3.2 Die funktionale Ansatz

Idee: Berechne Transferfunktionen f_p für Prozeduren $p()$

Ansatz: Summary-Gleichungssystem

$$Y_{entry} = id$$

$$Y_b = \bigsqcup \{ f_{b'} \circ Y_{b'} \mid (b', b) \in F \}$$

mit

$f_b := callret(Y_q)$, falls $b = [q()]^{call}_{return}$

$f_b :=$ wie in f angegeben, sonst.

Dabei ist

$callret(Y_q) : D \to D$ mit

$callret(Y_q)(d) := f_{return}(d, Y_q(f_{call}(d)))$

$$Y_p = f_{end} \circ Y_{end}.$$

Da Blöcke für Prozeduraufrufe zwei Labels haben, wird in Folgenden die Aufstellung des Summary-Gleichungssystems, genauer die Zahl der Variablen und Transferfunktionen, diskutiert.

Betrachte die Prozedur

```
proc [p()]^1 begin
    [x := a]^2;
    [q()]^3_4;
    [t := a']^5
    :
[end]
```

Seien f_{call} und f_{return} die Transferfunktionen, die Block $b = [q()]^3_4$ zugeordnet sind.

- Das oben definierte Gleichungssystem setzt für jeden Block $b = [q()]^3_4$ eine Variable Y_b ein, die wir mit dem Call-Label identifizieren.

  Im Beispiel also $Y_b = Y_3$ mit $f_3 = callret(Y_q)$.

Damit ergibt sich für Block 5 die Gleichung

$$y_5 = f_3 \circ y_3 = callret(y_4) \circ y_3.$$

- Alternativ (und <u>äquivalent</u>) lassen sich <u>zwei</u> Variablen y_3 und y_4 für $b = [q()]_4^3$ definieren.

Dann wird y_3 die Transferfunktion

$$f_3 = y_4 \circ fcall$$

zugewiesen. Für y_4 erhält man somit die Gleichung

$$y_4 = f_3 \circ y_3 = (y_4 \circ fcall) \circ y_3$$

Als Transferfunktion für y_4 wird

$$f_4 = freturn$$ verwendet.

Dann erhält man als Gleichung für Block 5:

$$y_5 = freturn(y_3(-), y_4(-)),$$

was äquivalent ist zu

$$y_5 = freturn(y_3(-), y_4(fcall(y_3(-))))$$
$$= callret(y_4) \circ y_3.$$

- Schließlich (und ebenfalls <u>äquivalent</u>) ist es möglich, Variablen y_3 und y_4 zu definieren mit

$$f_3 = callret(y_4)$$
$$f_4 = id.$$

Damit ergibt sich

$$y_4 = f_3 \circ y_3 = callret(y_4) \circ y_3$$
$$y_5 = f_4 \circ y_4 = id \circ y_4$$

Wiederum gilt

$$y_5 = callret(y_4) \circ y_3.$$

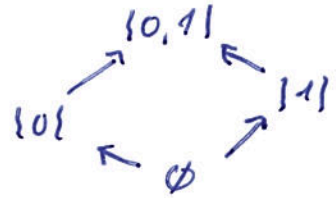
Beispiel: Reachable-Values-Analyse

Gegeben: Eine Boolesche Variable x

Berechne: Transferverhalten der Prozedur pos() auf den Wertemengen für x.

```
proc [pos]^1 begin
  if [x=0]^2 then
    [assert(x=0)]^3;
    [x := -x]^4;
    [pos()]^5_6;
  else
    [assert(x=1)]^7
[end]^8
```

Vollständige Verband der Wertemengen für x:



Transferfunktionen:

$f_1 := id = : f_2 = : f_{callpos} = : f_8$

mit

$\emptyset \mapsto \emptyset$	$\{1\} \mapsto \{1\}$
$\{0\} \mapsto \{0\}$	$\{0,1\} \mapsto \{0,1\}$

$f_3 := f_{get0}$:

$\emptyset \mapsto \emptyset$	
$\{0\} \mapsto \{0\}$	
$\{1\} \mapsto \emptyset$	
$\{0,1\} \mapsto \{0\}$	

$f_7 := f_{get1}$:

$\emptyset \mapsto \emptyset$	
$\{0\} \mapsto \emptyset$	
$\{1\} \mapsto \{1\}$	
$\{0,1\} \mapsto \{1\}$.	

