1; goto _exit; }
         b = a1 · a2;
         a2 = a2 − 1;
         a1 = b;
         b = 0; goto _f;
 _exit :
}
```

// Das funktioniert, weil wir Referenzen auf Variablen
// verbieten.

Transformation 11:



Achtung:

- → Diese Optimierung ist besonders wichtig bei Programmiersprachen ohne Iterationskonstrukte !!!
- → Duplizieren von Code ist nicht erforderlich :-)
- → Variablen brauchen nicht umbenannt zu werden :-)
- → Die Optimierung hilft auch bei nicht-rekursiven Endaufrufen :-)
- → Der entstehende Code kann Sprünge aus dem Rumpf einer Funktion in eine andere enthalten ???

Exkurs 4: Interprozedurale Analyse

Bisher können wir nur jede Prozedur einzeln analysieren.

- $\rightarrow$ Die Kosten halten sich in Grenzen :-)
- $\rightarrow$ Die Techniken funktionieren auch bei getrennter Übersetzung :-)
- $\rightarrow$ An Prozeduraufrufen müssen wir das Schlimmste annehmen :-(
- $\rightarrow$ Konstantenpropagation funktioniert nur für lokale Konstanten :-((

Frage:

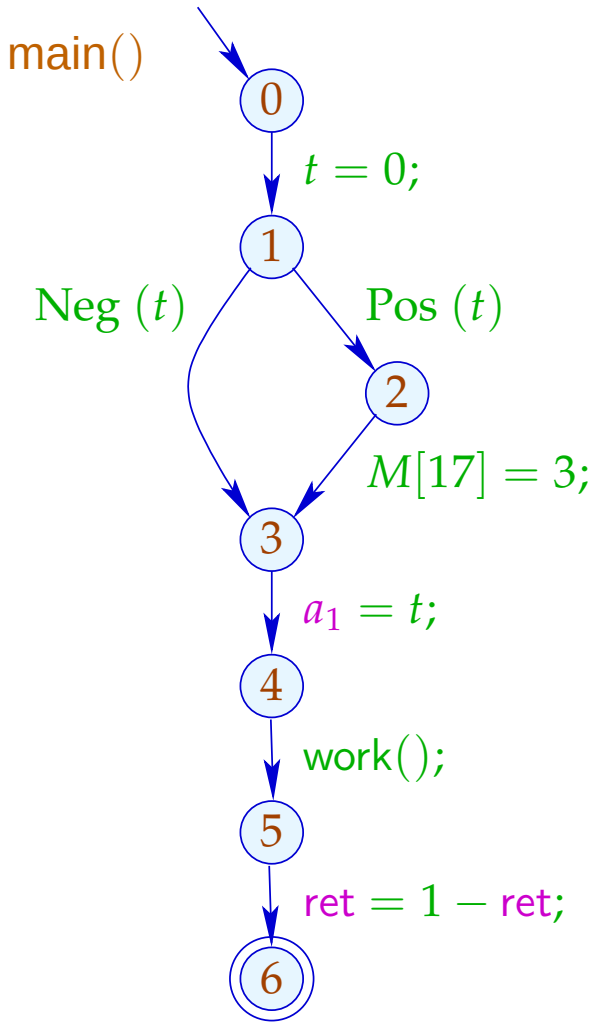
Wie analysiert man rekursive Programme ???

