

## Kostenkalkulation:

$$c_t[s] = c_{t_1}[s_1] + \dots + c_{t_k}[s_k] \quad \text{falls } s = a(s_1, \dots, s_k), t = a(t_1, \dots, t_k)$$

$$c_t[R] = \bigcap \{c[R, s] + c_t[s] \mid s \in Q(t)\} \quad \text{wobei}$$

$$c[R, s] \leq c[r] \quad \text{falls } r : R \rightarrow s$$

$$c[R, s] \leq c[r] + c[R', s] \quad \text{falls } r : R \rightarrow R'$$

Das Constraint-System für  $c[R, s]$  kann in Zeit  $\mathcal{O}(n \cdot \log n)$  gelöst werden — falls  $n$  die Anzahl der Paare  $R, s$  ist :-)

Für jedes  $R, s$  liefert die Fixpunkt-Berechnung eine Folge:

$$\pi[R, s] : R \Rightarrow R_1 \Rightarrow \dots \Rightarrow R_k \Rightarrow s$$

deren Kosten gerade  $c[R, s]$  ist :-)

Mithilfe der  $\pi[R, s]$  lässt sich eine billigste Ableitung topdown rekonstruieren :-)

Im Beispiel:

$$D_2 = c;$$

$$A_2 = D_2;$$

$$D_1 = M[A_1 + A_2];$$

mit Kosten 5. Die Alternative:

$$D_2 = c;$$

$$D_3 = A_1;$$

$$D_4 = D_3 + D_2;$$

$$A_2 = D_4;$$

$$D_1 = M[A_2];$$

hätte Kosten 7 :-)

## Diskussion:

- Die Code-Erzeugung muss schnell gehn :-)
- Anstelle für jeden Knoten neu zu überprüfen, wie die Regeln zusammen passen, kann die Berechnung auch in einen **endlichen Automaten** kompiliert werden :-))

Ein deterministischer endlicher Baumautomat (DTA)  $A$  besteht aus:

$Q$	$\equiv$	endliche Menge von Zuständen
$\Sigma$	$\equiv$	Operatoren und Konstanten
$\delta_a$	$\equiv$	Übergangsfunktion für $a \in \Sigma$
$F \subseteq Q$	$\equiv$	akzeptierende Zustände

Dabei ist:

$\delta_c : Q$  falls  $c$  Konstante

$\delta_a : Q^k \rightarrow Q$  falls  $a$   $k$ -stellig

Beispiel:

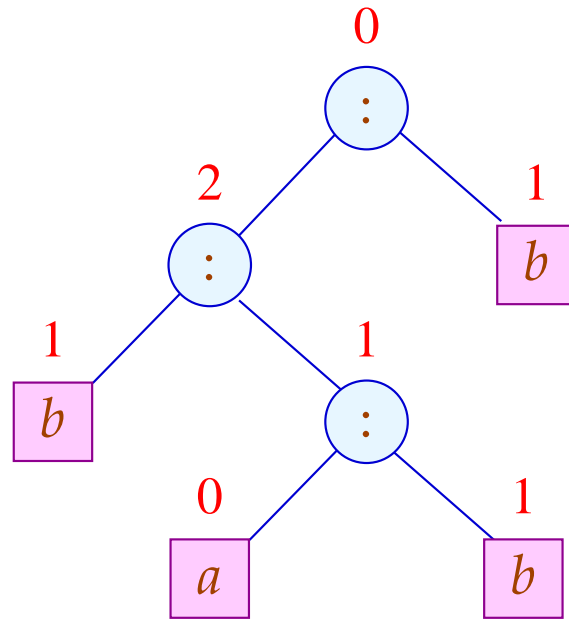
$$Q = \{0, 1, 2\} \quad F = \{0\}$$

$$\Sigma = \{a, b, :\}$$

$$\delta_a = 0 \quad \delta_b = 1$$

$$\delta : (s_1, s_2) = (s_1 + s_2) \% 3$$

// akzeptiert alle Bäume mit  $3 \cdot k$   $b$ -Blättern



Der Zustand an einem Knoten  $a$  ergibt sich aus den Zuständen der Kinder mittels  $\delta_a$  (-:

$$Q(c) = \delta_c$$

$$Q(a(t_1, \dots, t_k)) = \delta_a(Q(t_1), \dots, Q(t_k))$$

Die von  $A$  definierte Sprache (oder: Menge von Bäumen) ist:

$$\mathcal{L}(A) = \{t \mid Q(t) \in F\}$$

... in unserer Anwendung:

$Q$   $\equiv$  Teilmengen von  $\text{Reg} \cup \text{Term} \cup \{S\}$

// I.a. werden nicht **sämtliche** Teilmengen benötigt :-)

$F$   $\equiv$  gewünschter Effekt

$\delta_R$   $\equiv$  Move  $\{R\}$

$\delta_c$   $\equiv$  Move  $\{c\}$

$\delta_a(Q_1, \dots, Q_k)$   $\equiv$  Move  $\{s = a(s_1, \dots, s_k) \in \text{Term} \mid s_i \in Q_i\}$

... im Beispiel:

$$\begin{aligned}\delta_c &= \{A, D\} = q_0 \\ &= \delta_A \\ &= \delta_D\end{aligned}$$

$$\begin{aligned}\delta_+(q_0, q_0) &= \{A, D, A + A\} = q_1 \\ &= \delta_+(q_0, -) \\ &= \delta_+(-, q_0)\end{aligned}$$

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

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  $\implies$   
Die Kosten zwischen Registern differieren nur um eine Konstante :-)
- Komplexe rechte Seiten lassen sich i.a. mittels **elementarerer** Instruktionen simulieren  $\implies$   
Die Kosten zwischen Teilausdrücken und Registern differieren nur um eine Konstante :-))
- Die Kostenberechnung ist additiv  $\implies$   
Wir können statt mit absoluten Kosten-Angaben auch mit Kosten-Differenzen rechnen !!!  
Von diesen gibt es nur **endlich viele** :-)

... im Beispiel:

$$\begin{aligned}\delta_c &= \{A \mapsto 1, D \mapsto 0\} = \bar{q}_0 \\ &= \delta_D\end{aligned}$$

$$\delta_A = \{A \mapsto 0, D \mapsto 1\} = \bar{q}_1$$

$$\delta_+(\bar{q}_1, \bar{q}_0) = \{A \mapsto 2, D \mapsto 1, A + A \mapsto 0\} = \bar{q}_2$$

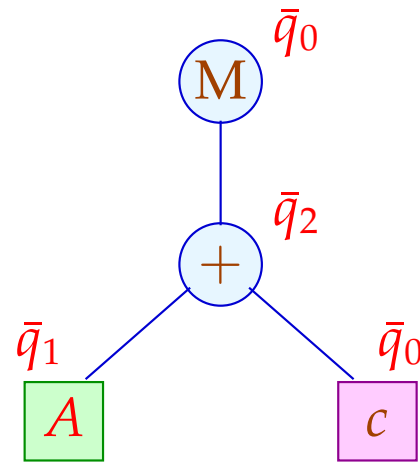
$$\delta_+(\bar{q}_0, \bar{q}_0) = \{A \mapsto 1, D \mapsto 0, A + A \mapsto 1\} = \bar{q}_3$$

$$\delta_+(\bar{q}_1, \bar{q}_1) = \{A \mapsto 4, D \mapsto 3, A + A \mapsto 0\} = \bar{q}_4$$

...

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

... das liefert die folgende Berechnung:



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

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

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

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

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 :-)

⇒ 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

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

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:

```
 $\delta_+(\bar{q}_i, \bar{q}_j) =$  let  $c = \text{class}[\bar{q}_i];$   
                           $d = \text{disp}[c];$   
in if ( $valid[d + j] \equiv c$ )  
    then  $A[d + j]$   
    else default  
end
```



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 :-)