In Linear Time from Regular Expressions to NFAs

Thompson’s Algorithm
Produces $O(n)$ states for regular expressions of length $n$.

Berry-Sethi Approach

Glushkov Algorithm
Produces exactly $n + 1$ states without $\epsilon$-transitions and demonstrates $\rightarrow$ Equality Systems and $\rightarrow$ Attribute Grammars

Idea:
The automaton tracks (conceptionally via a marker $\epsilon$), in the syntax tree of a regular expression, which subexpressions in $\epsilon$ are reachable consuming the rest of input $w$. 

... for example:

$$(a|b)^*a(a|b)$$
Berry-Sethi Approach

... for example:

\[ w = \emptyset \text{aa} : \]

\[ w = \text{baa} : \]

Berry-Sethi Approach

... for example:

\[ w = \emptyset \text{aa} : \]

\[ w = \text{baa} : \]

Berry-Sethi Approach

... for example:

\[ w = \emptyset \text{aa} : \]

\[ w = \emptyset \text{aa} : \]

Berry-Sethi Approach

... for example:

\[ w = \emptyset \text{aa} : \]

\[ w = \emptyset \text{aa} : \]
Berry-Sethi Approach

... for example:

\[ \eta = : \]

Berry-Sethi Approach

In general:

- Input is only consumed at the leaves.
- Navigating the tree does not consume input \( \epsilon \)-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.

\[ \Longrightarrow \text{ we enumerate the leaves} \ldots \]

Berry-Sethi Approach

... for example:

\[ \eta = : \]

Berry-Sethi Approach

... for example:
Berry-Sethi Approach (naive version)

Construction (naive version):

States: $r$, $r\epsilon$ with $r$ nodes of $\epsilon$;
Start state: $r\epsilon$;
Final state: $r\epsilon$;
Transitions: for leaves $r \equiv i \rightarrow x$ we require: $(r\epsilon, x, r\epsilon)$.
The leftover transitions are:

<table>
<thead>
<tr>
<th>$r$</th>
<th>Transitions</th>
<th>$r$</th>
<th>Transitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1 \mid r_2$</td>
<td>$(r\epsilon, c, r_1\epsilon)$ $r\epsilon, c, r_2\epsilon$ $(r_1\epsilon, c, r_\epsilon)$ $(r_2\epsilon, c, r_\epsilon)$</td>
<td>$r_1^\prime$</td>
<td>$(r\epsilon, c, r_\epsilon)$ $(r\epsilon, c, r_1\epsilon)$ $(r_1\epsilon, c, r_\epsilon)$ $(r_\epsilon, c, r_\epsilon)$ $(r_\epsilon, c, r_\epsilon)$</td>
</tr>
</tbody>
</table>

Berry-Sethi Approach

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

⇒ Strategy for the sophisticated version:
Avoid generating $\epsilon$-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 $\epsilon$-transitions

Idea:
Pre-compute helper attributes doing $D(\text{depth})F(\text{first})S(\text{search})$!
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

Idea:
Pre-compute helper attributes during D(eth)F(irst)S(earch)!

Necessary node-attributes:
- \textbf{first} the set of read states below \( r \), which may be reached \textit{first}, when descending into \( r \).
- \textbf{next} the set of read states to the \textit{right of} \( r \), which may be reached \textit{first} in the traversal after \( r \).
- \textbf{last} the set of read states below \( r \), which may be reached \textit{last} when descending into \( r \).
- \textbf{empty} can the subexpression \( r \) consume \( ε \) ?

Berry-Sethi Approach: 1st step

Implementation:
DFS post-order traversal

for leaves \( r \equiv \square \) we find \( \textbf{empty}[r] = \epsilon \).

Otherwise:
- \( \textbf{empty}[r_1 \cdot r_2] = \textbf{empty}[r_1] \land \textbf{empty}[r_2] \)
- \( \textbf{empty}[r_1 \cdot r_2] = \textbf{empty}[r_1] \lor \textbf{empty}[r_2] \)
- \( \textbf{empty}[r_1] = t \)
- \( \textbf{empty}[r] = t \)

Berry-Sethi Approach: 1st step

... for example:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>a</td>
<td>a</td>
<td>b</td>
</tr>
</tbody>
</table>

Berry-Sethi Approach: 2nd step

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

... for example:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>a</td>
<td>a</td>
<td>b</td>
</tr>
</tbody>
</table>

Diagram:

- \( ε \)
- \( f \)
- \( a \)
- \( b \)
- \( 0 \)
- \( 1 \)
- \( 2 \)
- \( 3 \)
- \( 4 \)
Berry-Sethi Approach: 2nd step

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

... for example:

![Diagram 1]

Berry-Sethi Approach: 2nd step

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

... for example:

![Diagram 2]
Berry-Sethi Approach: 2nd step

**Implementation:**

DFS *post-order* traversal

For leaves: \( r = x \) we find \( \text{first}[r] = \{ i \mid x \neq \epsilon \} \).

