Lecture 2: Languages via Logical Formulae

Languages can be defined (and this is perhaps the most natural way to define them) as words satisfying a particular property. For example, the set of words satisfying each of the following properties is a regular language:

1. Every occurance of an a is eventually followed by a b.
2. There is exactly one a.
3. The first letter is an a.
4. The last letter is a a.
5. Every occurance of a is immediately followed by a b.
6. There are even number of a's.

However, the set of words which satisfy the property *there are equal number of as and bs* is not a regular language. So what identifies properties that define regular languages?

1 First order logic of words

A natural formal language to describe properties such as the ones described above is the *first order logic over words*. This logic is quite easy to understand and we shall illustrate this logic with a few examples. For instance, consider the formula,

$$\forall x.\ (a(x) \Rightarrow \exists y.\ ((y > x) \wedge b(x))$$

Here, the variables x, y etc. refer to positions in the word. The formula $a(x)$ asserts that the letter at position x is a. The quantifiers have the usual meaning. The formula $y > x$ is true if the position y appears somewhere to the right of the position x. Thus, a word w satisfies the above formula only if for any position (x) with the letter a, there is some position to its right (y) with the letter b. This captures in the language of first order logic the first property listed above.

The second property can be expressed as

$$\forall x.\ \forall y.\ (a(x) \wedge a(y)) \Rightarrow x = y$$

Consider the formula $\text{First}(x) \stackrel{\triangle}{=} \forall y.(x = y) \vee (x < y)$. This formula evaluates to true at a position x if and only if it is the first position in the word. With this, the third property can be expressed as

$$\forall x.(a(x) \Rightarrow \text{First}(x))$$

Consider the formula $(x < y) \wedge \forall z.(x < z) \Rightarrow (y \leq z)$ (where $y \leq z$ is $(y = z) \vee (y < z)$.) It asserts that the position y is the position immediately to the right of the position x. We

shall write $(y = x + 1)$ to denote this formula and using this we write the fourth property above as

$$\forall x.a(x) \Rightarrow \exists y.(y = x + 1) \wedge a(y)$$

As it turns out, the fifth property, which does describe a regular language, has no equivalent in the first order logic of words. This is one of the results that we shall prove. But for the moment, we shall turn to extending the first order logic of words so that all regular languages can be described.

2 Monadic logic over words

In first order logic, the only kind of variables we have are those that represent positions in the word. In *monadic logic*, in addition to the position variables, we are also allowed to use variables that represent sets of positions. We shall use $X, Y, \ldots$ to denote variables that range over sets of positions. We are also allowed to quantify over such variables. For example

$$\forall X.\forall x.(x \in X) \Rightarrow a(x)$$

asserts that in every subset of positions, every position it contains has the letter a. Clearly this is true of a word if and only if every position in the word has the letter a. The following formula

$$\forall X.\exists x.(x \in X) \wedge a(x)$$

is not true of any word!! This is because, when X is the empty set of positions $\exists x.(x \in X) \wedge a(x)$ cannot be satisfied.

Once we have quantification over sets, we can describe the set of even length words as follows:

The set of all positions can be divided into two sets X and Y such that

1. The first position is in X
2. The last position is in Y
3. for each position x if x is in X then the next position (if it exists) lies in Y
4. for each position x if x is in Y then the next position (if it exists) is in X

The third and fourth properities ensure that the positions of the word are alternately in X and Y and the first and second assertions ensure that there are even number of alternations and thus the word must have even number of positions! The monadic formula expressing the above properties is:

$$\begin{array}{ll} \exists X.\exists Y. & (\forall x.(x \in X) \iff (x \notin Y) \\ & \wedge\forall x.\mathrm{First}(x) \Rightarrow (x \in X) \\ & \wedge\forall x.\mathrm{Last}(x) \Rightarrow (x \in Y) \\ & \wedge\forall x.\forall y.\ ((x \in X) \wedge (y = x + 1)) \Rightarrow (y \in Y) \\ & \wedge\forall x.\forall y.\ ((x \in Y) \wedge (y = x + 1)) \Rightarrow (y \in X)) \end{array}$$

The first line asserts that the set of all positions is divided into two sets X and Y and the following four lines describe the four properties mentioned above.

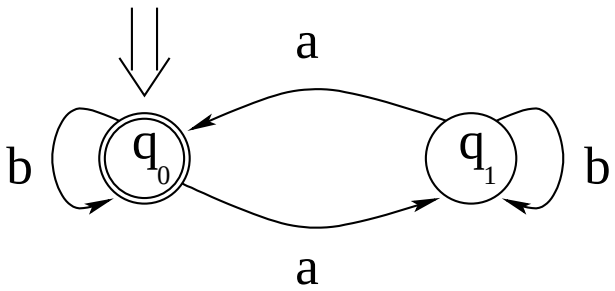
This extended logic which includes the use of set variables (and $x \in X$) is called the second-order monadic logic of words (or second-order monadic logic of order). We shall often write MSO to denote this logic. It is also called S1S (or second-order monadic logic of one successor).