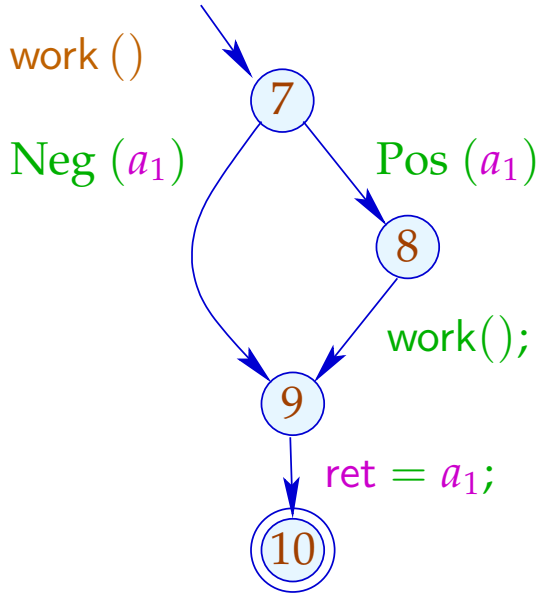
Beispiel: Konstantenpropagation

```
main() { int t;
         t = 0;
         if (t) M[17] = 3;
         a1 = t;
         work ();
         ret = 1 − ret;
}
```

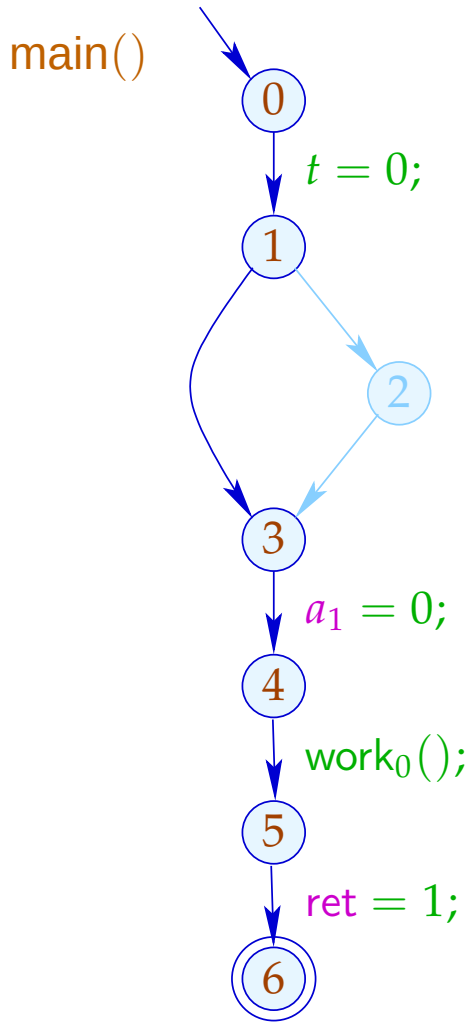
```
work() {
         if (a1) work();
         ret = a1;
}
```

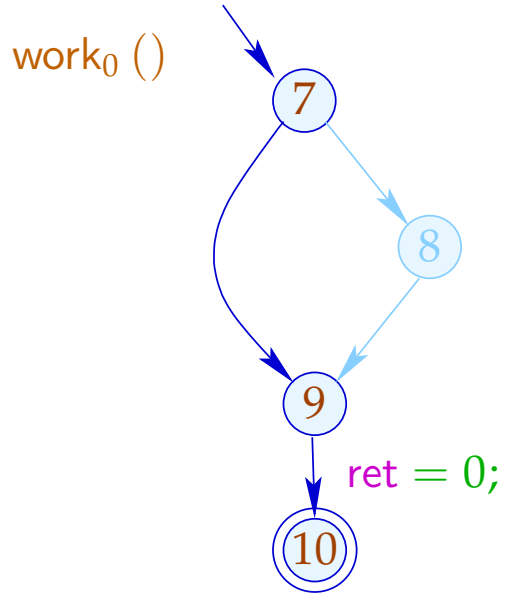
Beispiel: Konstantenpropagation





Beispiel: Konstantenpropagation





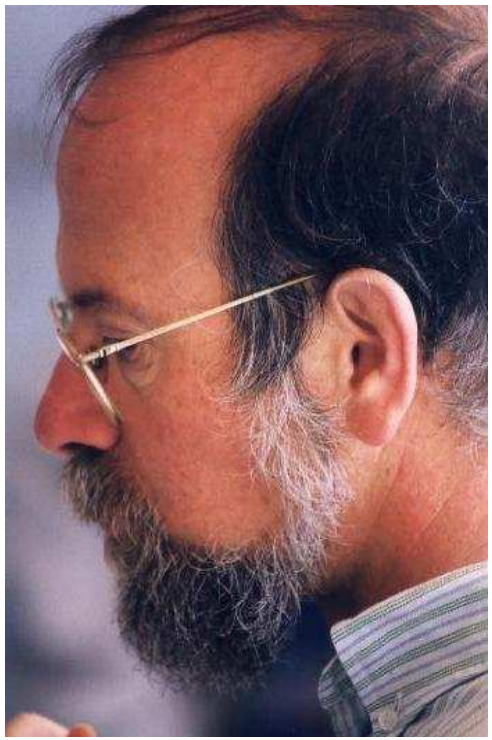
(1) Funktionaler Ansatz:

Sei $\mathbb{D}$ ein vollständiger Verband von (abstrakten) Zuständen.

