Script **generated by TTT**

Title: Petter: Compilerbau (19.04.2018)

Date: Thu Apr 19 14:14:31 CEST 2018

Duration: 99:42 min

Pages: 27

Chapter 2:
Basics: Finite Automata

Finite Automata

Michael Rabin

Dana Scott

Definition Finite Automata

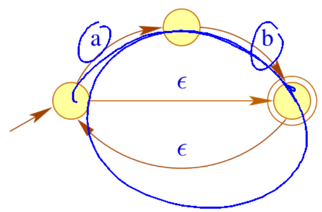
A non-deterministic finite automaton (NFA) is a tuple $A = (Q, \Sigma, \delta, I, F)$ with:

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)

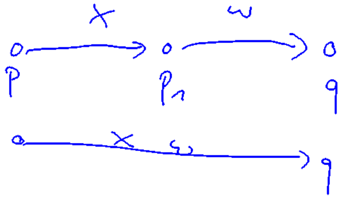
Finite Automata

- 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 ...

abab



Finite Automata



Once again, more formally:

- We define the transitive closure δ^* of $\boxed{\delta}$ as the smallest set $\boxed{\delta'}$ with:

  $\boxed{(p,\boxed{\epsilon},p)\in\delta'}$ and
  $\boxed{(p,xw,q)\in\delta'}$ if $\boxed{(p,x,p_1)\in\delta}$ and $\boxed{(p_1,w,q)}\in\delta'$.

  δ^* characterizes for a path between the states p and q the words obtained by concatenating the labels along it.

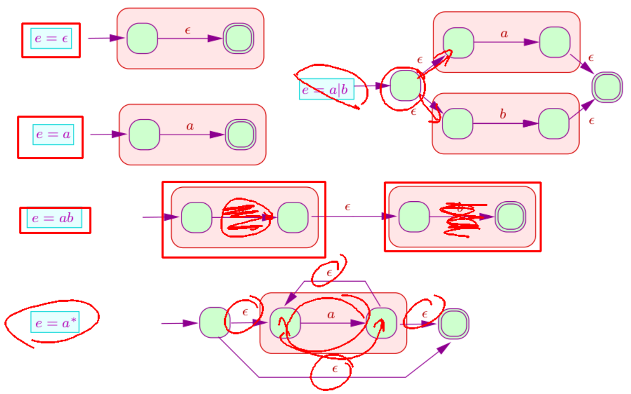
- The set of all accepting words, i.e. A's accepted language can be described compactly as:

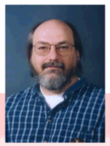
$$\boxed{\mathcal{L}(\boxed{A})} = \boxed{\{w\in\Sigma^*} \mid \boxed{\exists i\in I}, \boxed{f\in F} : \boxed{(\boxed{i},\boxed{w},\boxed{f})\in\delta^*}\}$$

Chapter 3:

Converting Regular Expressions to NFAs

In Linear Time from Regular Expressions to NFAs



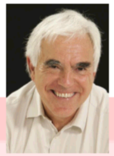


Ken Thompson

Thompson's Algorithm

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

Berry-Sethi Approach



Gerard Berry

Ravi Sethi

Berry-Sethi Algorithm

Produces exactly $n+1$ states without ϵ-transitions and demonstrates $\rightarrow$ *Equality Systems* and $\rightarrow$ *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


Viktor M. Glushkov

Glushkov Automaton

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

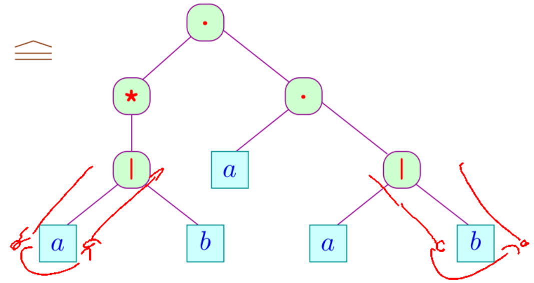
Idea:

The automaton tracks (conceptionally via a marker "$\bullet$"), 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:

$(a|b)^*a(a|b)$ $\widehat{=}$



Berry-Sethi Approach

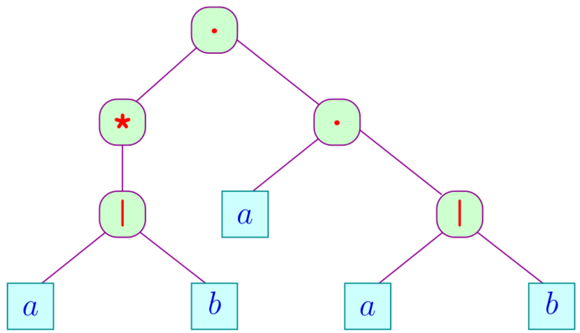
In general:

- Input is only consumed at the leaves.
- Navigating the tree does not consume input $\rightarrow$ ϵ-transitions
- For a formal construction we need identifiers for states.
- For a node n's identifier we take the subexpression, corresponding to the subtree dominated by n.
- There are possibly identical subexpressions in one regular expression.