3 Monadic logic and regular languages

Büchi and Elgot showed that the class of languages definable using formulas in monadic logic over words is precisely the class of regular languages.

Theorem 1 *(Büchi/Elgot) A language L is regular if and only if it can be described using a formula in MSO.*

The translation from a finite automaton accepting a regular language to a MSO formula describing the same language is the easier direction. Before we give the details, we shall illustrate the ideas with a simple example. Consider the following automaton with 2 states.



Let us examine a run of this automaton on some word, say *ababaa*.

$$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_1 \xrightarrow{a} q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_0$$

Observe that a run on a word of length n involves a sequence of states of length $n+1$. For the moment if we omit the last state reached on a input of length n, we have a sequence of of states of length n. Thus, we can think of the run (without the last state) as a labelling of the positions of the word with states (the state reached before the letter at that position is read). Can we then write out a formula that "describes" the run of the automaton on any given word?

Here is an outline of how to do this: The trick is to use one second order variable X_q for each $q \in Q$, and use X_q to pick out the positions that are labelled by state q. Let the states of the automaton be $q_0, q_1, \ldots q_k$. (Thus, for the above example we use two variables X_{q_0} and X_{q_1}.) We then assert that the set of positions can be decomposed into $|Q|$ sets, $X_{q_0}, X_{q_1}, \ldots X_{q_k}$ such that

1. The first position belongs to X_{q_0} (since q_0 is the start state of the automaton).

2. If x and $x+1$ are positions in the given word and the letter at position x is some a then, x and $x+1$ belong to sets X_p and X_q such that $\delta(p,a) = q$.

3. If x is the last position in the word w, the letter at this position is a and x belongs to X_q then $\delta(q,a) \in F$.

For the above automaton these assertions can be expressed in MSO as follows:

$$\begin{array}{ll} \exists X_{q_0}.\exists X_{q_1}. & (\forall x.(x \in X_{q_0}) \iff (x \notin X_{q_1}) \\ & \wedge \forall x.\mathrm{First}(x) \Rightarrow (x \in X_{q_0}) \\ & \wedge \forall x.\forall y.\ ((x \in X_{q_0}) \wedge a(x) \wedge (y = x+1)) \Rightarrow (y \in X_{q_1}) \\ & \wedge \forall x.\forall y.\ ((x \in X_{q_0}) \wedge b(x) \wedge (y = x+1)) \Rightarrow (y \in X_{q_0}) \\ & \wedge \forall x.\forall y.\ ((x \in X_{q_1}) \wedge a(x) \wedge (y = x+1)) \Rightarrow (y \in X_{q_0}) \\ & \wedge \forall x.\forall y.\ ((x \in X_{q_1}) \wedge b(x) \wedge (y = x+1)) \Rightarrow (y \in X_{q_1}) \\ & \wedge \forall x.(\mathrm{Last}(x) \Rightarrow ((x \in X_{q_0} \wedge b(x)) \vee (x \in X_{q_1} \wedge a(x))) \end{array}$$

The first line says each position is labelled either with q_0 or q_1 (to represent the state reached before leading the letter at that position). The second line asserts that state corresponding to the first position is the start state. The subsequent four lines encode the four possible transitions of the automaton. The last line ensures that the state assigned to the last position is such that the transition from this state on the letter at the last position takes the run into a final state.

In general given a finite automaton $A = (\{q_0, \ldots, q_k\}, \Sigma, \delta, q_0, F)$, the following formula asserts that " there is an accepting run for A over the given word". Thus the models of this formula are precisely the words in the language accepted by A.

$$\begin{array}{ll} \exists X_{q_0}.\exists X_{q_1} \ldots \exists X_{q_k}. & \forall x.\ \bigvee_{0 \le i \le k} x \in X_{q_i} \\ & \forall x.\ \bigwedge_{i \neq j} (x \in X_i) \Rightarrow \neg(x \in X_j) \\ & \forall x.\mathrm{First}(x) \Rightarrow x \in X_{q_0} \\ & \bigwedge_{\delta(p,a)=q} \forall x.((x \in X_p) \wedge a(x) \wedge (y = x+1)) \Rightarrow (y \in X_q) \\ & \forall x.\mathrm{Last}(x) \Rightarrow (\bigvee_{\delta(p,a) \in F}((x \in X_p) \wedge a(x))) \end{array}$$

The first and second lines assert that the set of positions is decomposed into a collection of sets, one for each state in Q. The the next three lines assert the 3 properties mentioned above and thus a word satisfies this property if and only if it is accepted by the automaton A. This completes a sketch of the argument showing that every regular language can be described by a MSO formula.

Notes: In this lecture and the next our presentation follows that of Straubing [1].

References

[1] Howard Straubing: *Finite Automata, Formal Logic and Circuit Complexity*, Birkhäuser, 1994.