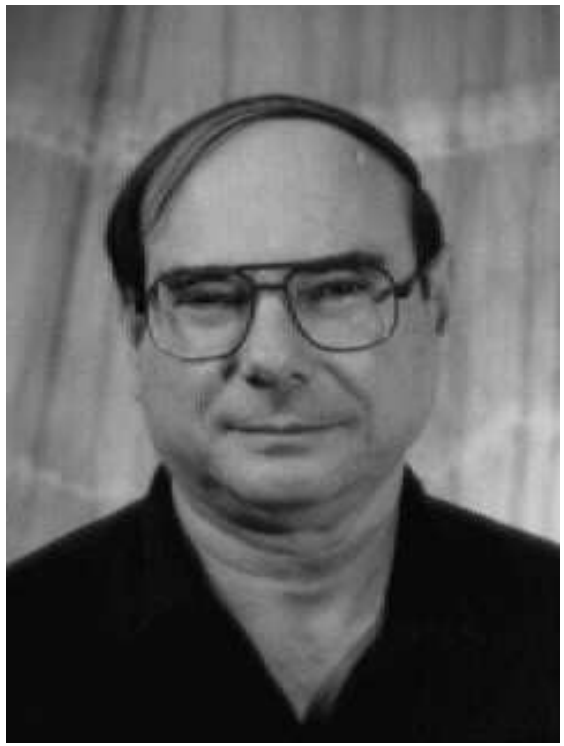
Idee:

Repräsentiere den Effekt von $f()$ durch eine Funktion:

$$[\![f]\!]^\sharp \; : \; \mathbb{D} \to \mathbb{D}$$



Micha Sharir, Tel Aviv University



Amir Pnueli, Weizmann Institute

Um den Effekt einer Aufrufskante $k = (u, f\ (); , v)$ zu ermitteln, benötigen wir abstrakte Funktionen:

$$\begin{array}{lcl} \mathsf{enter}^\sharp & : & \mathbb{D} \to \mathbb{D} \\ \mathsf{combine}^\sharp & : & \mathbb{D}^2 \to \mathbb{D} \end{array}$$

Damit erhalten wir:

$$[\![k]\!]^\sharp\ D \quad = \quad \mathsf{combine}^\sharp\ (D, [\![f]\!]^\sharp\ (\mathsf{enter}^\sharp\ D))$$

... für Konstantenpropagation:

$$\mathbb{D} = (\mathit{Vars} \to \mathbb{Z}^\top)_\perp$$

$$\mathsf{enter}^\sharp\ D = \begin{cases} \perp & \text{falls} \quad D = \perp \\ D|_{\mathit{Globals}} \oplus \{x \mapsto 0 \mid x \in \mathit{Locals}\} & \text{sonst} \end{cases}$$

$$\mathsf{combine}^\sharp\ (D_1, D_2) = \begin{cases} \perp & \text{falls} \quad D_1 = \perp \vee D_2 = \perp \\ D_1|_{\mathit{Locals}} \oplus D_2|_{\mathit{Globals}} & \text{sonst} \end{cases}$$

Um die Effekte $[\![f]\!]^\sharp$ zu ermitteln, stellen wir ein Ungleichungssystem über dem vollständigen Verband $\mathbb{D} \to \mathbb{D}$ auf:

$$
\begin{array}{lcll}
[\![v]\!]^\sharp & \sqsupseteq & \mathsf{Id} & v \quad \text{Eintrittspunkt} \\
[\![v]\!]^\sharp & \sqsupseteq & [\![k]\!]^\sharp \circ [\![u]\!]^\sharp & k = (u, \_, v) \quad \text{Kante} \\
[\![f]\!]^\sharp & \sqsupseteq & [\![stop_f]\!]^\sharp & stop_f \quad \text{Endpunkt von} \quad f
\end{array}
$$

$[\![v]\!]^\sharp : \mathbb{D} \to \mathbb{D}$ beschreibt den Effekt aller Präfixe der Berechnungswälder w einer Prozedur, die vom Eintrittspunkt nach v führen :-)

