Script generated by TTT

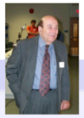
Title: Petter: Compilerbau (04.05.2017)

Date: Thu May 04 14:16:03 CEST 2017

Duration: 100:56 min

Pages: 24

Finite Automata



Michael Rabin

Dana Scott

Definition Finite Automata

A non-deterministic finite automaton (NFA) is a tuple $A = (Q, \Sigma, \delta, I, F)$ with:

Q	a finite set of states;
Σ	a finite alphabet of inputs;
$I \subseteq Q$	the set of start states;
$F \subseteq Q$	the set of final states and
δ	the set of transitions (-relation)

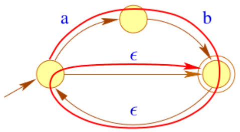
For an NFA, we reckon:

Definition Deterministic Finite Automata

Given $\delta : Q \times \Sigma \to Q$ a function and $|I| = 1$, then we call the NFA A deterministic (DFA).

Finite Automata

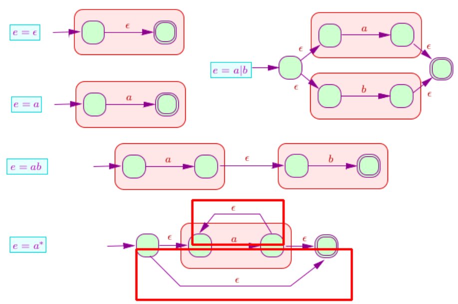
- Computations are paths in the graph.
- Accepting computations lead from I to F.
- An accepted word is the sequence of lables along an accepting computation ...



Chapter 3:

Converting Regular Expressions to NFAs

In Linear Time from Regular Expressions to NFAs



Thompson's Algorithm

Produces $\mathcal{O}(n)$ states for regular expressions of length n.

Berry-Sethi Approach

Berry-Sethi Algorithm

Produces exactly $n + 1$ states without ϵ-transitions and demonstrates → *Equality Systems* and → *Attribute Grammars*

Idea:

The automaton tracks (conceptionally via a marker "•"), in the syntax tree of a regular expression, which subexpressions in e are reachable consuming the rest of input w.

Berry-Sethi Approach

Glushkov Automaton

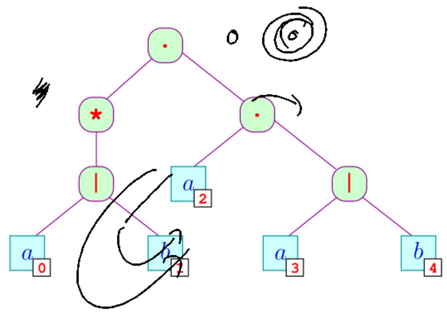
Produces exactly $n + 1$ states without ϵ-transitions and demonstrates → *Equality Systems* and → *Attribute Grammars*

Idea:

The automaton tracks (conceptionally via a marker "•"), in the syntax tree of a regular expression, which subexpressions in e are reachable consuming the rest of input w.

Berry-Sethi Approach

... for example:



Berry-Sethi Approach (naive version)

Construction (naive version):

States: $\bullet r,\ r\bullet$ with r nodes of e;
Start state: $\bullet e$;
Final state: $e\bullet$;
Transitions: for leaves $r \equiv$ $\boxed{i \mid x}$ we require: $(\bullet r, x, r\bullet)$.

The leftover transitions are:

r	Transitions
$r_1 \mid r_2$	$(\bullet r, \epsilon, \bullet r_1)$
	$(\bullet r, \epsilon, \bullet r_2)$
	$(r_1\bullet, \epsilon, r\bullet)$
	$(r_2\bullet, \epsilon, r\bullet)$
$r_1 \cdot r_2$	$(\bullet r, \epsilon, \bullet r_1)$
	$(r_1\bullet, \epsilon, \bullet r_2)$
	$(r_2\bullet, \epsilon, r\bullet)$

r	Transitions
r_1^*	$(\bullet r, \epsilon, r\bullet)$
	$(\bullet r, \epsilon, \bullet r_1)$
	$(r_1\bullet, \epsilon, \bullet r_1)$
	$(r_1\bullet, \epsilon, r\bullet)$
$r_1?$	$(\bullet r, \epsilon, r\bullet)$
	$(\bullet r, \epsilon, \bullet r_1)$
	$(r_1\bullet, \epsilon, r\bullet)$

