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;
- $\Sigma$ a finite alphabet of inputs;
- $I \subseteq Q$ the set of start states;
- $F \subseteq Q$ the set of final states and
- $\delta$ the set of transitions (relation).
Finite Automata

Once again, more formally:

- We define the transitive closure $\delta^*$ of $\delta$ as the smallest set $\delta'$ with:
  
  $$(p, p, p) \in \delta' \quad \text{and} \quad (p, xw, q) \in \delta'$$

  if $(p, x, p_1) \in \delta$ and $(p_1, w, q) \in \delta'$.

$\delta^*$ 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:

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

Chapter 3:
Converting Regular Expressions to NFAs

In Linear Time from Regular Expressions to NFAs

Berry-Sethi Approach

Berry-Sethi Algorithm

Produces exactly $n + 1$ states without $\epsilon$-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 $\epsilon$ are reachable consuming the rest of input $w$. 

Thompson's Algorithm

Produces $O(n)$ states for regular expressions of length $n$. 

Lexical Analysis
Berry-Sethi Approach

Glushkov Automaton

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

Viktor M. Glushkov

Idea:

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

Berry-Sethi Approach

... for example:

$(a/b)^*a(a/b)$

```
                  *
                 / \
                *   *
               /   /   \
              l   a   l
             /   /   /   \
            a   b   a   b
```

Berry-Sethi Approach

In general:

- Input is only consumed at the leaves.
- Navigating the tree does not consume input $\rightarrow \epsilon$-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.
  $\Rightarrow$ we enumerate the leaves ...

Berry-Sethi Approach

... for example:

```
                  *
                 / \
                *   *
               /   /   \
              l   a   l
             /   /   /   \
            a   b   a   b
```
Berry-Sethi Approach (naive version)

Construction (naive version):

- States: $s_r$, $s_\epsilon$ with $r$ nodes of $\epsilon$;
- Start state: $s_\epsilon$;
- Final state: $s_\epsilon$;
- Transitions: for leaves $r = \star x$ we require: $(s_r, x, s_\epsilon)$.

The leftover transitions are:

<table>
<thead>
<tr>
<th>$r$</th>
<th>Transitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1 \cdot r_2$</td>
<td>$(s_\epsilon, \epsilon, s_r)$, $(s_\epsilon, \epsilon, s_\epsilon)$</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

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 $\epsilon$ ?

Berry-Sethi Approach

Discussion:

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

... for example:

$\emptyset[r] = t$ if and only if $\epsilon \in [r]$
Berry-Sethi Approach: 1st step

(\xi \mid r)

Implementation:

DFS post-order traversal

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

Otherwise:

\[
\begin{align*}
\text{empty}[r_1 \mid r_2] &= \text{empty}[r_1] \lor \text{empty}[r_2] \\
\text{empty}[r_1 \cdot r_2] &= \text{empty}[r_1] \land \text{empty}[r_2] \\
\text{empty}[r_1] &= \text{first}[r_1] \\
\text{empty}[r_1 ?] &= \text{first}[r_1]
\end{align*}
\]

Berry-Sethi Approach: 2nd step

The may-set of first reached read states: The set of read states, that may be reached from $\epsilon \cdot r$ (i.e. while descending into $r$) via sequences of $\epsilon$-transitions:

$\text{first}[r] = \{ i \mid r \in \delta \wedge i \in r \}$

... for example:

\begin{itemize}
\item $0 \rightarrow 1 \rightarrow 4 \rightarrow 3 \rightarrow 4$
\item $2 \rightarrow 4$
\item $3 \rightarrow 0 \rightarrow 1 \rightarrow 2$
\end{itemize}

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 $\epsilon$-transitions:

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

... for example:

\begin{itemize}
\item $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4$
\item $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$
\item $2 \rightarrow 3 \rightarrow 4$
\item $3 \rightarrow 4$
\end{itemize}
**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>$r_1 \mid r_2$</td>
<td>$\text{next}[r_1] = \text{next}[r]$</td>
</tr>
<tr>
<td>$\text{next}[r_2]$</td>
<td>$\text{next}[r]$</td>
</tr>
</tbody>
</table>
| $r_1 \circ r_2$ | $\text{next}[r_1] = \{ \text{first}[r_2] \cup \text{next}[r] \}$ if $\text{empty}[r_2] = t$
| | $\text{first}[r_2]$ if $\text{empty}[r_2] = f$
| $\text{next}[r_2]$ | $\text{next}[r]$ |
| $r_1^*$ | $\text{next}[r_1] = \text{first}[r] \cup \text{next}[r]$ |
| $r_1^?$ | $\text{next}[r_1] = \text{next}[r]$ |

---

**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:

The may-set of next read states: The set of read states reached after reading $r$, that may be reached next via sequences of $\epsilon$-transitions.

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

... for example:

```
0 1 2
0 1 0 1 f
0 1 0 1 f
0 1 0 1 f
3 4 f
3 4 f
3 4 f
```

---

**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 $\epsilon$-transitions.

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

... for example:

```
0 1 2
0 1 0 1 f
0 1 0 1 f
0 1 0 1 f
3 4 f
3 4 f
3 4 f
```

---

**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 $\epsilon$-transitions.

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

... for example:

```
0 1 2
0 1 0 1 f
0 1 0 1 f
0 1 0 1 f
3 4 f
3 4 f
3 4 f
```

---

**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 $\epsilon$-transitions.

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

... for example:

```
0 1 2
0 1 0 1 f
0 1 0 1 f
0 1 0 1 f
3 4 f
3 4 f
3 4 f
```

---

**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 $\epsilon$-transitions.

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

... for example:

```
0 1 2
0 1 0 1 f
0 1 0 1 f
0 1 0 1 f
3 4 f
3 4 f
3 4 f
```

---

**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 $\epsilon$-transitions.

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

... for example:

```
0 1 2
0 1 0 1 f
0 1 0 1 f
0 1 0 1 f
3 4 f
3 4 f
3 4 f
```

---

**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 $\epsilon$-transitions.

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

... for example:

```
0 1 2
0 1 0 1 f
0 1 0 1 f
0 1 0 1 f
3 4 f
3 4 f
3 4 f
```

---

**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 $\epsilon$-transitions.

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

... for example:

```
0 1 2
0 1 0 1 f
0 1 0 1 f
0 1 0 1 f
3 4 f
3 4 f
3 4 f
```

---

**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 $\epsilon$-transitions.

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

... for example:

```
0 1 2
0 1 0 1 f
0 1 0 1 f
0 1 0 1 f
3 4 f
3 4 f
3 4 f
```

---

**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 $\epsilon$-transitions.

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

... for example:

```
0 1 2
0 1 0 1 f
0 1 0 1 f
0 1 0 1 f
3 4 f
3 4 f
3 4 f
```

---

**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 $\epsilon$-transitions.

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

... for example:

```
0 1 2
0 1 0 1 f
0 1 0 1 f
0 1 0 1 f
3 4 f
3 4 f
3 4 f
```

---

**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 $\epsilon$-transitions.

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

... for example:

```
0 1 2
0 1 0 1 f
0 1 0 1 f
0 1 0 1 f
3 4 f
3 4 f
3 4 f
```
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: 
\[
\text{last}[r] = \{ i \mid i \in \Delta^\ast \mid (\Delta^\ast, i, r) \in \delta^\ast, x \neq \epsilon \}
\]

... for example:

```
      0 1 2
        f  

     2   2
    / \  /  \  
   0 1 3 4
  /   /   /   
 0 1 2 3 4
```

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

Otherwise:

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

```
Berry-Sethi Approach: (sophisticated version)

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

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

We call the resulting automaton \( A_e \).

Berry-Sethi Approach

... for example:

```
1 - a  2 - a  3 - a  4 - b
  b  a  a  b
```

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...