COMS W3261
Computer Science Theory
Lecture 3
Regular Expressions

Alexander Roth

2014-09-08

Outline

- Review
- Regular Expressions
- Examples of regular expressions
- Finite automata with epsilon transitions

1 Review

- A deterministic finite automaton defines a regular language.
- In the last lecture we showed that using the subset construction we can transform a nondeterministic finite automaton into an equivalent DFA. Hence, NFA's also define the regular languages.
- An ϵ-NFA is an NFA with epsilon transitions. Using the subset construction we can transform an ϵ-NFA into an equivalent DFA. Hence, ϵ-NFA's are also another way to define the regular languages.
- In this lecture we will define another very different formalism called regular expressions that provide yet another way to define the regular languages

2 Regular Expressions

- A regular expression E is an algebraic expression that denotes a language $L(E)$.

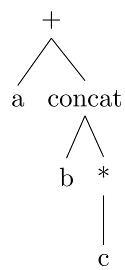
- Programming languages such as awk, java, javascript, perl, python use regular expressions to match patterns in strings.
- There are differences in the regular expression notations used by various programming languages, the most common variants being POSIX regular expressions and perl-compatible regular expressions.
- Virtually all regular-expressions notations have the operations of union, concatenation, and Kleene closure. We shall call regular expressions with just these three operator Kleene regular expressions.

Kleene regular expressions

- We can specify Kleene regular expressions over an alphabet Σ and the languages they denote using the following inductive definition:
  - The constants ε and ϕ are regular expressions that denote the language $\{ \varepsilon \}$ and $\{ \quad \}$, respectively.
  - A symbol c in Σ by itself is a regular expression that denotes the language $\{ c \}$
- Induction: Let E and F be regular expressions.
  - $E + F$ is a regular expression that denotes $L(E) \cup L(F)$.
  - EF is a regular expression that denotes $L(E)L(F)$, the concatenation of $L(E)$ and $L(F)$.
  - E^* is a regular expression that denotes $(L(E))^*$.
  - (E) is a regular expression that denotes $L(E)$.
- Precedence and associativity of the regular-expression operators
  - The regular-expression operator star has the highest precedence and is left associative.
  - The regular-expression operator concatenation has the next highest precedence and is left associative.
  - The regular-expression operator + has the lowest precedence and is left associative.
  - Thus the regular expression $a + b * c$ would be grouped $a + ((b^*)c$.
- If a regular expression E denotes a language L and a string w is in L, we will often say that E *matches* w.

Notes From Class

$a + b + c \equiv (a + b) + c$
Let's look at a syntax tree for this expression:



$E = a + bc^*$ and the language is $L(E) = \{a, b, bc, bcc, bccc, \ldots\}$
$\{a\} \cup \{bc^i \mid i \geq 0\} = \{a, bc^i | i \geq 0\}$
$E = a^*b^* \Longrightarrow L(E) = \{\epsilon, a, b, aa, ab, bb, \ldots\}$

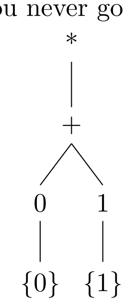
Examples of Kleene regular expressions and the languages they denote

- 0^*10^* denotes the set of all strings of 0's and 1's containing a single 1.
- $(0 + 1)^*1(0 + 1)^*$ denotes the set of all strings of 0's and 1's containing at least one 1.
- $1^*(01^*01^*)^*$ denotes the set of all strings of 0's and 1's containing an even number of 0's.
- $(a + b)^*abba(a + b)^*$ denotes the set of all strings of a's and b's containing the substring $abba$.

Notes from Class

- $(0 + 1)^*$ denotes all strings of 0's and 1's
- $(0^*1^*)^*$ denotes all strings of 0's and 1's
- 0^*10^* denotes all strings of 0's and 1's with exactly one 1.

You never go wrong by going back to basics.



Syntax tree for the $(0+1)^*$.
`g/re/p` – globally search for regular expression and print.
`egrep` – extended globally search for regular expression and print.

3 POSIX Regular Expressions

- The IEEE standards group POSIX added a number of additional operators to Kleene regular expressions to make it easier to specify languages. It also tried to standardize the different regular-expressions conventions used by various Unix utilities.
- Here we list some of the more useful Posix regular-expression operators and describe the string they match.

Some POSIX regular expression operators

1. Posix uses `?` to mean "zero or one instance of" The regular expression `a?b?c` denotes the language { {`ε , a, b, c, ab, ac, bc, abc`}. Thus `a?b?c` matches any of the eight strings in this language.
2. `.` matches any character except a newline.
3. `^` matches the empty string at the beginning of a line.
4. `$` matches the emtpy string at the end of a line.
5. `[abc]` matches an `a`, `b`, or `c`.
6. `[a-z]` matches any lowercase letter from `a` to `z`.
7. `[A-Za-z0-9]` matches any alphanumeric character.
8. `[^abc]` matches any character except an `a`, `b`, or `c`.
9. `[^0-9]` matches any nonnumeric character.
10. a^* matches an string of zero or more `a`'s (including the empty string).
11. `a?` matches any string of zero or one `a`'s (including the empty string)
12. `a`{`2,5`} matches any string consisting of two to five `a`'s.
13. `(a)` matches an `a`.
14. Note that in POSIX regular expressions the operator — (rather than +) is used to denote union. In POSIX regular expressions + means one or more instances of.
15. \ is a metacharacter that turns off any special meaning of the following character. For example `d\*g` matches the string `d*g`. Another example, \\ matches the string consisting of the single character \.

