... im Beispiel:

$$\begin{array}{rcl} \delta_c & = & \{A, D\} \quad = \quad q_0 \\ & = & \delta_A \\ & = & \delta_D \end{array}$$

$$\begin{array}{rcl} \delta_+(q_0, q_0) & = & \{A, D, A + A\} \quad = \quad q_1 \\ & = & \delta_+(q_0, \_) \\ & = & \delta_+(\_, q_0) \end{array}$$

$$\begin{array}{rcl} \delta_M(q_0) & = & \{A, D\} \quad = \quad q_0 \\ & = & \delta_M(q_1) \end{array}$$

Um die Anzahl der Zustände zu reduzieren, haben wir die vollständigen rechten Seiten, die keine echten Teilmuster sind, in den Zuständen weggelassen :-)

Integration der Kostenberechnung:

Problem:

Kosten können (im Prinzip) beliebig groß werden ;-(

Unser FTA besitzt aber nur endlich viele Zustände :-((

Idee:

Pelegri-Lopart 1988

Betrachte nicht absolute Kosten — sondern relative !!!

Eduardo Pelegri-Llopart,
Sun Microsystems, Inc.

Beobachtung:

- In gängigen Prozessoren kann man Werte von jedem Register in jedes andere schieben $\Longrightarrow$

  Die Kosten zwischen Registern differieren nur um eine Konstante :-)

- Komplexe rechte Seiten lassen sich i.a. mittels elementarerer Instruktionen simulieren $\Longrightarrow$

  Die Kosten zwischen Teilausdrücken und Registern differieren nur um eine Konstante :-))

- Die Kostenberechnung ist additiv $\Longrightarrow$

  Wir können statt mit absoluten Kosten-Angaben auch mit Kosten-Differenzen rechnen !!!

  Von diesen gibt es nur endlich viele :-)

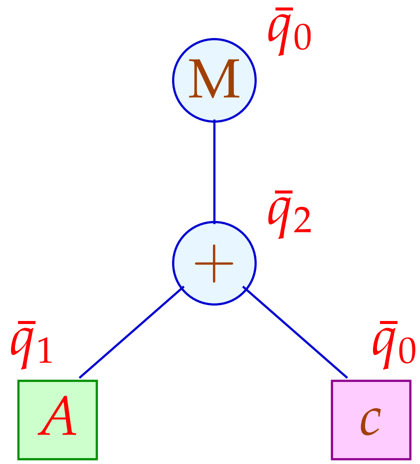
... im Beispiel:

$$\begin{array}{rcl} \delta_c & = & \{A \mapsto 1, D \mapsto 0\} = \bar{q}_0 \\ & = & \delta_D \\ \delta_A & = & \{A \mapsto 0, D \mapsto 1\} = \bar{q}_1 \end{array}$$

$$\begin{array}{rcl} \delta_+(\bar{q}_1, \bar{q}_0) & = & \{A \mapsto 2, D \mapsto 1, A+A \mapsto 0\} = \bar{q}_2 \\ \delta_+(\bar{q}_0, \bar{q}_0) & = & \{A \mapsto 1, D \mapsto 0, A+A \mapsto 1\} = \bar{q}_3 \\ \delta_+(\bar{q}_1, \bar{q}_1) & = & \{A \mapsto 4, D \mapsto 3, A+A \mapsto 0\} = \bar{q}_4 \\ & \ldots & \end{array}$$

$$\begin{array}{rcl} \delta_M(\bar{q}_2) & = & \{A \mapsto 1, D \mapsto 0\} = \bar{q}_0 \\ & = & \delta_M(\bar{q}_i) \qquad , \quad i = 0, \ldots, 4 \end{array}$$

... das liefert die folgende Berechnung:



Für jede Konstanten-Klasse c und jedes Register R in δ_c tabellieren wir die zu wählende billigste Berechnung:

$$c \;:\quad \{A \mapsto 5, 3, D \mapsto 3\}$$

Analog tabellieren wir für jeden Operator a, jedes $\tau \in \bar{Q}^k$ und jedes R in $\delta_a(\tau)$:

M	select_M
$\bar{q}_0$	$\{A \mapsto 5, 1, D \mapsto 1\}$
$\bar{q}_1$	$\{A \mapsto 5, 1, D \mapsto 1\}$
$\bar{q}_2$	$\{A \mapsto 5, 0, D \mapsto 0\}$
$\bar{q}_3$	$\{A \mapsto 5, 1, D \mapsto 1\}$
$\bar{q}_4$	$\{A \mapsto 5, 0, D \mapsto 0\}$

Für "$+$" ist die Tabelle besonders einfach:

$+$	$\bar{q}_j$
$\bar{q}_i$	$\{A \mapsto 5, 3, D \mapsto 3\}$

Problem:

- Für reale Instruktionssätze benötigt man leicht um die 1000 Zustände.
- Die Tabellen für mehrstellige Operatoren werden riesig :-(

$\Longrightarrow$ Wir benötigen Verfahren der Tabellen-Komprimierung ...