Berry-Sethi Approach

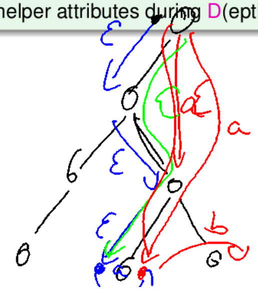
Discussion:

- Most transitions navigate through the expression
- The resulting automaton is in general nondeterministic

$\Rightarrow$ Strategy for the sophisticated version:
Avoid generating ϵ-transitions

Idea:

Pre-compute helper attributes during D(epth)F(irst)S(earch)!



Berry-Sethi Approach

Discussion:

- Most transitions navigate through the expression
- The resulting automaton is in general nondeterministic

$\Rightarrow$ Strategy for the sophisticated version:
Avoid generating ϵ-transitions

Idea:

Pre-compute helper attributes during D(epth)F(irst)S(earch)!

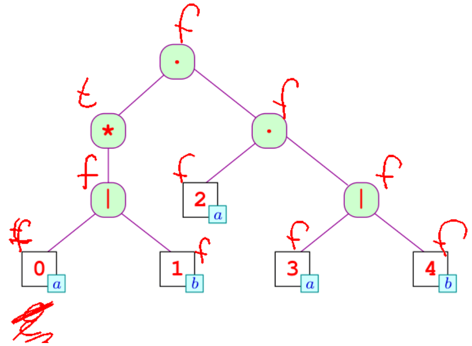
Necessary node-attributes:

- **first** the set of read states below r, which may be reached first, when descending into r.
- **next** the set of read states to the right of r, which may be reached first in the traversal after r.
- **last** the set of read states below r, which may be reached last when descending into r.
- **empty** can the subexpression r consume ϵ ?

Berry-Sethi Approach: 1st step

empty$[r] = t$ if and only if $\boxed{\epsilon} \in [\![r]\!]$

... for example:



Berry-Sethi Approach: 1st step

Implementation:
DFS post-order traversal

for leaves $r \equiv \boxed{i \mid x}$ we find $\text{empty}[r] = (x \equiv \epsilon)$.

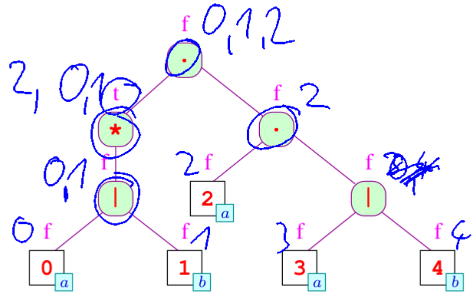
Otherwise:

$$\begin{aligned}
\text{empty}[r_1 \mid r_2] &= \text{empty}[r_1] \vee \text{empty}[r_2] \\
\text{empty}[r_1 \cdot r_2] &= \text{empty}[r_1] \wedge \text{empty}[r_2] \\
\text{empty}[r_1^*] &= t \\
\text{empty}[r_1?] &= t
\end{aligned}$$

Berry-Sethi Approach: 2nd step

The may-set of first reached read states: The set of read states, that may be reached from $\bullet r$ (i.e. while descending into r) via sequences of ϵ-transitions: $\text{first}[r] = \{i \text{ in } r \mid (\bullet r, \epsilon, \bullet\boxed{i \mid x}) \in \delta^*, x \neq \epsilon\}$

... for example:



Berry-Sethi Approach: 2nd step

Implementation:
DFS post-order traversal

for leaves $r \equiv \boxed{i \mid x}$ we find $\text{first}[r] = \{i \mid x \neq \epsilon\}$.

