Script generated by TTT

Title: Simon: Compilerbau (14.04.2014)

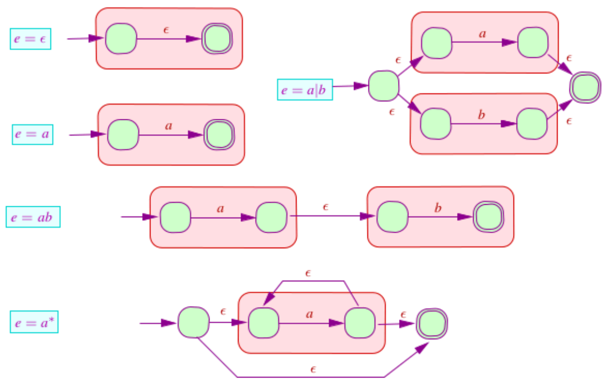
Date: Mon Apr 14 14:15:06 CEST 2014

Duration: 94:06 min

Pages: 59

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 → *Equality Systems* and → *Attribute Grammars*

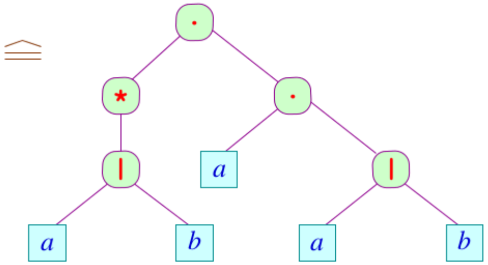
Idea:

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:

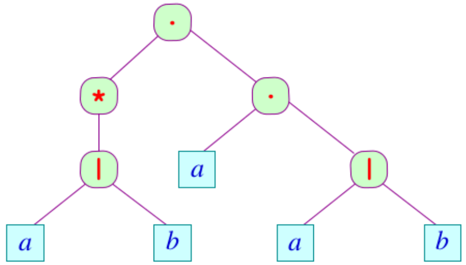
$(a|b)^* a (a|b)$ $\widehat{=}$



Berry-Sethi Approach

... for example:

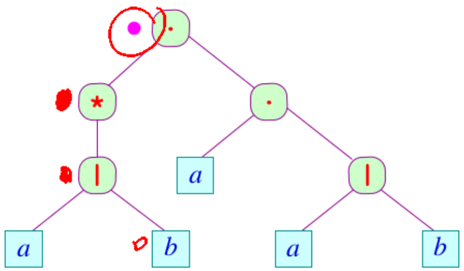
$w = bbaa$:



Berry-Sethi Approach

... for example:

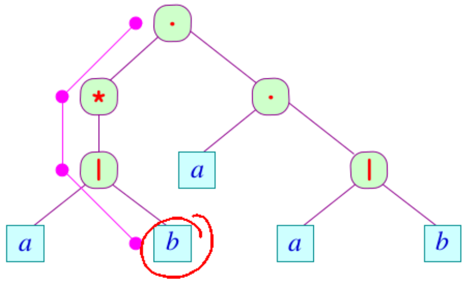
$w = bbaa$:



Berry-Sethi Approach

... for example:

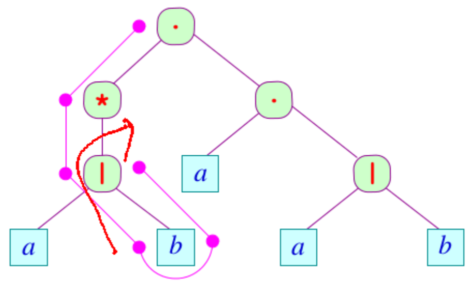
$w = bbaa$:



Berry-Sethi Approach

... for example:

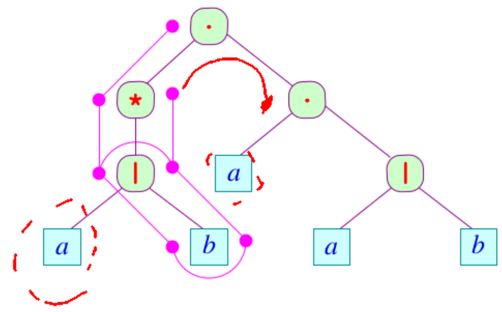
$w \;=\; baa \;:$



Berry-Sethi Approach

... for example:

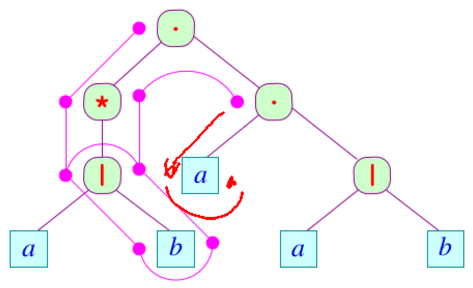
$w \;=\; aa \;:$



Berry-Sethi Approach

... for example:

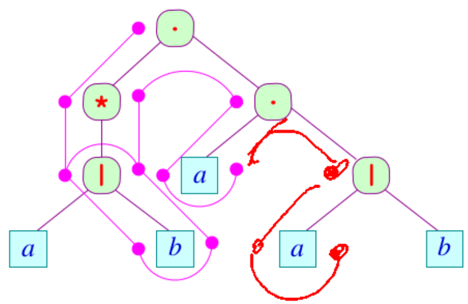
$w \;=\; aa \;:$



Berry-Sethi Approach

... for example:

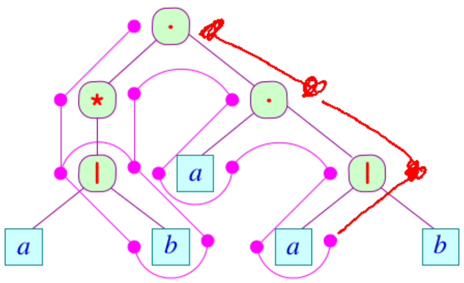
$w \;=\; a \;:$



Berry-Sethi Approach

... for example:

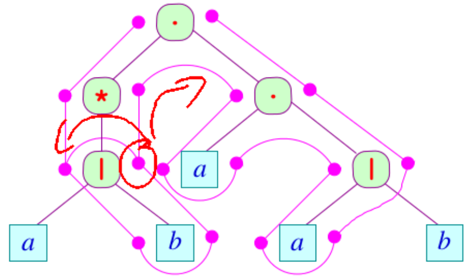
$w \quad = \qquad :$



Berry-Sethi Approach

... for example:

$w \quad = \qquad :$



Berry-Sethi Approach

In general:

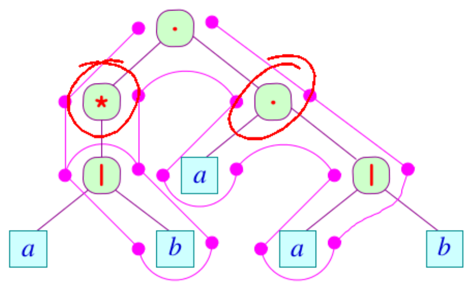
- Input is only consumed by the leaves.
- Navigation in the tree is done without consuming input, i.e. via ϵ-transition.
- For a formal construction we need identifiers for states.
- Therefore we use the subexpression, corresponding to the subtree, dominated by the particular node.
- There are possibly identical subexpressions in one regular expression.

$\Longrightarrow$ we enumerate the leaves ...

Berry-Sethi Approach

... for example:

$w \quad = \qquad :$



Berry-Sethi Approach

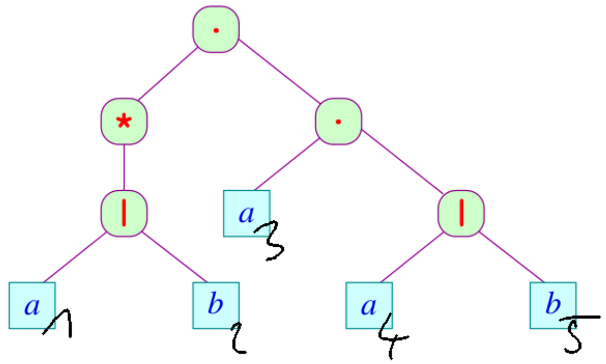
In general:

- Input is only consumed by the leaves.
- Navigation in the tree is done without consuming input, i.e. via ϵ-transition.
- For a formal construction we need identifiers for states.
- Therefore we use the subexpression, corresponding to the subtree, dominated by the particular node.
- There are possibly identical subexpressions in one regular expression.

$\Longrightarrow$ we enumerate the leaves ...

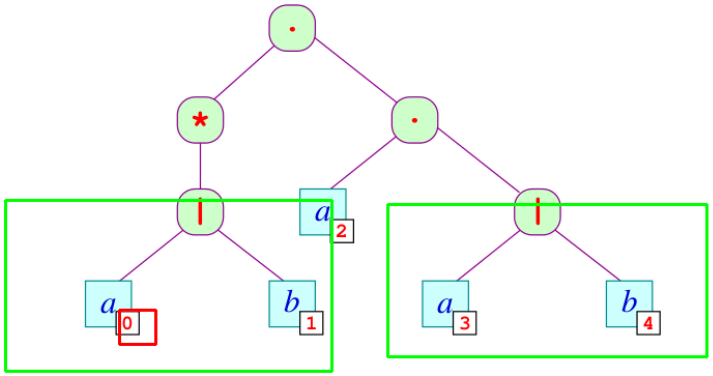
Berry-Sethi Approach

... for example:



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 $r \equiv$ i x we require: $(\bullet r, x, r\bullet)$.