Otherwise:

\[
\begin{align*}
\text{first}[r_1 \cdot r_2] &= \text{first}[r_1] \cup \text{first}[r_2] \\
\text{first}[r_1 \cdot r_2] &= \begin{cases} 
\text{first}[r_1] \cup \text{first}[r_2] & \text{if empty}(r_1) = \text{false} \\
\text{first}[r_1] & \text{if empty}(r_1) = \text{true}
\end{cases}
\end{align*}
\]

Berry-Sethi Approach: 3rd step

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

... for example:

```
012
```

Berry-Sethi Approach: 3rd step

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

... for example:

```
012
```
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:

<table>
<thead>
<tr>
<th>( r )</th>
<th>Equalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_1 \rightarrow r_2 )</td>
<td>( \text{next} [r_1] = \text{next} [r] )</td>
</tr>
<tr>
<td>( r_1 \rightarrow r_2 )</td>
<td>( \text{next} [r_2] = \text{next} [r] )</td>
</tr>
<tr>
<td>( r_1 \cdot r_2 )</td>
<td>( \text{next} [r_1] = \text{first} [r_2] \cup \text{next} [r] ) if empty ( r_2 ) = ( i )</td>
</tr>
<tr>
<td>( r_1 \cdot r_2 )</td>
<td>( \text{next} [r_2] = \text{first} [r_1] ) if empty ( r_2 ) = ( f )</td>
</tr>
<tr>
<td>( r^* )</td>
<td>( \text{next} [r_1] = \text{next} [r] )</td>
</tr>
<tr>
<td>( r_1 )</td>
<td>( \text{next} [r_1] = \text{next} [r] )</td>
</tr>
</tbody>
</table>

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 \( \epsilon \)-transitions only:

\( \text{last} [r] = \{ i \in r \mid (i \rightarrow x \bullet, \epsilon, r) \in \delta^*, x \neq \epsilon \} \)

... for example:

![Diagram](image)

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 \( \epsilon \)-transitions only:

\( \text{last} [r] = \{ i \in r \mid (i \rightarrow x \bullet, \epsilon, r) \in \delta^*, x \neq \epsilon \} \)

... for example:

![Diagram](image)
Berry-Sethi Approach: 4th step

Implementation:

\textbf{DFS post-order traversal}

for leaves \( r = \{ i | x \} \) we find \( \text{last}[r] = \{ i | x \neq \epsilon \} \).

Otherwise:

\[
\begin{align*}
\text{last}[r_1 \cdot r_2] &= \text{last}[r_1] \cup \text{last}[r_2] \\
\text{last}[r_1 \cup r_2] &= \text{last}[r_1] \cup \text{last}[r_2] \quad \text{if empty}[r_2] = t \\
\text{last}[r_1] &= \text{last}[r_1] \quad \text{if empty}[r_2] = f
\end{align*}
\]

Berry-Sethi Approach

... for example:

\begin{center}
\begin{tikzpicture}
  \node (1) at (0,0) [shape=circle] {1};
  \node (2) at (1,1) [shape=circle] {2};
  \node (3) at (2,2) [shape=circle] {3};
  \node (4) at (1,-1) [shape=circle] {4};
  \node (0) at (1,0) [shape=circle] {0};

  \path[->, thick]
    (1) edge node {a} (2)
    (2) edge node {a} (3)
    (2) edge node {b} (1)
    (2) edge node {a} (4)
    (0) edge node {a} (1)
    (0) edge node {a} (2)
    (0) edge node {b} (1)
    (0) edge node {a} (3)
    (4) edge node {b} (1);
\end{tikzpicture}
\end{center}

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...

Berry-Sethi Approach: (sophisticated version)

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

- States: \( \{ \epsilon \} \cup \{ i | i \text{ a leaf} \} \)
- Start state: \( \epsilon \)
- Final states: \( \text{last}[e] \) if empty\([e] = f \)
  \( \{ \epsilon \} \cup \text{last}[e] \) otherwise
- Transitions: \( \{ \epsilon, a, \epsilon \} \) if \( i \in \text{first}[e] \) and \( i \text{ labeled with } a \)
  \( \{ i, a, \epsilon \} \) if \( i' \in \text{next}[i] \) and \( i' \text{ labeled with } a \)

We call the resulting automaton \( A_c \).

Lexical Analysis

Chapter 4:
Turning NFAs deterministic