CMPSCI 250: Introduction to Computation

Lecture #28: Regular Expressions and Languages
David Mix Barrington
2 April 2014

Regular Expressions and Languages

- Regular Expressions
- The Formal Inductive Definition
- The Kleene Star Operation
- Finite Languages
- The Language $(a + ab)^*$
- Logically Describable Languages
- Languages From Number Theory

Regular Expressions

- We're now entering the final segment of the course, dealing with regular expressions and finite-state machines. A **regular expression** is a way to denote a **language** (a subset of Σ^* for some finite alphabet Σ).
- A **finite-state machine** is a particular kind of computer that reads a string in Σ^* and gives a boolean answer. Thus the machine **decides** some language.

Regular Expressions

- Our major result will be **Kleene's Theorem**, which says that a language is *denoted* by a regular expression (is a regular language) if and only if it is *decided* by a finite-state machine.
- We'll learn algorithms to go from an expression to an equivalent machine, and vice versa.

Regular Expressions

- Regular expressions, with slightly different notation, occur in programming languages and operating systems such as Unix.
- It's common for a language (or a property of strings) to be defined by a regular expression. The system then has to build a finite-state machine to decide that language, and it uses the very algorithm we will present here.

The Formal Inductive Definition

- Fix an alphabet Σ. A **regular expression** over Σ is a string over the alphabet $\Sigma \cup \{\emptyset, +, \cdot, ^*, (,)\}$ that can be built by the rules below.
- Each regular expression R denotes a language L(R), also determined by the rules below.
- "$\emptyset$" is a regular expression and denotes the empty set.
- If a is any letter in Σ, then "a" is a regular expression and denotes the language {a}.

The Formal Inductive Definition

- If R and S are two regular expressions denoting languages L(R) and L(S), then "R·S" (often written "RS") is a regular expression denoting the concatenation L(R)L(S), and "R+S" is a regular expression denoting the union L(R) ∪ L(S).
- If R is a regular expression denoting the language L(R), then "R^*" is a regular expression denoting the Kleene star of L(R), which is written $L(R)^*$.
- Nothing else is a regular expression.

The Kleene Star Operation

- If A is any language, the **Kleene star** of A, written A^*, is the set of all strings that can be written as the concatenation of zero or more strings from A.
- If $A = \emptyset$, $A^* = \{\lambda\}$ because we can only have a concatenation of zero strings from A.
- If $A = \{a\}$, then $A^* = \{\lambda, a, aa, aaa, aaaa, \ldots\}$, the set of all strings of a's.

The Kleene Star Operation

- If $A = \Sigma$, then A^* is just Σ^*, so the star notation we have been using for "Σ^*" is just this same Kleene star operation. A string over Σ is just the concatenation of zero or more letters from Σ.
- In general, A^* is the union of the languages A^0, $A^1, A^2, A^3, \ldots$ where $A^0 = \{\lambda\}, A^1 = A, A^2 = AA$, $A^3 = AAA$, and so on. (Note that some of the laws of exponents still work, like $A^iA^j = A^{i+j}$ and $(A^i)^j = A^{ij}$.)

Finite Languages

- The regular expression "aba" denotes the concatenation {a}{b}{a} = {aba}, by the definition of concatenation of languages.
- Thus any language consisting of a single non-empty string has a regular expression, which is (up to a type cast) itself.
- The language {λ}, as we just saw, can be written "$\emptyset^*$".

Finite Languages

- If I have any **finite** language, I can denote it by a regular expression, as the union of the one-string languages for each of the strings in it.
- For example, the finite language {λ, aba, abbb, b} is denoted by the regular expression "$\emptyset^*$ + aba + abbb + b".
- (Note that this "+" is *not* the Java concatenation operator!)

Finite Languages

- A regular expression that never uses the star operator *must* denote a finite set of non-empty strings. (We can prove this fact using induction!)
- If we use the star operator on any language that contains a *non-empty* string, the result is an **infinite** language, such as $(aa)^* = \{\lambda, aa, aaaa, aaaaaa, \ldots\}$.

Clicker Question #1

- Which of these sets of strings is denoted by the regular expression (ab + ba)(aa + bb)ba?
- (a) {ab, ba, aa, bb, ba}
- (b) {aabbba, baaaba, babbba, abaaba}
- (c) {abbbba, baaaba, abaaba, babbba}
- (d) {abbaaabbba}

Answer #1

- Which of these sets of strings is denoted by the regular expression (ab + ba)(aa + bb)ba?
- (a) {ab, ba, aa, bb, ba}
- (b) {aabbba, baaaba, babbba, abaaba}
- (c) *{abbbba, baaaba, abaaba, babbba}*
- (d) {abbaaabbba}
- (in lecture none of the four were correct)

The Language $(a + ab)^*$

- Here is a more interesting regular language, denoted by the regular expression "$(a + ab)^*$. (Note that the parentheses are important -- "$a + ab^*$" and "$a + (ab)^*$" denote quite different languages.)
- The strings in $(a + ab)^*$ are exactly those strings that can be made by concatenating zero or more strings, each of which is equal to either a or ab.

