Lecture II
REGULAR EXPRESSIONS

Lecture 2: Honors Theory, Spring'02.

We study some additional properties of regular languages. First, we introduce regular expressions as another way to specify regular languages. The pumping lemma for regular languages is introduced as a method for proving irregularity.

§1. Regular Operations

We can view a language A simply as a set, and thus it is common to talk about "regular sets" instead of "regular languages". Viewed as sets, we obtain the usual set-theoretic operations such as complement, union, intersection, set difference and symmetric difference:

$$\overline{A}, A \cup B, A \cap B, A \setminus B, A \oplus B.$$

One comment about complement: this is only well-defined when A is implicitly viewed as the subset of some set U, in which case $\overline{A} = U \setminus A$. But what is U? There is a natural candidate for U, namely $U = \Sigma_A^*$ where Σ_A comprise all the symbols that appear in some word of A. Sometimes, we wish to view a language A as a subset of Σ^* where Σ is a proper superset of Σ_A. In this case $\overline{A} = \Sigma^* \setminus A$. In view of this inherent ambiguity, we will formally define a **language** as a pair (A, Σ) where $A \subseteq \Sigma^*$ for some alphabet Σ. When we union two languages, we should also form the union of their underlying alphabet. Similar remarks apply to the other operations. Having said this, we continue to abuse notation and write languages as sets. Usually, the implied alphabet of A can be taken to be Σ_A.

Among the set-theoretic operations above, only union $A \cup B$ is intimately connected with in the concept of regular sets. We introduce two other operations on languages:

- Concatenation: $A \cdot B = \{u \cdot v : u \in A, v \in B\}$.
- Kleene Star: $A^* = \{w_1 w_2 \cdots w_n : w_i \in A, n \geq 0\}$

The three operations of union, concatenation and Kleene star constitute the **regular operations**. It follows from the definition of Kleene Star that $\epsilon \in A^*$ (choose $n = 0$) for any A. As usual, we may imply the concatenation operator by juxtaposition: $A \cdot B$ can be simply written as AB, It is easy to show simple properties such as $(A^*)^* = A^*$ and $(AB)C = A(BC)$.

Theorem 1 *The class REG is closed under the regular operations.*

We postpone the proof, which is easily done using nondeterminism.

§2. Regular Expressions

A regular expression is an algebraic expression in which the operators are regular operators. Here is a simple example of a regular expression

$$1 \cdot (0 + 1) * \cdot 0. \tag{1}$$

It denotes the language $\{1 \cdot w : w \in 0, 1^*\}$. A regular expression is a syntactical object (i.e., a sequence of symbols), but it denotes a language.

We formally define the concept of a regular expression. We need to reserve some special symbols

$$\underline{(}, \underline{)}, \underline{\epsilon}, \underline{\emptyset}, \underline{+}, \underline{\cdot}, \underline{^*}$$

which is assumed not to be in any alphabet. If Σ is an alphabet, let $\underline{\Sigma}$ denote the union of Σ with the set of these special symbols. Then a **regular expression** over an alphabet Σ is a string $R \in \underline{\Sigma}^*$ of the form:

- (BASIS) R is a or $\underline{\epsilon}$ or $\underline{\emptyset}$, where $a \in \Sigma$.
- (INDUCTION) R is $R_1 \underline{\cdot} R_2$ or $\underline{(} R_1 \underline{+} R_2 \underline{)}$ or $\underline{(} R_1 \underline{^*} \underline{)}$, where R_1, R_2 are regular expressions.

NOTE: Often, the operator $\underline{\cdot}$ is omitted (so that $1w0$ really stands for $1 \underline{\cdot} w \underline{\cdot} 0$. Also, if parenthesis are omitted, we use the operator precedence: $\underline{*} \succ \underline{\cdot} \succ \underline{+}$. E.g., $10\underline{+}1\underline{^*}$ is a short hand for the regular expression $\underline{(}1\underline{\cdot}0\underline{)}\underline{+}\underline{(}1\underline{^*}\underline{)}$.

The language $L(R)$ denoted by a regular expression R is defined as follows:

- (BASIS) $L(a) = \{a\}$, $L(\underline{\epsilon}) = \{\epsilon\}$ and $L(\underline{\emptyset}) = \emptyset$, where $a \in \Sigma$.
- (INDUCTION) $L(R_1 \underline{\cdot} R_2) = L(R_1) \cdot L(R_2)$, $L(ullpR_1 \underline{+} R_2 \underline{)}) = L(R_1) \cup L(R_2)$ and $L(ullpR_1 \underline{^*} \underline{)}) = L(R_1)^*$.

Note that the definition of $L(R)$ depends on the regular operations on languages. Also, there is a close parallel between syntax and semantics: $\underline{\cdot}$ denotes concatenation "$\cdot$" $\underline{+}$ denotes union "$\cup$", $\underline{^*}$ denotes Kleene star "*". Because of this close parallel, it is common to abuse of notation by writing $\cdot$ instead of $\underline{\cdot}$, $+$ instead of $\underline{+}$ and * instead of $\underline{^*}$ in regular expressions. This is seen in our original example (1) above.