$f_4 := f_{invert}$:

$\emptyset \mapsto \emptyset$	
$\{0\} \mapsto \{1\}$	
$\{1\} \mapsto \{0\}$	
$\{0,1\} \mapsto \{0,1\}$	

$f_{returnpos}(d_1, d_2) := d_2$

Weitere Funktionen, die für die Fixpunktberechnung relevant sind:

f_{make1}:

$\emptyset \mapsto \emptyset$	
$\{0\} \mapsto \{1\}$	
$\{1\} \mapsto \{1\}$	
$\{0,1\} \mapsto \{1\}$	

$f_{invert0}$:

$\emptyset \mapsto \emptyset$	
$\{0\} \mapsto \{1\}$	
$\{1\} \mapsto \emptyset$	
$\{0,1\} \mapsto \{1\}$	

$f_\perp$:

$\emptyset \mapsto \emptyset$	
$\{0\} \mapsto \emptyset$	
$\{1\} \mapsto \emptyset$	
$\{0,1\} \mapsto \emptyset$	

Folgende Gleichungen gelten:

- $f_{invert} \circ f_{get0} = f_{invert0}$
- $f_{get1} \circ f_{invert0} = f_{invert0}$
- $f_{make1} \circ f_{invert0} = f_{invert0}$.
- $f_{invert0} \sqcup f_{get1} = f_{make1}$

- $callret(y_{pos})$
  $= f_{returnpos}(-, y_{pos}(f_{callpos}(-)))$
  $= y_{pos} \circ id$
  $= y_{pos}$

Summary – Gleichungssystem:

$$
\begin{aligned}
Y_1 &= id \\
Y_2 &= f_1 \circ Y_1 \\
Y_3 &= f_2 \circ Y_2 \\
Y_4 &= f_3 \circ Y_3 \\
Y_5 &= f_4 \circ Y_4 \\
Y_7 &= f_2 \circ Y_2 \\
Y_8 &= (f_5 \circ Y_5) \sqcup (f_7 \circ Y_7) \\
Y_{pos} &= f_8 \circ Y_8
\end{aligned}
$$

Mit den Definitionen der Transferfunktionen ergibt sich

$$
\begin{aligned}
Y_1 &= id \\
Y_2 &= id \circ Y_1 \\
Y_3 &= id \circ Y_2 \\
Y_4 &= f_{get0} \circ Y_3 \\
Y_5 &= f_{invot} \circ Y_4 \\
Y_7 &= id \circ Y_2 \\
Y_8 &= (callat(Y_{pos}) \circ Y_5) \\
&\quad \sqcup (f_{get1} \circ Y_7) \\
&= (Y_{pos} \circ Y_5) \sqcup (f_{get1} \circ Y_7) \\
Y_{pos} &= id \circ Y_8
\end{aligned}
$$

Nach Vereinfachung (Entfernen von Identitäten) ergibt sich:

$$
\begin{aligned}
Y_3 &= id \\
Y_4 &= f_{get0} \circ Y_3 \\
Y_5 &= f_{invot} \circ Y_4 \\
Y_7 &= id \\
Y_{pos} &= (Y_{pos} \circ Y_5) \sqcup (f_{get1} \circ Y_7)
\end{aligned}
$$

Iteration	Y_3	Y_4	Y_5	Y_7	Y_{pos}
0	$f_\perp$	$f_\perp$	$f_\perp$	$f_\perp$	$f_\perp$
1	id	$f_\perp$	$f_\perp$	id	$f_\perp$
2	id	f_{get0}	$f_\perp$	id	f_{get1}
3	id	f_{get0}	f_{invot0} *	id	f_{get1} **
4	id	f_{get0}	f_{invot0}	id	f_{make1} ***
5	id	f_{get0}	f_{invot0}	id	f_{make1} ****

* Da $f_{invot} \circ f_{get0} = f_{invot0}$.

** Da
$(f_{get1} \circ f_\perp) \sqcup (f_{get1} \circ id)$
$= f_\perp \sqcup f_{get1}$
$= f_{get1}$.

*** Da
$(f_{get1} \circ f_{invot0}) \sqcup f_{get1}$
$= f_{invot0} \sqcup f_{get1}$
$= f_{make1}$.

**** Da
$(f_{make1} \circ f_{invot0}) \sqcup f_{get1} = f_{invot0} \sqcup f_{get1} = f_{make1}$

Sind die Transferfunktionen berechnet,
lässt sich auch für rekursive Datenflusssysteme
ein Gleichungssystem aufstellen.

Betrachte $S = (G, (D, \leq), i, f)$ mit $G = (B, E, F, IF)$.
Das Gleichungssystem zur Datenflussanalyse ist

$$X_b = \bigsqcup \{ f_{b'}(X_{b'}) \mid (b', b) \in F \}, \text{ falls } b \notin E.$$

Dabei ist

↳ $f_{b'}(X_{b'})$ wie in f angegeben, falls b gewöhnlicher Block

↳ $f_{b'}(X_{b'}) = f_{return}(X_{b'}, f_p(f_{call_p}(X_{b'})))$,
falls $b' = [p()]^{call}_{return}$.

$$X_b = \bigsqcup \{ f_{call_p}(X_{b'}) \mid (b', b, *, *) \in IF \}, \text{ falls } b \in E \setminus E_{main}$$

$X_b = i$, falls $b \in E_{main}$.