The Language $(a + ab)^*$

- We can systematically list $(a + ab)^*$ by listing $(a + ab)^i$ in turn for each natural i.
- We get $(a + ab)^0 = \{\lambda\}$, $(a + ab)^1 = \{a, ab\}$, $(a + ab)^2 = \{aa, aab, aba, abab\}$, $(a + ab)^3 = (aaa, aaab, aaba, aabab, abaa, abaab, ababa, ababab\}$, and so forth.

The Language $(a + ab)^*$

- How can we tell whether a given string of a's and b's is in $(a + ab)^*$?
- If it ends in a, we know that the last string used in the concatenation was "a", and if it ends in b, the last string used was "ab". So we can delete a's and ab's from the right as long as we can, and if we produce λ then the string was in the language.
- It turns out that $(a + ab)^*$ is the set of strings that don't begin with b and never have two b's in a row. (How would you prove this assertion?)

Clicker Question #2

- Let $X \subseteq \{a, b\}^*$ be the language denoted by the regular expression $(ab + bbb)^*$. Which of these is *not* true of every string in X?
- (a) The string must end with b.
- (b) There are at least as many b's as a's.
- (c) There are never two a's in a row.
- (d) There cannot be two a's separated by exactly five letters that are all b's.

Answer #2

- Let $X \subseteq \{a, b\}^*$ be the language denoted by the regular expression $(ab + bbb)^*$. Which of these is *not* true of every string in X?
- *(a) The string must end with b. (λ doesn't.)*
- (b) There are at least as many b's as a's.
- (c) There are never two a's in a row.
- (d) There cannot be two a's separated by exactly five letters that are all b's.

Logically Describable Languages

- We can say "the first letter is not b and there are never two b's in a row" in the predicate calculus.
- One way to do it is to have variables that range over positions in the string.
- Our atomic predicates are "$C_a(x)$" ("position x contains an a", "$C_b(x)$" ("position x contains a b"), "x = y" ("x and y are the same position"), and "x < y" ("x is to the left of y").

Logically Describable Languages

- So we can say that the first letter is not b, "$\neg\exists x$: $C_b(x) \wedge \forall y$: $x \leq y$", and that there are never two b's in a row, "$\neg\exists x$: $\exists y$: $C_b(x) \wedge C_b(y) \wedge (x < y) \wedge \forall z$: $(z \leq x) \vee (z \geq y)$".
- One way to say both things at once is "$\forall x$: $C_b(x) \rightarrow \exists y$: $Pred(y, x) \wedge C_a(y)$", where "$Pred(y, x)$" abbreviates "$(y < x) \wedge \forall z$: $(z \leq y) \vee (z \geq x)$".
- It turns out that languages are logically describable if and only if they have a regular expression of a certain kind, and that $(aa)^*$ is not describable.

Clicker Question #3

- Let L be the language described by the logical formula "$\forall x: \exists y: (x \leq y) \wedge C_b(y)$". Consider the three strings λ, baa, and abba. *How many* of them are in the language L?
- (a) all three
- (b) two
- (c) one
- (d) none

Answer #3

- Let L be the language described by the logical formula "$\forall x: \exists y: (x \leq y) \wedge C_b(y)$". Consider the three strings λ, baa, and abba. *How many* of them are in the language L?
- (a) all three
- (b) two
- *(c) one (statement says "no last letter of a")*
- (d) none

Languages From Number Theory

- We can easily make a regular expression for the set of even-length strings of a's, "$(aa)^*$", or the odd-length strings of a's, "$(aa)^*a$", or the set of strings of a's whose length is congruent to 3 modulo 7, "$a^3(a^7)^*$", or the set of strings whose length is congruent to 1, 2, or 5 modulo 6, "$(a + a^2 + a^5)(a^6)^*$".
- What about the set of strings over {a,b} that have an even number of a's? A good first guess is that such a string is a concatenation of zero or more strings, each of which has exactly two a's. This would be the language $(b^*ab^*ab^*)^*$.

Languages From Number Theory

- But this isn't exactly right, because "bb", for example, has 0 a's and 0 is even. A correct expression for this language is $(b + ab^*a)^*$ -- we can divide any such string into pieces which either have exactly two a's (with some number of b's between) or are just b's themselves.
- It's harder to get the strings with a number of a's congruent to 3 mod 7, or the strings with an even number of a's *and* an even number of b's, but both are possible.