CS420

Introduction to the Theory of Computation

Lecture 11: Regular expressions & NFAs

Tiago Cogumbreiro
Today we will learn...

- Define a data type that represents Regular Expressions
- Define inductively acceptance for Regular Expressions
- Define equivalence of Regular Expressions
Typical solutions

- Collect all milestones and wildcard matching
- Analytical window functions
- Graph data model

v=WykSdgtLDD0

Pattern Matching @ Scale Using Finite State Machine, by Ajit Koti and Rashmi Shamprasad

Learn how Netflix engineers use Regular Expressions to explore their data.
Implementing Regular Expressions

- We identified a set of language-operators
- We want to explore their expressiveness: *What kind of questions can we pose using that set of operators?*
- How do we implement such theory in Coq?
Regular expressions

Inductive definition

\[
\text{Inductive } \text{regex} ::= \\
\text{r\_void}: \text{regex} \\
\text{r\_nil}: \text{regex} \\
\text{r\_char}: \text{Ascii.ascii} \rightarrow \text{regex} \\
\text{r\_app}: \text{regex} \rightarrow \text{regex} \rightarrow \text{regex} \\
\text{r\_union}: \text{regex} \rightarrow \text{regex} \rightarrow \text{regex} \\
\text{r\_star}: \text{regex} \rightarrow \text{regex}.
\]

Informal description

- \text{r\_void}: represents the Void language
- \text{r\_nil}: the empty string Nil language
- \text{r\_char}: the Char language
- \text{r\_union}: represents the union of two languages
- \text{r\_app}: represents concatenation of languages
- \text{r\_star}: represents zero-or-more copies of an expression
Informal description

- \texttt{r\_void}: the Void language
- \texttt{r\_nil}: the Nil language
- \texttt{r\_char}: the Char language
- \texttt{r\_union}: the Union operator (notation \(r_1 || r_2\))
- \texttt{r\_app}: the Append operator (notation \(r_1 ;; r_2\))
- \texttt{r\_star}: the Star operator

Exercises

1. Strings with a's and b's that end with "aa" "aa", "aaa", "baaa", "bbbbbaa"
2. Strings that have at an even number of a's "aa", "", "aaaa", "aaaaaa"
3. Nonempty strings that only contain any number of a's and b's "a", "b", "ab", "aaaaa", "bbbbbb", "abaaa"
4. Strings that interleave one "a" with one "b" "a", "b", "ab", "ba", "aba", "bab", "abab", "baba"
Exercise

Strings with a's and b's that end with "aa"
Examples: "aa", "aaa", "baaa", "bbbbbaa"
Exercise

Strings with a's and b's that end with "aa"
Examples: "aa", "aaa", "baaa", "bbbbbaa"

Solution

$(a|b)^* \cdot aa$
Exercise

Strings that have at an even number of a's
Examples: "aa", "", "aaaa", "aaaaaa"
Exercise

Strings that have at an even number of a's
Examples: "aa", "", "aaaa", "aaaaaa"

Solution

\( (aa)^* \)
Exercise

Nonempty strings that only contain any number of a's and b's
Examples: "a", "b", "ab", "aaaaa", "bbbbbb", "abaaa"
Exercise

Nonempty strings that only contain any number of a's and b's
Examples: "a", "b", "ab", "aaaaa", "bbbb", "abaaa"

Solution

\((a\mid b)^* \cdot (a\mid b)\)
Exercise

Strings that interleave one "a" with one "b"
Examples: "a", "b", "ab", "ba", "aba", "bab", "abab", "baba"
Exercise

Strings that interleave one "a" with one "b"
Examples: "a", "b", "ab", "ba", "aba", "bab", "abab", "baba"

Solution

\[(ab)^* | (ab)^* a | (ba)^* | (ba)^* b\]
Inductive propositions: acceptance

Rules accept_nil and accept_char

\[
\begin{align*}
\emptyset & \in \text{r}_\text{nil} \\
[c] & \in c
\end{align*}
\]

Rule accept_app

\[
\begin{align*}
w_1 & \in R_1 \\
w_2 & \in R_2
\end{align*} \quad \frac{w_1 \cdot w_2 \in R_1 \cdot R_2}{w_1 \cdot w_2 \in R_1 \cdot ;R_2}
\]

Rules accept_union_l and accept_union_r

\[
\begin{align*}
w & \in R_1 \\
\frac{w \in R_1 || R_2}{w \in R_1 || R_2}
\end{align*}
\]

Rules accept_star-nil and accept-star-cons_neq

\[
\begin{align*}
\emptyset & \in R^* \\
$w_1 \neq \emptyset$ & w_1 \in R \\
w_2 & \in R^*
\end{align*} \quad \frac{w_1 \cdot w_2 \in R^*}{w_1 \cdot w_2 \in R^*}
\]
Nondeterministic Finite Automata (NFA)
The diagram is a graph. Nodes are called states and edges are called transitions. Accepting a word: a path in the graph.

Examples: "aa", "aaa", "baaa", "bbbbbbbaa"

Strings with a's and b's that end with "aa"

State diagram

- The diagram is a graph
- Nodes are called states
- Edges are called transition
- Accepting a word: a path in the graph

- Initial state, identified start →
- Accepting state, double edge
- Consume one character per transition
- Comma in transitions means OR
NFA by example

Strings with a's and b's that end with "aa"
Examples: "aa", "aaa", "baaa", "bbbbbbbaa"

1. In $q_1$ read as many a's and b's as needed
2. Eventually, read one a and move to $q_2$
3. Finally, if we are able to read two a's, then we can accept the string ($q_3$)

As long as we can find one path, we can accept the input. There may exist multiple paths in the same state diagram (nondeterminism).
Exercise

Strings that have at an even number of a's
Examples: "aa", "", "aaaa", "aaaaaa"

1. State $q_1$ accepts the empty string
2. If we consume an a, then we have read an odd-number of a's. Thus, $q_2$ is non-accepting
3. If we read another a, we have read an even-number of a's. Thus, we go back to $q_1$, which is an accepting state.
Exercise

Nonempty strings that only contain any number of a's and b's
Examples: "a", "b", "ab", "aaaaa", "bbbbbb", "abaaa"
Exercise

Nonempty strings that only contain any number of a's and b's
Examples: "a", "b", "ab", "aaaaa", "bbbbbb", "abaaa"

- In state $q_1$ we can read as many a's as we want
- Eventually, we read at least one a or b and proceed to $q_2$
Exercise

Strings that interleave one "a" with one "b"
Examples: "a", "b", "ab", "ba", "aba", "bab", "abab", "baba"