Probleme:

- Wie beschreibt man eine Funktion $f : \mathbb{D} \to \mathbb{D}$???
- Ist $\#\mathbb{D} = \infty$, hat $\mathbb{D} \to \mathbb{D}$ unendliche aufsteigende Ketten :-(

Vereinfachung: Kopier-Konstanten

$\to$ Bedingungen interpretieren wir wie ein $;$:-)

$\to$ Wir behandeln exakt nur Zuweisungen $x = e;$ mit $e \in \mathit{Vars} \cup \mathbb{Z}$:-)

Beobachtung:

→ Die Effekte von Zuweisungen sind:

$$[\![x = e;]\!]^\sharp\, D \quad = \quad \begin{cases} D \oplus \{x \mapsto c\} & \text{falls} \quad e = c \in \mathbb{Z} \\ D \oplus \{x \mapsto (D\ y)\} & \text{falls} \quad e = y \in \mathit{Vars} \\ D \oplus \{x \mapsto \top\} & \text{sonst} \end{cases}$$

→ Sei $\mathbb{V}$ die (endliche !!!) Menge der konstanten rechten Seiten. Dann haben Variablen stets Werte aus $\mathbb{V}^\top$:-))

→ Die auftretenden Effekte sind enthalten in

$$\mathbb{D}_f \to \mathbb{D}_f \qquad \text{mit} \qquad \mathbb{D}_f = (\mathit{Vars} \to \mathbb{V}^\top)_\perp$$

→ Dieser Verband ist riesig, aber endlich !!!

Verbesserung:

→ Nicht alle Funktionen aus $\mathbb{D}_f \to \mathbb{D}_f$ kommen wirklich vor :-)

→ Alle vorkommenden Funktionen $\lambda D.\, \bot \neq M$ sind von der Form:

$$\begin{array}{lcll} M & = & \{x \mapsto (b_x \sqcup \bigsqcup_{y \in I_x} y) \mid x \in \mathit{Vars}\} & \text{wobei:} \\ M\ D & = & \{x \mapsto (b_x \sqcup \bigsqcup_{y \in I_x} D\ y) \mid x \in \mathit{Vars}\} & \text{für} \quad D \neq \bot \end{array}$$

→ Sei $\mathbb{M}$ die Menge aller dieser Funktionen. Dann gilt für $M_1, M_2 \in \mathbb{M}$ $(M_1 \neq \lambda\, D.\, \bot \neq M_2)$:

$$(M_1 \sqcup M_2)\ x \quad = \quad (M_1\ x) \sqcup (M_2\ x)$$

→ Für $k = \#\mathit{Vars}$ hat $\mathbb{M}$ die Höhe $\mathcal{O}(k^2)$:-)