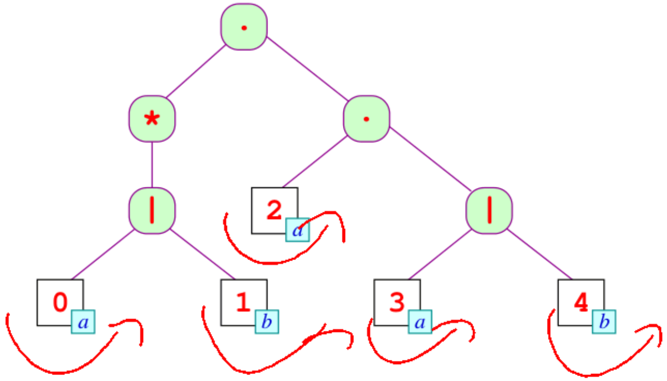
The leftover transitions are:

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

... 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 $r \equiv$ [i | 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

⇒ Strategy for the sophisticated version:
Avoid generating ϵ-transitions

Berry-Sethi Approach

Discussion:

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

⇒ Strategy for the sophisticated version:
Avoid generating ϵ-transitions

Necessary node-attributes:

empty can the subexpression r consume ϵ ?

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

next the set of read states on the right of r, which may 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.

Berry-Sethi Approach

Discussion:

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

⇒ Strategy for the sophisticated version:
Avoid generating ϵ-transitions

Necessary node-attributes:

empty can the subexpression r consume ϵ ?

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

next the set of read states on the right of r, which may 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.

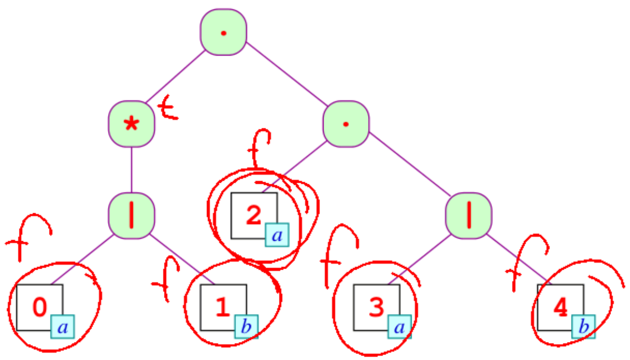
Idea:

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

Berry-Sethi Approach: 1st step

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

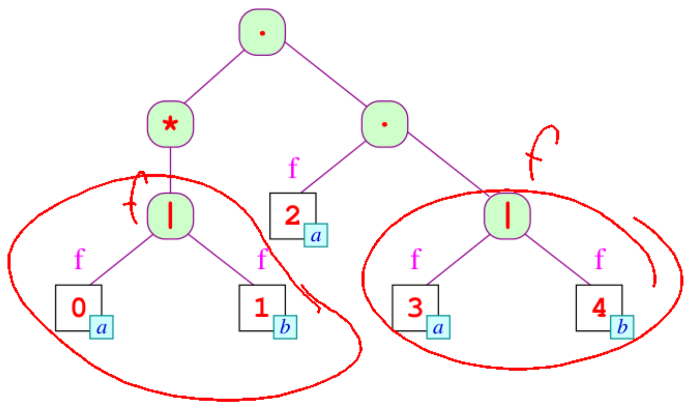
... for example:



Berry-Sethi Approach: 1st step

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

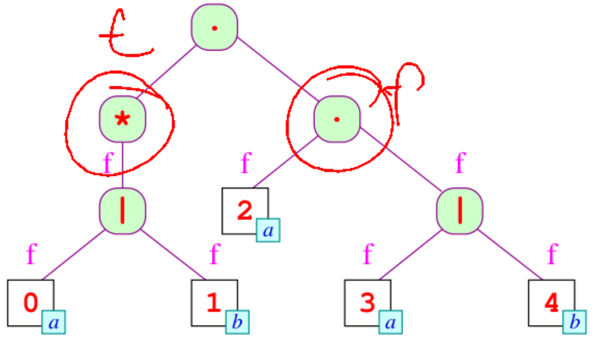
... for example:



Berry-Sethi Approach: 1st step

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

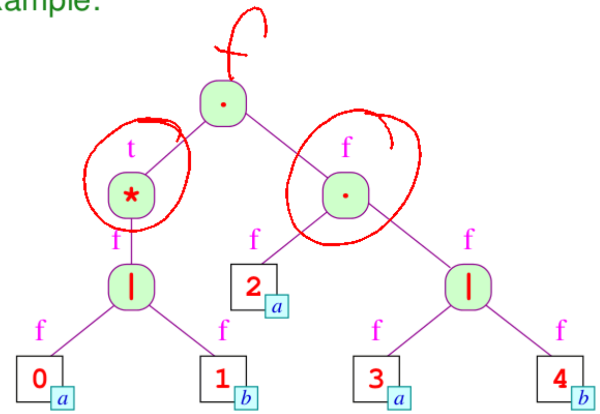
... for example:



Berry-Sethi Approach: 1st step

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

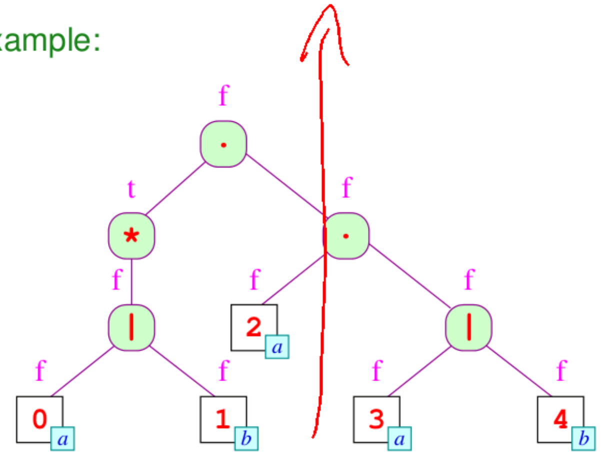
... for example:



Berry-Sethi Approach: 1st step

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

... for example:



Berry-Sethi Approach: 2nd step

Implementation:

DFS post-order traversal

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

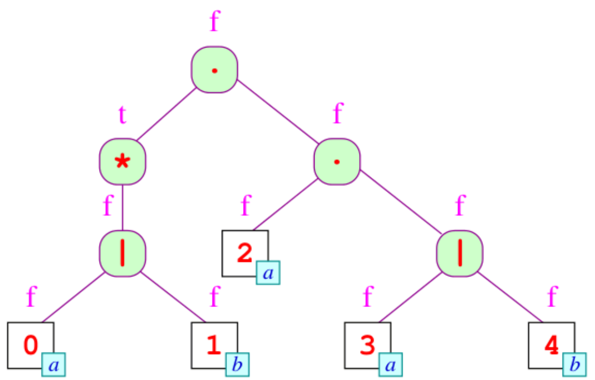
Otherwise:

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

Berry-Sethi Approach: 1st step

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

... for example:



Berry-Sethi Approach: 2nd step

Implementation:
DFS post-order traversal

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

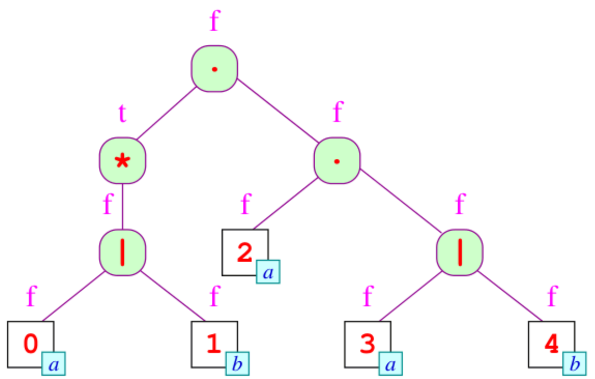
Otherwise:

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

Berry-Sethi Approach: 2nd step

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

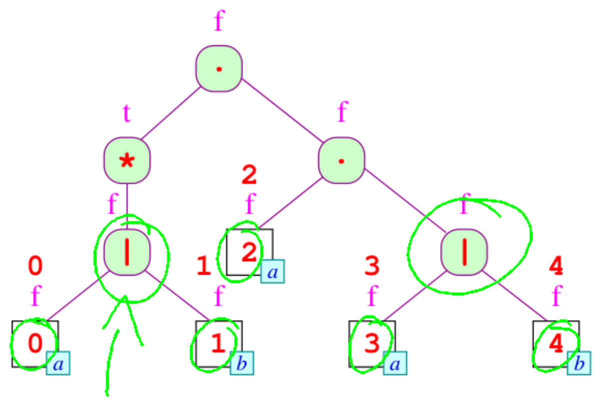
... for example:



Berry-Sethi Approach: 2nd step

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

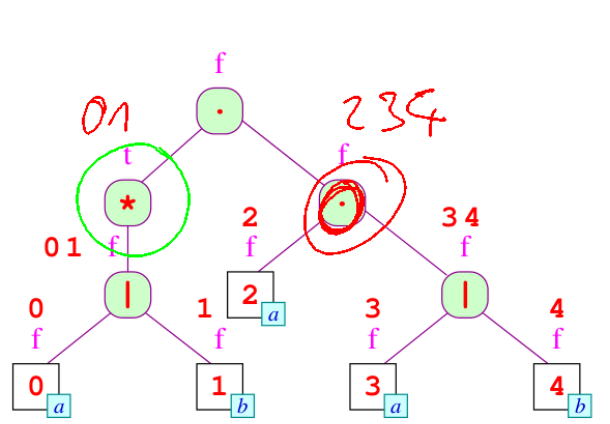
... for example:



Berry-Sethi Approach: 2nd step

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

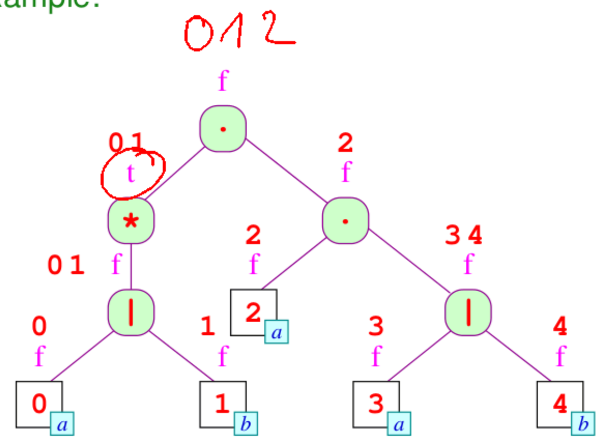
... for example:



Berry-Sethi Approach: 2nd step

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

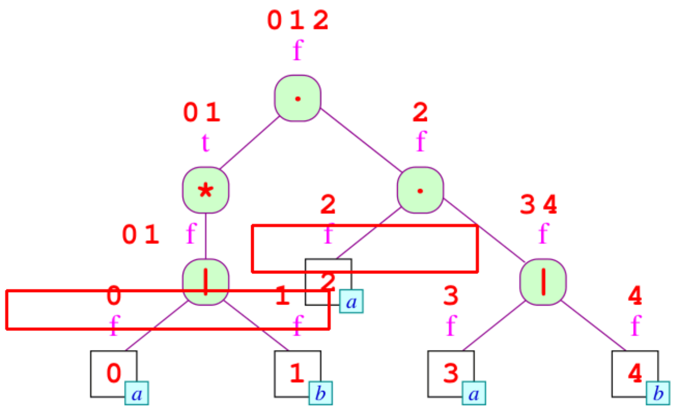
... for example:



Berry-Sethi Approach: 2nd step

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

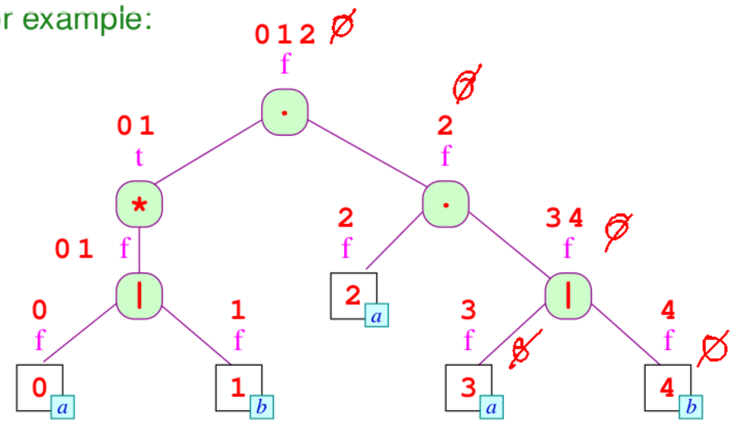
... for example:



Berry-Sethi Approach: 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. next$[r] = \{i \mid (r\bullet, \epsilon, \bullet \boxed{i \mid x}) \in \delta^*, x \neq \epsilon\}$

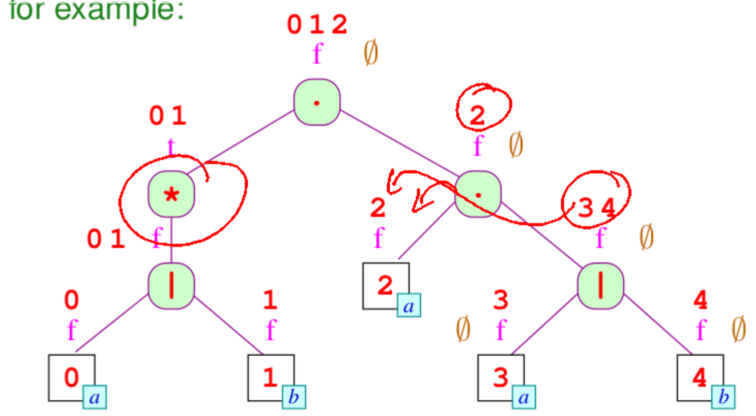
... for example:



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. $\text{next}[r] = \{i \mid (r\bullet, \epsilon, \bullet\boxed{i \mid x}) \in \delta^*, x \neq \epsilon\}$

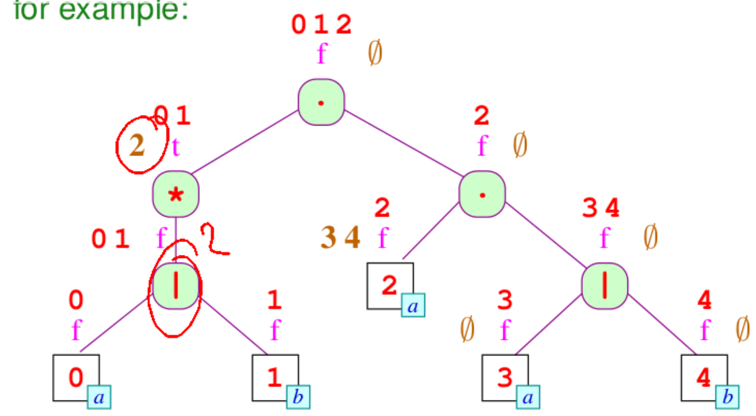
... for example:



Berry-Sethi Approach: 3rd step

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. $\text{next}[r] = \{i \mid (r\bullet, \epsilon, \bullet\boxed{i \mid x}) \in \delta^*, x \neq \epsilon\}$

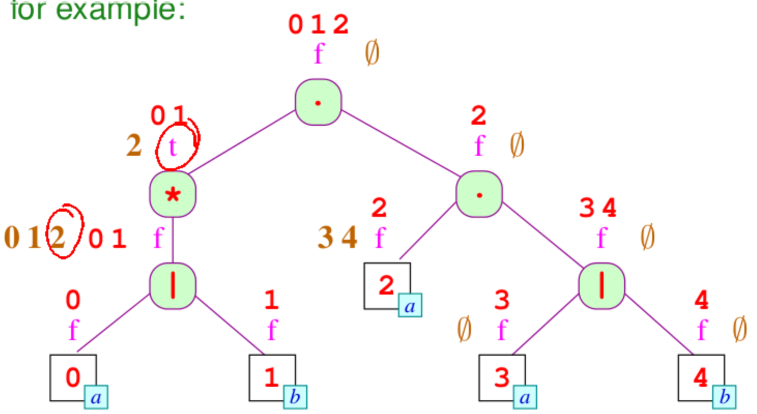
... for example:



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. $\text{next}[r] = \{i \mid (r\bullet, \epsilon, \bullet\boxed{i \mid x}) \in \delta^*, x \neq \epsilon\}$

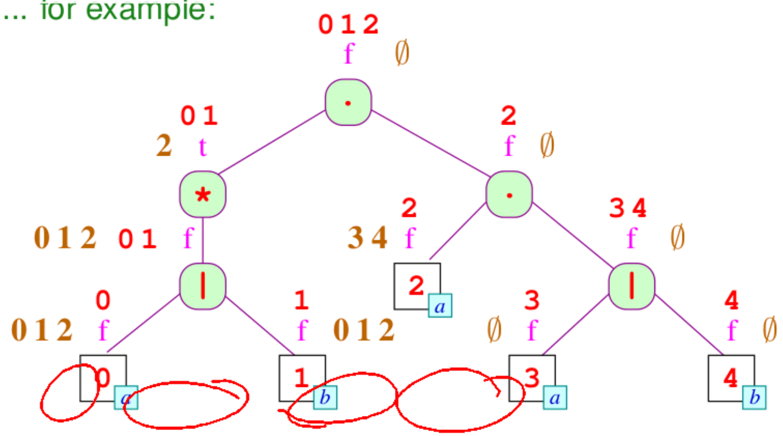
... for example:



Berry-Sethi Approach: 3rd step

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. $\text{next}[r] = \{i \mid (r\bullet, \epsilon, \bullet\boxed{i \mid x}) \in \delta^*, x \neq \epsilon\}$

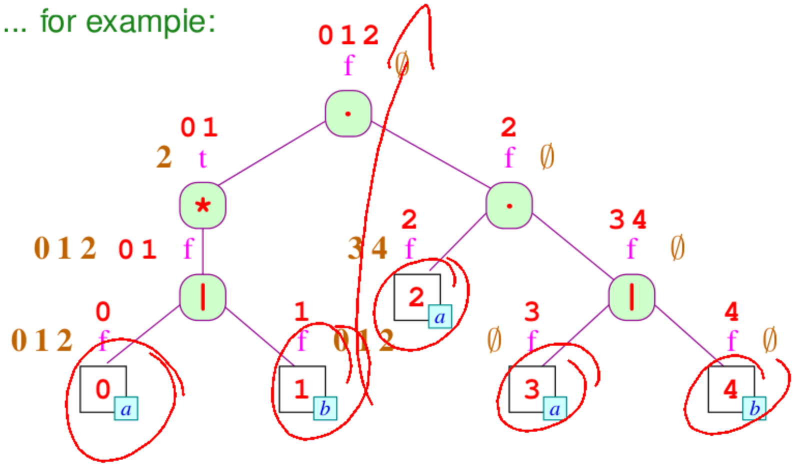
... for example:



Berry-Sethi Approach: 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: $\text{last}[r] = \{i \text{ in } r \mid (\boxed{i \mid x}\bullet, \epsilon, r\bullet) \in \delta^*, x \neq \epsilon\}$