$\Longrightarrow$ we enumerate the leaves ...

Berry-Sethi Approach

... for example:



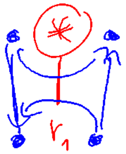
Berry-Sethi Approach (naive version)

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

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, which may be reached first in the traversal after r.

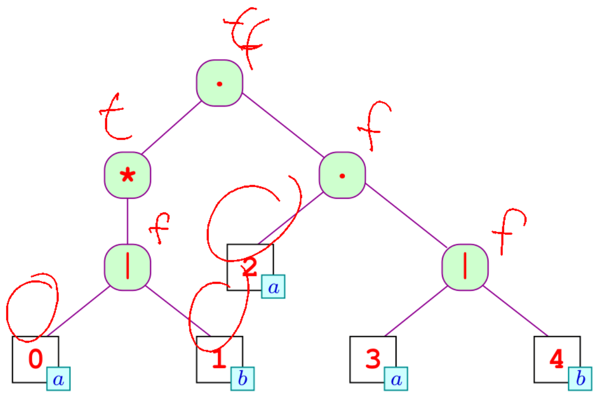
last — the set of read states below r, which may be reached last when descending into r.

empty — can the subexpression r consume ϵ ?

Berry-Sethi Approach: 1st step

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

... for example:



Berry-Sethi Approach: 1st step

$(\epsilon \mid r)$

Implementation:
DFS post-order traversal

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

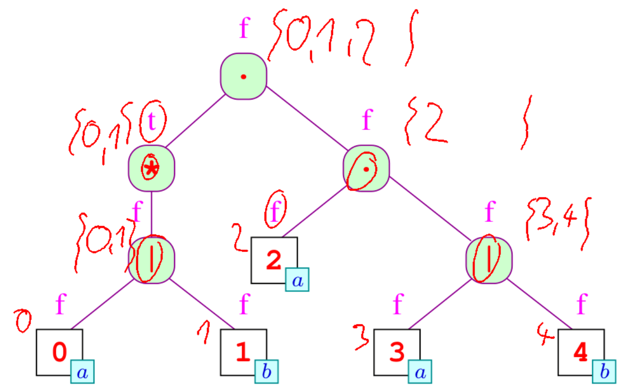
Otherwise:

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

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 $\text{first}[r] = \{i \mid x \neq \epsilon\}$.

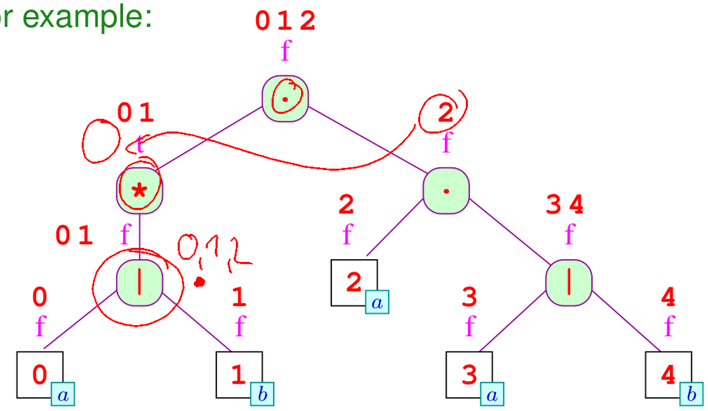
Otherwise:

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

Berry-Sethi Approach: 3rd step

The may-set of next read states: The set of read states reached after reading r, 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