<u>Satz (Sharir & Pnueli '87)</u>

Sei $S = (G, (D, \leq), i, f)$ ein rekursives Datenflusssystem.
Sei $lfp(g_S) = (X_1^{LFP}, \ldots, X_{|B|}^{LFP})$ die Fixpunktlösung
des assoziierten Gleichungssystems.
Sei ferner $JOVP(S) = (X_1^{JOVP}, \ldots, X_{|B|}^{JOVP})$ die JOVP-Lösung.

(a) Für alle $b \in B$ gilt $X_b^{JOVP} \leq X_b^{LFP}$.

(b) Falls alle Transferfunktionen distributiv sind,
gilt sogar $X_b^{JOVP} = X_b^{LFP}$ für alle $b \in B$.

<u>Korollar</u>:
Das <u>Control-Stack-Reachability-Problem</u>

<u>Gegeben</u>: Rekursives Boolesches Programm prog, Block b in prog.
<u>Problem</u>: Gibt es einen Ablauf, der zu b führt?

ist <u>entscheidbar</u>.

3.3 Der Call-String-Ansatz

Kontext(in)sensitivität:

- Der funktionale Ansatz berücksichtigt keine Information über den Kontext, in dem eine Prozedur aufgerufen wird.
  ↳ Falls $p()$ von $q()$ aus aufgerufen wird, könnte x immer 1 sein.
- Das macht die Analyse ggfs. unpräzise.

Ziel: Entwickle eine kontextsensitive interprozedurale Datenflussanalyse.

Ansatz:
- Erweitere die Domäne der Datenflusswerte um Informationen über den Stackinhalt an.
- Sei $(D, \sqsubseteq)$ der vollständige Verband der Datenflusswerte.
  Sei T die Menge der Prozedurnamen im Programm.
  Nutze die neue Domäne

$$(T^* \to D, \sqsubseteq^*) \text{ mit}$$

$$cs_1 \sqsubseteq^* cs_2, \text{ falls } cs_1(\alpha) \sqsubseteq cs_2(\alpha) \text{ für alle } \alpha \in T^*.$$

Es werden also Call-Strings Datenflussinformationen zugewiesen.
Es lässt sich prüfen, dass $(T^* \to D, \sqsubseteq^*)$ wieder ein vollständiger Verband ist.

Formal: Um ein gegebenes Datenflusssystem $S = (G, (D, \sqsubseteq), c, f)$ unter Berücksichtigung von Call-Strings zu analysieren, modifiziere (1) den Initialwert und (2) die Transferfunktionen.

Zu (1):
Aus $c \in D$ wird $cs_c : T^* \to D$ mit

$$cs_c(\varepsilon) := c \text{ und}$$
$$cs_c(\alpha) := \bot \text{ für } \varepsilon \neq \alpha \in T^*.$$

Zu (2):
Aus $f_b : D \to D$ wird

$$\overline{f}_b : (T^* \to D) \to (T^* \to D).$$

Dabei ist

$\widetilde{f_b}(cs) := f_b \circ cs$, falls b gewöhnlicher Block.

$\widetilde{f_{callp}}(cs) := cs'$ mit $cs'(\gamma.p) := f_{callp}(cs(\gamma))$
$cs'(\alpha) := \bot$ für $\gamma.p \neq \alpha \in T^*$.

$\widetilde{f_{return}}(cs) := cs'$ mit $cs'(\gamma) := f_{return}(cs(\gamma.p))$.

Definition:
Sei $S = (G, (D, \leq), c, f)$ ein rekursives Datenflußsystem.
Dann ist das induzierte Call-String-Gleichungssystem:

$X_b = cs_0$, falls $b \in E_{main}$

$X_b = \bigsqcup \{ \widetilde{f_{b'}}(X_{b'}) \mid (b', b) \in IF$ oder
$(b', b, *, *) \in IF$ und $\widetilde{f_{b'}} = \widetilde{f_{callp}}$
oder $(*, *, b', b) \in IF$ und $\widetilde{f_{b'}} = \widetilde{f_{return}} \}$

Satz:
Die Call-String-Lösung überapproximiert JOVP.

Problem: T^* ist unendlich.

Ansätze: (1) Bounded Call-Strings
- Stelle nur oberste n Elemente des Stacks dar,
  nutze also Call-Strings aus $T^{\leq n} := \bigcup_{i=0}^{n} T^i$
- In der Praxis werden 0 oder 1 verwendet.
- Es gibt Fälle wo ausreichende Call-String-Länge,
  die exakte Analyse garantiert
  (Beschränkte Lösung stimmt überein mit Lösung für alle Call-Strings)

(2) Regular Abstraction
Anstelle von $T^{\leq n}$ nutze endliche Automaten $A^{\leq n}$ der Größe $\leq n$,
um den Stackinhalt darzustellen.

(3) Kontextinformation
Aus irgendeiner Menge I.

Andere Sicht des Call-String-Ansatzes:
n-fache Replizierte Prozedur $p()$ für jeden Call-String.