In the literature, the union symbol $\cup$ is used instead of $\underline{+}$ in regular expressions. We prefer $\underline{+}$ as this make regular expressions look like arithmetic expressions, and thus obeying the usual precedence rules.

Uses of Regular Expressions. The fact that we can describe regular sets syntactically, in a linear sequence of symbols, is very useful in many applications. Unix utilities such as awk, grep, sed exploit this fact. Text editors use regular expressions in their search facility. Many programming language use regular expressions in various ways. For instance, Perl has regular expressions built into many of its operators.

§3. Regular Expression

The main result about regular expressions is this:

Theorem 2 *The regular expressions denotes all and only regular languages.*

Proof. In one direction, we must show that any regular expression R denotes a regular language: this is easy. It is trivial in the basis case. In the inductive case, we know that REG is closed under the regular operations.

In the other direction, we want to show that every dfa M gives rise to a regular expression R_M such that $L(M) = L(R_M)$. This is slightly harder.

We introduce generalized nfa's to do this proof. It is defined by allowing the transitions to specify a regular expression. This means that we can take a given transition by reading any substring of the input that belongs to the corresponding regular set. For now we simply accept that such nfa's can only accept regular sets.

The proof is now easy to complete.

Q.E.D.

Note the special form of a gnfa implied by this definition. CLAIM: for every nfa, we can construct an equivalent gnfa. Proof that to every nfa, we can construct an equivalent 2-state gnfa

§4. Pumping Lemma

We consider a natural question: are there langauges that are not regular? How do we show that any particular languages that are not regular? Consider the following languages.

- $L_1 = \{0^n 1^n : n \geq 0\}$.
- $L_2 = \{w \in 0, 1^* : \#_1(w) = \#_0(w)\}$.
- $L_3 = \{w \in 0, 1^* : \#_0 1(w) = \#_1 0(w)\}$.

We might argue that L_1 is non-regular since any dfa has only finitely many states. This informal argument may be persuasive, but is actually no proof! For, we might argue in the same way about L_2 and L_3. This turns out to be false for L_3.

So we need some tools for showing non-regularity of a language. The most important tool for this is the pumping lemma which we now introduce.

Theorem 3 (Pumping Lemma for Regular Sets) *If L is a regular language, then there exists a $p > 0$ (called the pumping number) such that for all words in $w \in L$ with length at least p, we can write $w = xyz$ such that*
(i) For all $i \geq 0$, $xy^i z \in L$.
(ii) $|y| > 0$
(iii) $|xy| \leq p$

Proof. Let p to be the number of states in a dfa M that accepts L. Suppose $w \in L$ and $n = |w| \geq p$. Let w_i denote the prefix of w of length i, $i \geq 0$. Consider the sequence

$$q_0, q_1, \ldots, q_n$$

of states where $q_i = \delta^*(w_i)$. Thus $w_0 = \epsilon$ and q_0 is the start state of M.

Since there are more than p states in this sequence, some state must be repeated in this sequence. Choose $0 \leq i < j \leq p$ such that $q_i = q_j$. Then we may write $w = xyz$ where $x = w_i$ and $xy = w_j$. It is now easy to verify that (i), (ii) and (iii) holds.

Q.E.D.

E.g., SHOW THAT NON-PALINDROMES ARE NON-REGULAR.

§5. Myhill-Nerode Theorem

Another way to prove non-regularity is through a characterization of regular sets due to J. Myhill and A. Nerode.

For any language $L \subseteq \Sigma^*$, we define the **L-equivalence** relationship on Σ^* as follows: for $x, y \in \Sigma^*$ are L-equivalent, written $x \equiv_L y$, if

$$(\forall z \in \Sigma^*)[xz \in L \text{iff } yz \in L]$$

In particular, $x \equiv_L y$ implies that both x and y belong to L or both do not belong to L. This relationship is seen to be an equivalence relation and for any $x \in \Sigma^*$, we let $[x]_L$ denote its equivalence class modulo $\equiv_L$.

Theorem 4 (Myhill-Nerode) *A language L is regular if and only if $\equiv_L$ has finitely many equivalence classes.*

Before we prove this, let us see illustrate an application. If $L = \{0^n 1^n : n \geq 0\}$, then clearly $[0^i]_L \neq [0^j]_L$ for all $i \neq j$. Hence L is not regular.

NOTES

The Myhill-Nerode theorem is from [1].

Exercises

Exercise 5.1: Let $L = \{0^n : n \text{ is prime}\}$. Show that L is not regular. ♢

Exercise 5.2: Note that the pumping lemma, unlike the Myhill-Nerode theorem, is not claimed to be a characterization of regular languages. Is there a non-regular language that satisfies pumping lemma? ♢

End Exercises

"Kleene-ness is next to Gödelness"

References

[1] A. Nerode. Linear automaton transformations. *Proc. AMS*, 9:541–544, 1958.