Lexical Analysis

Kapitel 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 $\varepsilon$-transitions and demonstrates $\rightarrow$ Equality Systems and $\rightarrow$ Attribute Grammars

Idea:
The automaton tracks (conventionally via a marker "•"), in the syntax tree of a regular expression, which subexpression in $e$ are reachable consuming the rest of input $w$. 

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

Simon: Compilerbau (22.04.2013)
Date: Mon Apr 22 14:20:22 CEST 2013
Duration: 91:47 min
Pages: 41
Berry-Sethi Approach

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

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

... for example:

\[ w = bbaa : \]

... for example:

\[ w = \emptyset bbaa : \]
Berry-Sethi Approach

... for example:

\[ w = bbbaa : \]

\[
\begin{array}{c}
\ast \\
/ \\
\ast \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
\end{array}
\]

Berry-Sethi Approach

... for example:

\[ w = bbaa : \]

\[
\begin{array}{c}
\ast \\
/ \\
\ast \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
\end{array}
\]

Berry-Sethi Approach

... for example:

\[ w = bbbaa : \]

\[
\begin{array}{c}
\ast \\
/ \\
\ast \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
\end{array}
\]

Berry-Sethi Approach

... for example:

\[ w = bbbaa : \]

\[
\begin{array}{c}
\ast \\
/ \\
\ast \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
a \\
/ \\
b \\
/ \\
\end{array}
\]
### Berry-Sethi Approach

**Attention:**
- Input is only consumed by the leaves.
- Navigation in the tree is done without consuming input, i.e. via $\varepsilon$-transition.
- For a formal construction we need to come up with identifiers for states.
- Therefore we use the subexpression, corresponding to the subtree, dominated by the particular node.
- There are possibly same subexpressions in one regular expression.

\[ \rightarrow \quad \text{we enumerate the leaves} \ldots \]

---

**... for example:**

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

\[
\begin{array}{c}
\ast \\
\ast \\
a \\
a \\
b \\
b \\
\end{array}
\]

---

**Berry-Sethi Approach**

---

**... for example:**

\[
\begin{array}{c}
\ast \\
\ast \\
a \\
\end{array}
\]

\[
\begin{array}{c}
a \\
b \\
a \\
b \\
\end{array}
\]
Berry-Sethi Approach

Construction:

- **States:** $\cdot r, \cdot e$ with $r$ nodes of $e$;
- **Start state:** $\cdot r$;
- **Final state:** $e \cdot e$;
- **Transitions:** for leaves $r \equiv r_x \ x$ we require: $(\cdot r, x, \cdot e)$.

The leftover transitions are:

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

Berry-Sethi Approach

Discussion:

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

⇒ **Strategy:** Avoid generating $\epsilon$-transitions

**Necessary node-attributes:**

- **empty** can the subexpression $r$ consume $\epsilon$?
- **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 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: Compute these attributes for the nodes via DFS!

Berry-Sethi Approach: 1st step

$\text{empty}(r) = 1$ if and only if $\epsilon \in [r]$

... for example:
Berry-Sethi Approach: 1st step

\[ \text{empty}[r] = \top \text{ if and only if } \epsilon \in [r] \]

... for example:

![Diagram]

Berry-Sethi Approach: 1st step

\[ \text{empty}[r] = \top \text{ if and only if } \epsilon \in [r] \]

... for example:

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

Implementation:
DFS post-order traversal

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

Otherwise:

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

Berry-Sethi Approach: 2nd step

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

... 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 \( r \) (i.e. while descending into \( r \)) via sequences of \( \epsilon \)-transitions: \( \text{first}[r] = \{ i \mid i \in r \mid (r \epsilon, \epsilon \{ \alpha \}) \in \delta^*, x \neq \epsilon \} \)

... 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 $r$ (i.e. while descending into $r$) via sequences of $\epsilon$-transitions: $\text{first}[r] = \{ i \in r \mid (r, \epsilon, i) \in \delta^*, x \neq \epsilon \}$

... for example:

```
Implementation:
DFS post-order traversal
```

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

Otherwise:

$\text{first}[r_1 \cdot r_2] = \text{first}[r_1] \cup \text{first}[r_2]$ if empty$[r_1] = i$

$\text{first}[r_1 \cdot ?] = \text{first}[r_1]$

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

... for example:
Berry-Sethi Approach: 3rd step

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

... for example:

Berry-Sethi Approach: 3rd step

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

... for example:
Berry-Sethi Approach: 3rd step

**Implementation:**

DFS pre-order traversal

For the root, we find:

\[ \text{next}[r] = \emptyset \]

Apart from that we distinguish, based on the context:

\[
\begin{array}{c|c|c}
 r & \text{next}[r] & \text{next}[r_1] \\
 \hline
 r_1 & \text{next}[r] & \text{next}[r] \\
 r_2 & \text{next}[r] & \text{next}[r] \\
 r_1 \cdot r_2 & \text{next}[r] & \text{next}[r] \\
 r_1^* & \text{next}[r] & \text{next}[r] \\
 r_1^2 & \text{next}[r] & \text{next}[r] \\
\end{array}
\]

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:

\[ \text{last}[r] = \{ i \text{ in } r | \{ \text{first}[r], \epsilon, r^* \} \in \delta^*, x \neq \epsilon \} \]

... for example:

```
\begin{verbatim}
012 f ()
2 01

34 f ()
2 f ()

012 f ()
34 f ()
\end{verbatim}
```

Berry-Sethi Approach: Integration

**Construction:** 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:**
  - last[\epsilon] \quad \text{if} \text{empty}[\epsilon] = f.
  - \{\epsilon\} \cup \text{last}[\epsilon] \quad \text{else.}
- **Transitions:**
  - \{\epsilon, a, i\} \quad \text{if} \text{first}[i] \text{ and } i \text{ labeled with } a.
  - \{\epsilon, a, i^*\} \quad \text{if} \text{first}[i] \text{ and } i^* \text{ labeled with } a \text{ beschriftet}

We call the resulting automaton \( A_r \).
Berry-Sethi Approach

... for example:

![Diagram of Berry-Sethi Approach]

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...