CS420

Introduction to the Theory of Computation

Lecture 12: 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

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

\[
\text{Inductive } \text{regex} := \\
| \ r\_\text{void}: \text{regex} \\
| \ r\_\text{nil}: \text{regex} \\
| \ r\_\text{char}: \text{Ascii.ascii} \rightarrow \text{regex} \\
| \ r\_\text{app}: \text{regex} \rightarrow \text{regex} \rightarrow \text{regex} \\
| \ r\_\text{union}: \text{regex} \rightarrow \text{regex} \rightarrow \text{regex} \\
| \ r\_\text{star}: \text{regex} \rightarrow \text{regex}. \\
\]

Informal description

- `r_void`: represents the Void language
- `r_null`: the empty string Nil language
- `r_char`: the Char language
- `r_union`: represents the union of two languages
- `r_app`: represents concatenation of languages
- `r_star`: represents zero-or-more copies of an expression
Informal description

- `r_void`: the Void language
- `r_nil`: the Nil language
- `r_char`: the Char language
- `r_union`: the Union operator (notation `r1 || r2`)
- `r_app`: the Append operator (notation `r1 ;; r2`)
- `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

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

Strings that intertwine 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

\[
\varepsilon \in r_{\text{nil}} \quad [c] \in c
\]

Rule accept_app

\[
\frac{w_1 \in R_1 \quad w_2 \in R_2}{w_1 \cdot w_2 \in R_1; R_2}
\]

Rules accept_union_l and accept_union_r

\[
\frac{w \in R_1}{w \in R_1 \parallel R_2}
\quad
\frac{w \in R_2}{w \in R_1 \parallel R_2}
\]

Rules accept_star_nil and accept_star_cons_neq

\[
\varepsilon \in R^* \quad w_1 \neq \varepsilon \quad \frac{w_1 \in R \quad w_2 \in R^*}{w_1 \cdot w_2 \in R^*}
\]
Regex.v
Nondeterministic Finite Automata (NFA)
NFA by example

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

State diagram

About

- 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 \(\rightarrow\)
- 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", "bbbb", "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"
Exercise

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

- We start in an accepting state
- Reading an a moves us to $q_2$ which expects a b
- Reading a b moves us to $q_3$ which expects an a
- All states are accepting. **However, not all strings are accepted.**
Acceptance in an NFA

Acceptance is path finding

The given string must be a path from the starting node into the accepting node.
Acceptance in an NFA

Acceptance of $\text{a}bb\text{a}ba$

![Diagram of an NFA](image)
Acceptance in an NFA

Acceptance of $ababa$
Acceptance in an NFA

Acceptance of $aba$
Acceptance in an NFA

Acceptance of $abbaba$

Diagram:

- Start state $q_1$
- Transitions:
  - $a, b$ from $q_1$ to $q_1$
  - $a$ from $q_1$ to $q_2$
  - $b$ from $q_2$ to $q_3$
  - $a$ from $q_3$ to $q_4$

- Final state $q_4$
Acceptance in an NFA

Acceptance of $ababa$

\[
\begin{array}{c}
\text{start} \quad q_1 \quad a \quad q_2 \quad b \quad q_3 \quad a \quad q_4 \\
\end{array}
\]
Acceptance in an NFA

Acceptance of $\text{abba\text{a}}$

![NFA Diagram]

- Start state: $q_1$
- Transitions:
  - $q_1 \xrightarrow{a} q_2$
  - $q_2 \xrightarrow{b} q_3$
  - $q_3 \xrightarrow{a} q_4$
- Acceptance state: $q_4$
Acceptance in an NFA

Acceptance of abbaba
Acceptance in an NFA

- There are multiple concurrent possible paths and a current state
- Given a current state, if there are no transitions for a given input, the path ends
- Once we reach the final path, we check if the there are accepting states
Epsilon transitions
Epsilon transitions

Exercise 2

Let $\Sigma = \{a, b\}$. Give an NFA with four states that recognizes the following language

$$\{w \mid w \text{ contains the strings } aba \text{ or } aa\}$$

Note

- NFAs can also include $\epsilon$ transitions, which may be taken without consuming an input
Exercise 2: acceptance of $aaba$

Interleave input with $\epsilon$.

Read $a$
Exercise 2: acceptance of $a\epsilon aba$

Interleave input with $\epsilon$.
Read $\epsilon$
Exercise 2: acceptance of $aaba$

Interleave input with $\epsilon$.

Read $a$

```
... 
```
Exercise 2: acceptance of $aab\epsilon a$

Interleave input with $\epsilon$.

Read $\epsilon$
Exercise 2: acceptance of $aaba$

Interleave input with $\epsilon$.

Read $b$
Exercise 2: acceptance of $aab\epsilon$

Interleave input with $\epsilon$.

Read $a$
Exercise 2: acceptance of aabaε

Interleave input with ε.
Read ε
Exercise 2: acceptance of aaba

Interleave input with $\epsilon$.

Read $\epsilon$
Note $\epsilon$ transitions in the initial state

We looked at $\epsilon$ in the middle of the state diagram. Let us observe their effect in the initial state.
Exercise 2: acceptance of bd

Read $\epsilon$

[Diagram of a finite automaton with states $q_1$, $q_2$, $q_3$, $q_4$, and $q_5$. The transitions are labeled with $\epsilon$, $b$, $c$, and $d$.]
Exercise 2: acceptance of bd

Read b
Exercise 2: acceptance of $bd$

Read $\epsilon$ and then read $d$
Exercise 2: acceptance of bd

Accepted!