Examples of Posix regular expressions and the strings they match

- The Unix command `egrep ‘regexp‘ file` prints all lines in `file` that contains a substring matched by `regexp`. Examples:
  1. The command `egrep ‘dog‘ file` would print all lines in `file` containing the substring `dog`.
  2. The command `egrep ‘â?b?c?d?e?$‘ file` would print all lines in `file` consisting of the letters `a, b, c, d, e` in increasing alphabetic order. The metacharacters `^` and `$` match the empty string at the beginning and end of a line, respectively. `aegilops` is the longest English word whose letters are in increasing alphabetic order.

Notes from Class

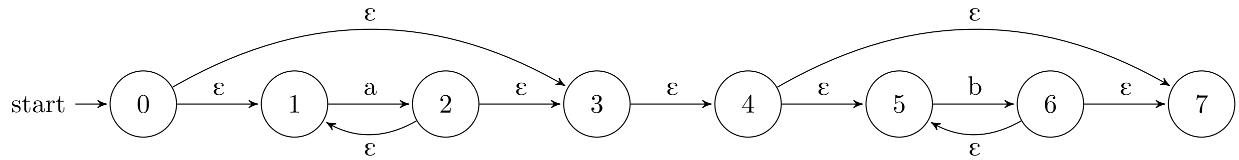
All words with exactly one vowel: `‘^[^aeiou]*[aeiou][^aeiou]*$’`
Mono-tomic word that goes up `‘^a?b?c?...z?$’`

4 ε-NFA: an NFA with Epsilon-Transitions

- An ε-NFA is an NFA $(Q, \Sigma, \delta, q_0, F)$ whose transition function δ is a mapping from $\Sigma \cup \{\varepsilon\}$ to $P(Q)$, the set of subsets of Q.
- The language of an ε-NFA is the set of all strings that spell out a path from the start state to a final state. There can be ε-transitions along this path.
- Epsilon-closures
  - We define ECLOSE(q), the ε-closure of a state q of an ε-NFA, recursively as follows:
    - State q is in ECLOSE(q).
    - If state p is in ECLOSE(q), then all states in $\delta(p, \varepsilon)$ are also in ECLOSE(q).
  - We can compute the ε-closure of a set of states S by taking the union of the ε-closures of all the individual states in S.

Notes from Class

We can simulate any ε-NFA with an equivalent DFA.
We have a^*b^*

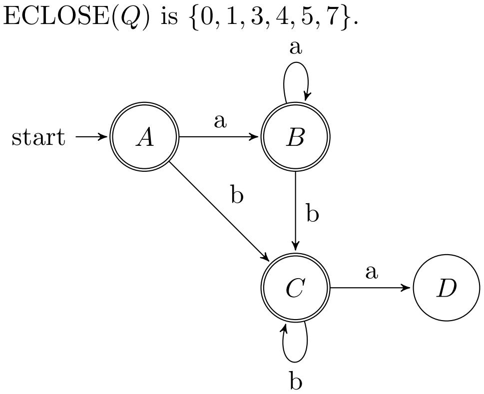


5 Converting an ε-NFA to an equivalent DFA

- We can eliminate all ε-transitions from an ε-NFA by converting it into an equivalent DFA using the subset construction.
- Given an ε-NFA $E = (Q_E, \Sigma, \delta_E, q_E, F_E)$, we construct the DFA $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$ as follows:
  - $Q_D = P(Q_E)$.
  - δ_D is computed all a in Σ and S in $P(Q_E)$ as follows:
    Let $S = \{p_1, p_2, \ldots, p_k\}$ and let $\{r_1, r_2, \ldots, r_m\}$ be the union of $\delta_E(p_i, a)$ for $i = 1, 2, \ldots, k$.
    Then, $\delta_D(S, a) = \text{ECLOSE}(\{r_1, r_2, \ldots, r_m\})$.
  - $q_D = \text{ECLOSE}(q_E)$.
  - $F_D = \{\, S \mid S \text{ is in } Q_D \text{ and } S \text{ contains a state in } F_E \,\}$.
- As with the subset construction, we can prove by induction that $L(D) = L(E)$.

Notes from Class

Q represents a set of ε-transitions. Using the diagram from the last section ECLOSE(Q) is $\{0, 1, 3, 4, 5, 7\}$.



6 Practice Problems

1. Do the two regular expressions $(a + b)^*$ and $(a^*b^*)^*$ denote the same language?
2. Write a Kleene regular expression for all strings of 0's and 1's with an even number of 0's and an even number of 1's.
3. Let L be the language { `abxba`|`x`is any string of `a`'s,`b`'s, and `c`'s not containing `ba`}. This language models comments in the programming language C.

(a) Construct a regular expression for L.

(b) Show how your regular expression defines the string `abcbaba`.

4. Write a Kleene regular expression for all strings of a's and b's that begin and end with an a.

5. Write a Posix regular expression that matches all English words ending in dous.

6. Write a Posix regular expression that matches all English words with five vowels a,e,i,o,u in order. (The vowels do not have to be next to one another.)

7. HMU Exercise 2.5.1.

7 References

- HMU: Ch. 2, Sects. 3.1, 3.3.1
- `http://en.wikipedia.org/wiki/Regular_expression`