CS420

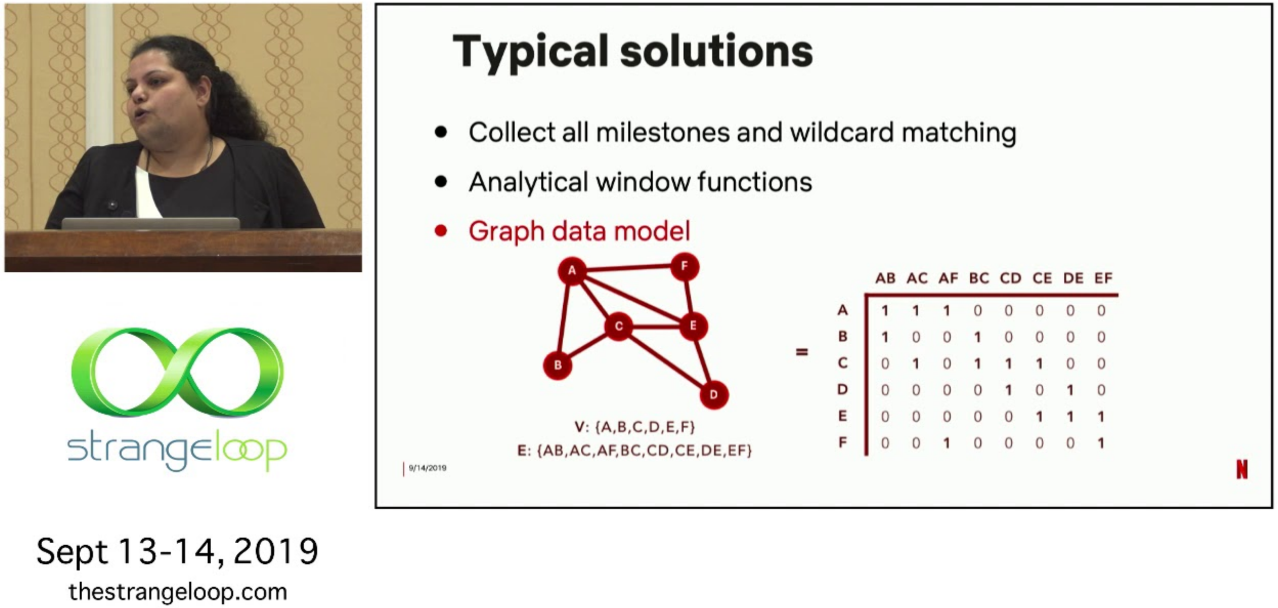
Introduction to the Theory of Computation

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



www.youtube.com/watch?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

```
Inductive regex :=
  | r_void: regex
  | r_nil: regex
  | r_char: Ascii.ascii → regex
  | r_app: regex → regex → regex
  | r_union: regex → regex → regex
  | r_star: regex → regex.
```

Informal description

- `r_void`: represents the `Void` language
- `r_nil`: 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

Regular expression Coq notation

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", "bbbbb", "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

$$(a||b)^\star \cdot aa$$

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

$(aa)^\star$

Exercise

Nonempty strings that only contain any number of a's and b's

Examples: "a", "b", "ab", "aaaaa", "bbbbb", "abaaa"

Exercise

Nonempty strings that only contain any number of a's and b's

Examples: "a", "b", "ab", "aaaaa", "bbbbb", "abaaa"

Solution

$(a||b)^\star \cdot (a||b)$

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

$$(ab)^\star || (ab)^\star a || (ba)^\star || (ba)^\star b$$

Inductive propositions: acceptance

Rules `accept_nil` and `accept_char`

$$[] \in \mathbf{r\_nil} \qquad\qquad [c] \in c$$

Rule `accept_app`

$$\frac{w_1 \in R_1 \qquad w_2 \in R_2}{w_1 \cdot w_2 \in R_1 \,;;\, R_2}$$

Rules `accept_union_l` and `accept_union_r`

$$\frac{w \in R_1}{w \in R_1 \mid\mid R_2} \qquad\qquad \frac{w \in R_2}{w \in R_1 \mid\mid R_2}$$

Rules `accept_star_nil` and `accept_star_cons_neq`

$$[] \in R^\star \qquad\qquad \frac{w_1 \neq [] \qquad w_1 \in R \qquad w_2 \in R^\star}{w_1 \cdot w_2 \in R^\star}$$

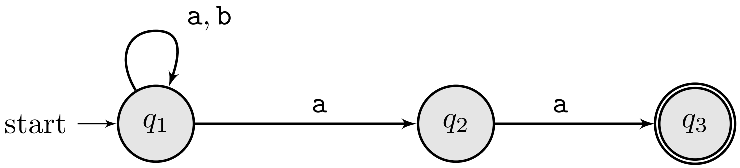
Regex.v

Nondeterministic Finite Automata (NFA)

NFA by example

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

State diagram

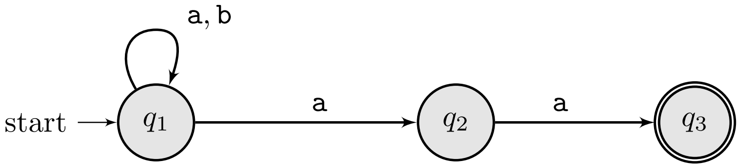


About

- 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` →
- 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", "bbbbbbaa"

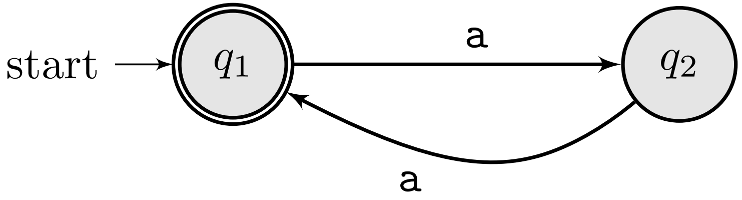


1. In q_1 read as many a's and b's as needed
2. Eventually, read one a and move to q_2
3. Finally, if we are able to read two a's, then we can accept the string (q_3)

> 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.

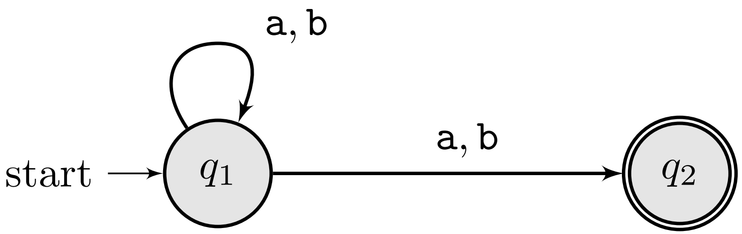
Exercise

Nonempty strings that only contain any number of a's and b's

Examples: "a", "b", "ab", "aaaaa", "bbbbb", "abaaa"

Exercise

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