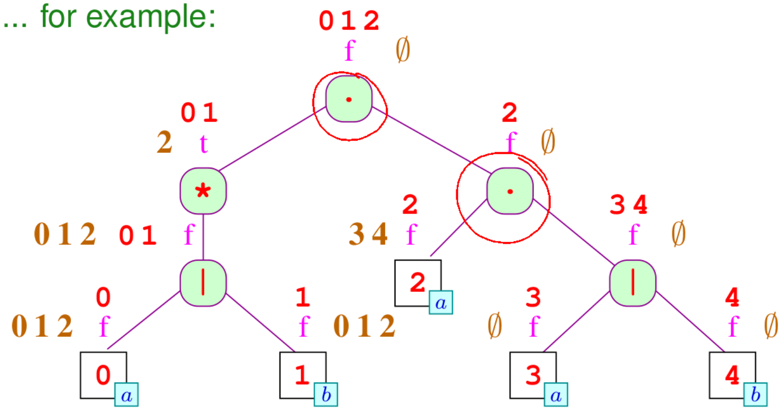
For the root, we find: $\text{next}[e] = \emptyset$
Apart from that we distinguish, based on the context:

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 reached after reading r, 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:

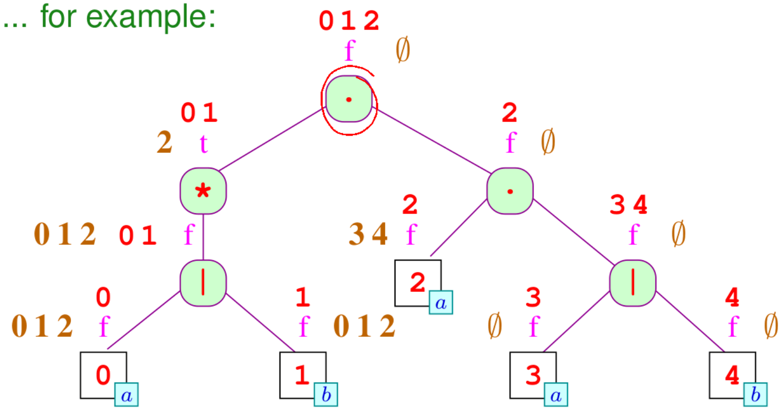
r	Equalities
$r_1 \mid r_2$	$\text{next}[r_1] = \text{next}[r]$ $\text{next}[r_2] = \text{next}[r]$
$r_1 \cdot r_2$	$\text{next}[r_1] = \begin{cases} \text{first}[r_2] \cup \text{next}[r] & \text{if } \text{empty}[r_2] = t \\ \text{first}[r_2] & \text{if } \text{empty}[r_2] = f \end{cases}$ $\text{next}[r_2]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

38 / 288

Berry-Sethi Approach: 3rd step

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

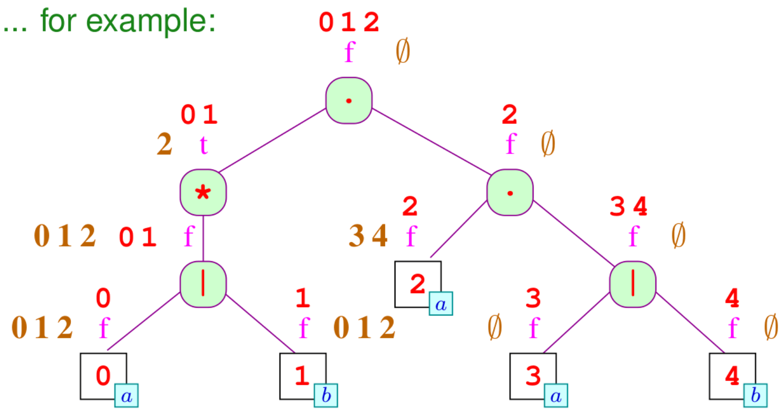
... for example:



Berry-Sethi Approach: 4th step

The may-set of last reached read states: The set of read states, which may be reached last during the traversal of r connected to the root via ϵ-transitions only: $\text{last}[r] = \{i \text{ in } r \mid (\boxed{i \mid x}\bullet, \epsilon, r\bullet) \in \delta^*, x \neq \epsilon\}$

... for example:



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:

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

Berry-Sethi Approach: (sophisticated version)

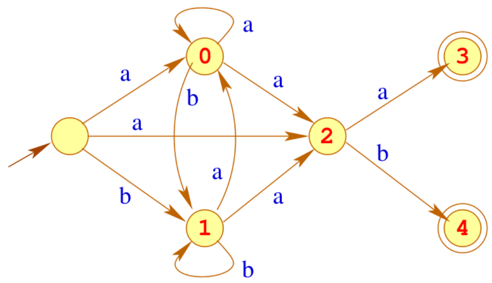
Construction (sophisticated version):
Create an automanton based on the syntax tree's new attributes:

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...