Tabellen-Kompression:

Die Tabelle für "$+$" sieht im Beispiel so aus:

$+$	$\bar{q}_0$	$\bar{q}_1$	$\bar{q}_2$	$\bar{q}_3$	$\bar{q}_4$
$\bar{q}_0$	$\bar{q}_3$	$\bar{q}_2$	$\bar{q}_3$	$\bar{q}_3$	$\bar{q}_3$
$\bar{q}_1$	$\bar{q}_2$	$\bar{q}_4$	$\bar{q}_2$	$\bar{q}_2$	$\bar{q}_2$
$\bar{q}_2$	$\bar{q}_3$	$\bar{q}_2$	$\bar{q}_3$	$\bar{q}_3$	$\bar{q}_3$
$\bar{q}_3$	$\bar{q}_3$	$\bar{q}_2$	$\bar{q}_3$	$\bar{q}_3$	$\bar{q}_3$
$\bar{q}_4$	$\bar{q}_3$	$\bar{q}_2$	$\bar{q}_3$	$\bar{q}_3$	$\bar{q}_3$

Die meisten Zeilen / Spalten sind offenbar ganz ähnlich ;-)

Idee 1: Äquivalenzklassen

Wir setzen $q \equiv_a q'$, genau dann wenn

$$\forall\, p: \quad \delta_a(q,p) = \delta_a(q',p) \quad \wedge \quad \delta_a(p,q) = \delta_a(p,q')$$
$$\wedge \quad \text{select}_a(q,p) = \text{select}_a(q',p) \quad \wedge \quad \text{select}_a(p,q) = \text{select}_a(p,q')$$

Im Beispiel:

$$Q_1 = \{\bar{q}_0, \bar{q}_2, \bar{q}_3, \bar{q}_4\}$$
$$Q_2 = \{\bar{q}_1\}$$

mit:

$+$	Q_1	Q_2
Q_1	$\bar{q}_3$	$\bar{q}_2$
Q_2	$\bar{q}_2$	$\bar{q}_4$

Idee 2: Zeilenverschiebung

Sind viele Einträge gleich (im Beispiel etwa default $= \bar{q}_3$), genügt es, die übrigen Einträge zu speichern ;-)

Im Beispiel:

$+$	$\bar{q}_0$	$\bar{q}_1$	$\bar{q}_2$	$\bar{q}_3$	$\bar{q}_4$
$\bar{q}_0$		$\bar{q}_2$			
$\bar{q}_1$	$\bar{q}_2$	$\bar{q}_4$	$\bar{q}_2$	$\bar{q}_2$	$\bar{q}_2$
$\bar{q}_2$		$\bar{q}_2$			
$\bar{q}_3$		$\bar{q}_2$			
$\bar{q}_4$		$\bar{q}_2$			

Dann legen wir:

(1) gleiche Zeilen übereinander;

(2) verschiedene (Klassen von) Zeilen auf Lücke verschoben übereinander:

	$\bar{q}_0$	$\bar{q}_1$	$\bar{q}_2$	$\bar{q}_3$	$\bar{q}_4$
class	0	1	0	0	0

	0	1
disp	0	2

	0	1	2	3	4	5	6
A	$\bar{q}_2$	$\bar{q}_2$	$\bar{q}_4$	$\bar{q}_2$	$\bar{q}_2$	$\bar{q}_2$	$\bar{q}_2$
valid	0	0	1	1	1	1	1

Für jeden Eintrag im ein-dimensionalen Feld A vermerken wir in valid , zu welcher Zeile der Eintrag gehört ...

Ein Feld-Zugriff $\delta_+(\bar{q}_i, \bar{q}_j)$ wird dann so realisiert:

$$\begin{array}{lll} \delta_+(\bar{q}_i, \bar{q}_j) & = & \mathsf{let} \quad c = \mathsf{class}[\bar{q}_i]; \\ & & \qquad\; d = \mathsf{disp}[c]; \\ & & \mathsf{in} \;\; \mathsf{if} \;\; (\mathsf{valid}[d+j] \equiv c) \\ & & \qquad \mathsf{then} \;\; \mathsf{A}[d+j] \\ & & \qquad \mathsf{else} \;\; \mathsf{default} \\ & & \mathsf{end} \end{array}$$

Reinhard Wilhelm, Saarbrücken

Diskussion:

- Die Tabellen werden i.a. erheblich kleiner.
- Dafür werden Tabellenzugriffe etwas teurer.
- Das Verfahren versagt in einigen (theoretischen) Fällen.
- Dann bleibt immer noch das dynamische Verfahren ...

  möglicherweise mit Caching der einmal berechneten Werte, um unnötige Mehrfachberechnungen zu vermeiden :-)