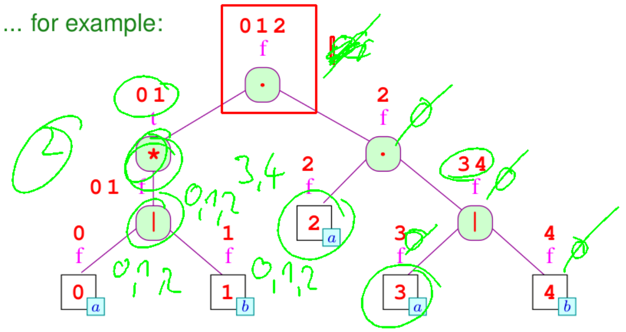
Otherwise:

$$\begin{aligned}
\text{first}[r_1 \mid r_2] &= \text{first}[r_1] \cup \text{first}[r_2] \\
\text{first}[r_1 \cdot r_2] &= \begin{cases} \boxed{\text{first}[r_1] \cup \text{first}[r_2]} & \text{if } \text{empty}[r_1] = t \\ \text{first}[r_1] & \text{if } \text{empty}[r_1] = f \end{cases} \\
\text{first}[r_1^*] &= \text{first}[r_1] \\
\text{first}[r_1?] &= \text{first}[r_1]
\end{aligned}$$

Berry-Sethi Approach: 3rd step

The may-set of next read states: The set of read states within the subtrees right of $\boxed{r\bullet}$, that may be reached next via sequences of ϵ-transitions. $\text{next}[r] = \{i \mid (r\bullet, \epsilon, \bullet\boxed{i \mid x}) \in \delta^*, x \neq \epsilon\}$

... for example:



Berry-Sethi Approach: 3rd step

Implementation:
DFS pre-order traversal

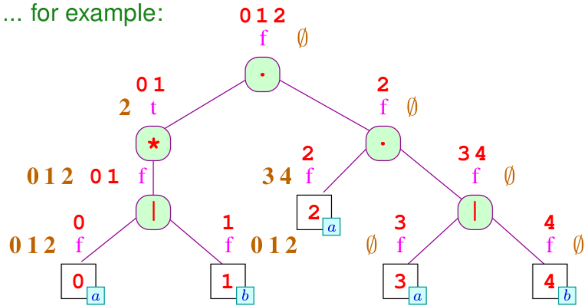
For the root, we find: $\text{next}[e] = \emptyset$
Apart from that we distinguish, based on the context:

r	Equalities
$r_1 \mid r_2$	$\text{next}[r_1] = \text{next}[r]$ $\text{next}[r_2] = \text{next}[r]$
$r_1 \cdot r_2$	$\text{next}[r_1] = \begin{cases} \text{first}[r_2] \cup \text{next}[r] & \text{if } \text{empty}[r_2] = t \\ \text{first}[r_2] & \text{if } \text{empty}[r_2] = f \end{cases}$ $\text{next}[r_2] = \text{next}[r]$
r_1^*	$\text{next}[r_1] = \text{first}[r_1] \cup \text{next}[r]$
$r_1?$	$\text{next}[r_1] = \text{next}[r]$

Berry-Sethi Approach: 3rd step

The may-set of next read states: The set of read states within the subtrees right of $r\bullet$, that may be reached next via sequences of ϵ-transitions. $\text{next}[r] = \{i \mid (r\bullet, \epsilon, \bullet\boxed{i \mid x}) \in \delta^*, x \neq \epsilon\}$

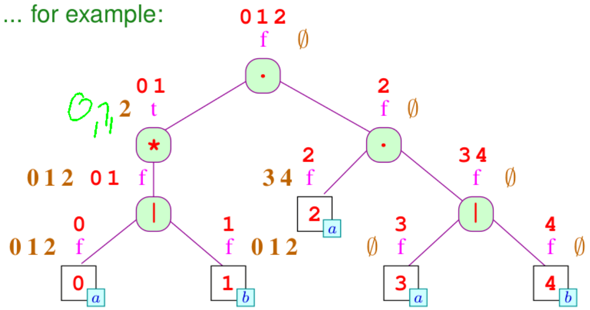
... for example:



Berry-Sethi Approach: 3rd step

The may-set of next read states: The set of read states within the subtrees right of $r\bullet$, that may be reached next via sequences of ϵ-transitions. $\text{next}[r] = \{i \mid (r\bullet, \epsilon, \bullet\boxed{i \mid x}) \in \delta^*, x \neq \epsilon\}$

... for example:



Berry-Sethi Approach: 3rd step

Implementation:
DFS pre-order traversal

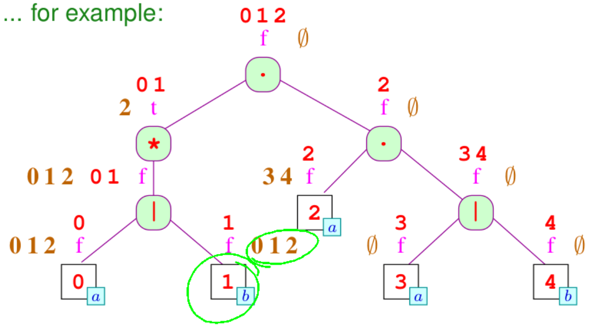
For the root, we find: $\text{next}[e] = \emptyset$
Apart from that we distinguish, based on the context:

r	Equalities
$r_1 \mid r_2$	$\text{next}[r_1] = \text{next}[r]$ $\text{next}[r_2] = \text{next}[r]$
$r_1 \cdot r_2$	$\text{next}[r_1] = \begin{cases} \text{first}[r_2] \cup \text{next}[r] & \text{if } \text{empty}[r_2] = t \\ \text{first}[r_2] & \text{if } \text{empty}[r_2] = f \end{cases}$ $\text{next}[r_2] = \text{next}[r]$
r_1^*	$\text{next}[r_1] = \text{first}[r_1] \cup \text{next}[r]$
$r_1?$	$\text{next}[r_1] = \text{next}[r]$

Berry-Sethi Approach: 3rd step

The may-set of next read states: The set of read states within the subtrees right of $r\bullet$ that may be reached next via sequences of ϵ-transitions.
$\text{next}[r] = \{i \mid (r\bullet, \epsilon, \bullet\boxed{i \mid x}) \in \delta^*, x \neq \epsilon\}$

... for example:



Berry-Sethi Approach: 3rd step

Implementation:
DFS pre-order traversal

For the root, we find: $\text{next}[e] = \emptyset$
Apart from that we distinguish, based on the context:

r	Equalities
$r_1 \mid r_2$	$\text{next}[r_1] = \text{next}[r]$ $\text{next}[r_2] = \text{next}[r]$
$r_1 \cdot r_2$	$\text{next}[r_1] = \begin{cases} \text{first}[r_2] \cup \text{next}[r] & \text{if } \text{empty}[r_2] = t \\ \text{first}[r_2] & \text{if } \text{empty}[r_2] = f \end{cases}$ $\text{next}[r_2] = \text{next}[r]$
r_1^*	$\text{next}[r_1] = \text{first}[r_1] \cup \text{next}[r]$
$r_1?$	$\text{next}[r_1] = \text{next}[r]$

Berry-Sethi Approach: 4th step

Implementation:
DFS post-order traversal

for leaves $r \equiv \boxed{i \mid x}$ we find $\text{last}[r] = \{i \mid x \neq \epsilon\}$.

Otherwise:

$$\begin{array}{lll} \text{last}[r_1 \mid r_2] & = & \text{last}[r_1] \cup \text{last}[r_2] \\ \text{last}[r_1 \cdot r_2] & = & \begin{cases} \text{last}[r_1] \cup \text{last}[r_2] & \text{if } \text{empty}[r_2] = t \\ \text{last}[r_2] & \text{if } \text{empty}[r_2] = f \end{cases} \\ \text{last}[r_1^*] & = & \text{last}[r_1] \\ \text{last}[r_1?] & = & \text{last}[r_1] \end{array}$$

Berry-Sethi Approach: (sophisticated version)

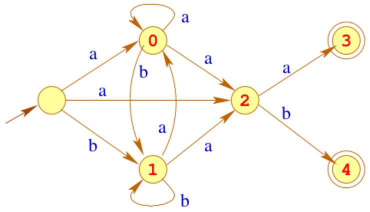
Construction (sophisticated version):
Create an automanton based on the syntax tree's new attributes:

States: $\{\bullet e\} \cup \{i\bullet \mid i \text{ a leaf}\}$
Start state: $\bullet e$
Final states: $\text{last}[e]$ if $\text{empty}[e] = f$
$\{\bullet e\} \cup \text{last}[e]$ otherwise
Transitions: $(\bullet e, a, i\bullet)$ if $i \in \text{first}[e]$ and i labled with a.
$(i\bullet, a, i'\bullet)$ if $i' \in \text{next}[i]$ and i' labled with a.

We call the resulting automaton A_e.

Berry-Sethi Approach

... for example:



Remarks:

- This construction is known as Berry-Sethi- or Glushkov-construction.
- It is used for XML to define Content Models
- The result may not be, what we had in mind...