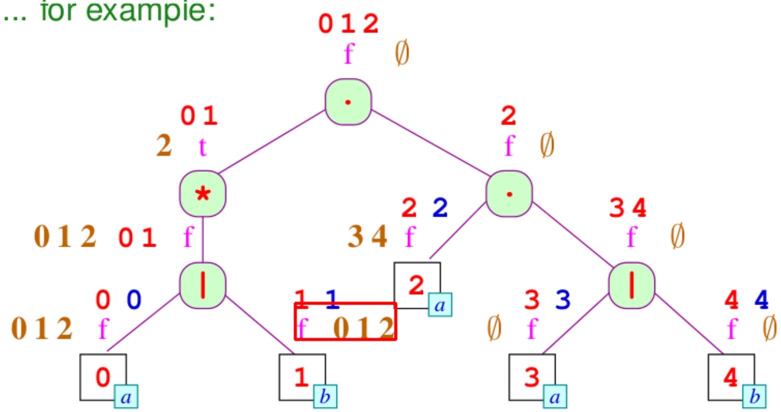
... for example:



Berry-Sethi Approach: 4th step

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: (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: $\bullet e$

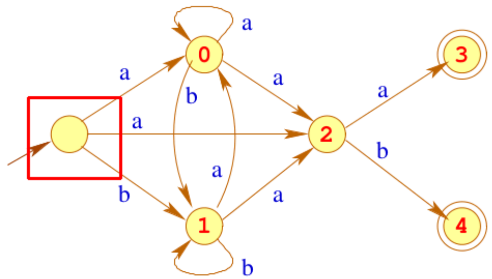
Final states: $\text{last}[e]$ if $\text{empty}[e] = f$
$\{\bullet e\} \cup \text{last}[e]$ otherwise

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:



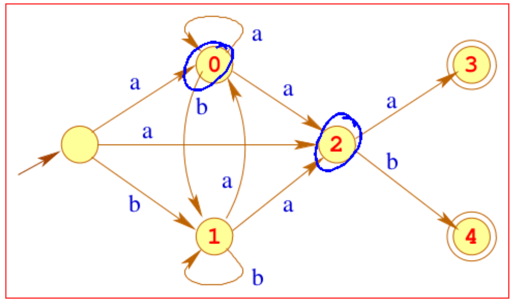
Remarks:

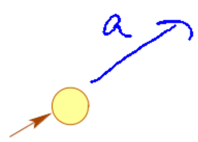
- This construction is known as Berry-Sethi- or Glushkov-construction.
- It is used for XML to define Content Models
- The result may not be, what we had in mind...

Chapter 4:
Turning NFAs deterministic

Powerset Construction

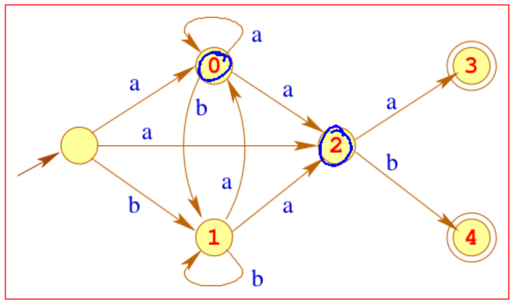
... for example:

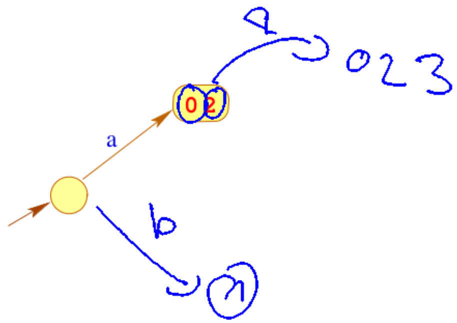




Powerset Construction

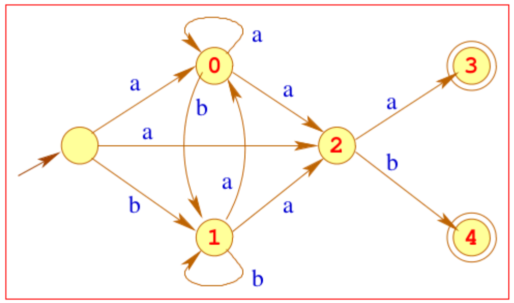
... for example:

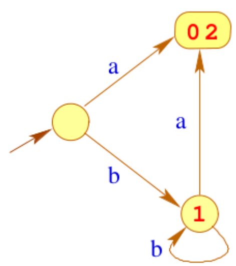




Powerset Construction

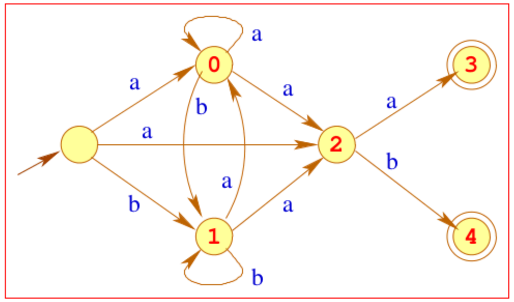
... for example:

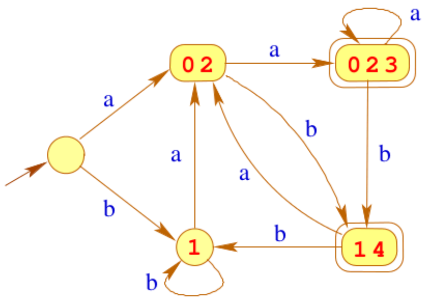




Powerset Construction

... for example:





Powerset Construction

Theorem:
For every non-deterministic automaton $A = (Q, \Sigma, \delta, I, F)$ we can compute a deterministic automaton $\mathcal{P}(A)$ with

$$\mathcal{L}(A) = \mathcal{L}(\mathcal{P}(A))$$

Construction:

States: Powersets of Q;
Start state: I;
Final states: $\{Q' \subseteq Q \mid Q' \cap F \neq \emptyset\}$;
Transitions: $\delta_\mathcal{P}(Q', a) = \{q \in Q \mid \exists p \in Q' : (p, a, q) \in \delta\}$

Powerset Construction

Bummer!
There are exponentially many powersets of Q

- Idea: Consider only contributing powersets. Starting with the set $Q_\mathcal{P} = \{I\}$ we only add further states by need ...
- i.e., whenever we can reach them from a state in $Q_\mathcal{P}$
- Even though, the resulting automaton can become enormously huge
  ... which is (sort of) not happening in practice

Powerset Construction

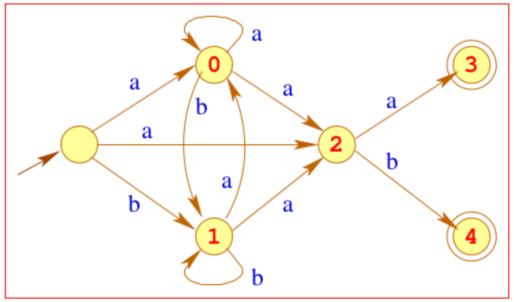
Bummer!
There are exponentially many powersets of Q

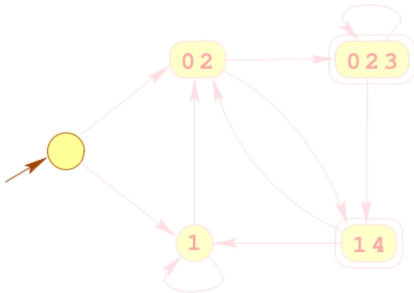
- Idea: Consider only contributing powersets. Starting with the set $Q_\mathcal{P} = \{I\}$ we only add further states by need ...
- i.e., whenever we can reach them from a state in $Q_\mathcal{P}$
- Even though, the resulting automaton can become enormously huge
  ... which is (sort of) not happening in practice
- Therefore, in tools like `grep` a regular expression's DFA is never created!
- Instead, only the sets, directly necessary for interpreting the input are generated while processing the input

Powerset Construction

... for example:

a	b	a	b

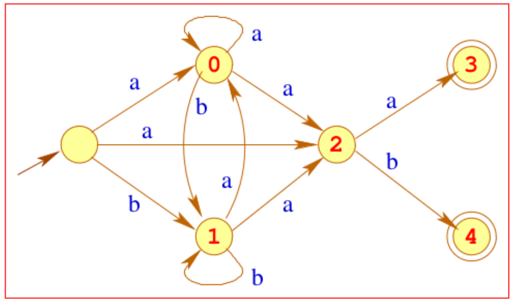


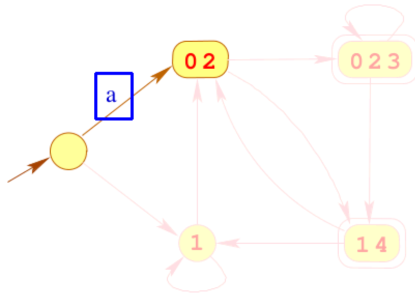


Powerset Construction

... for example:

a	b	a	b

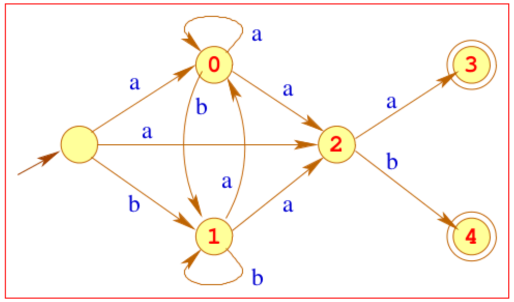


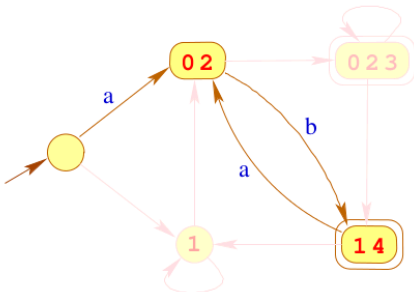


Powerset Construction

... for example:

a	b	a	b





Remarks:

- For an input sequence of length n, maximally $\mathcal{O}(n)$ sets are generated
- Once a set/edge of the DFA is generated, they are stored within a hash-table.
- Before generating a new transition, we check this table for already existing edges with the desired label.