Lecture 3: MSO to Regular Languages

To describe the translation from MSO formulas to regular languages one has to be a bit more formal! All the examples we used in the previous class were *sentences* i.e., every variable that occured in the formula occured within the scope of a quantifier. (A variable that is tied to quantifier is called a *bound* variable. Every variable in a sentence is a bound variable.) Given a sentence ϕ, any word w either satisfies ϕ or does not.

However, in order to reason about sentences, one has to reason about subformulas of sentences and these need not be sentences. As a matter of fact, subformulas of sentences are usually NOT sentences.

For example, consider formulas $\mathrm{First}(x)$ and $y = x + 1$ that were used repeatedly in the previous lecture. The former has x as a *free variable* while the latter has x and y as free variables. A free variable is one that is not "captured" by a quantifier. It does not make sense to ask if w satisfies $\mathrm{First}(x)$. Instead, one has to give a word w and a position i in the word w and then one may ask if $\mathrm{First}(i)$ is true. Similarly to evaluate $y = x + 1$, one needs values (i.e. positions) for the variables x and y before we can verify its truth.

For the moment let us restrict overselves to first order formulas. Then, to meaningfully discuss the truth or falsity of a formula with k free variables, we need a word along with assignment of positions to the k variables. For example, the formula $\phi = (x < y) \wedge a(x) \wedge b(y)$ is true of *bacabc* with x assigned position 2 and y assigned position 5. We represent such a word with assignments for x and y as a word decorated with the variables x and y as follows:

$$\begin{array}{cccccc} b & a & c & a & b & c \\ & x & & & y & \end{array}$$

On the other hand the formula ϕ is not true of *bacabc* if x and y are assigned position 5.

$$\begin{array}{cccccc} b & a & c & a & b & c \\ & & & & x & \\ & & & & y & \end{array}$$

Notice that these decorated words can themselves be thought of as words over the alphabet $\Sigma \times 2^V$ where V is the set of (free) variables. For instance, the two decorated models correspond to the words $(b, \emptyset)(a, \{x\})(c, \emptyset)(a, \emptyset)(b, \{y\})(c, \emptyset)$ and $(b, \emptyset)(a, \emptyset)(c, \emptyset)(a, \emptyset)(b, \{x, y\})(c, \emptyset)$ respectively. Often we shall a for $(a, \emptyset)$ and write $b(a, \{x\})ca(b, \{y\})c$ and $baca(b, \{x, y\})c$ instead.

For the purposes of this lecture let us fix the basic alphabet to be Σ. Following Straubing [2], given a set V of variables we define the set of V-words to be words over the alphabet $\Sigma \times 2^V$ to be those that describe a word over Σ and indicate the positions of all the variables V in the word. Formally, a V-word is a word $(a_1, U_1)(a_2, U_2) \ldots (a_k, U_k)$ where

1. $U_i \cap U_j = \emptyset$ for $i \neq j$.
2. $\bigcup_{1 \leq i \leq k} U_i = V$.

Thus, a V-word associates a unique position of the underlying word, over the alphabet Σ, with each variable in V.

Given a formula ϕ with all of its free variables ($\mathrm{free}(\phi)$) coming from V and a V-word w we can define whether w satisfies ϕ in the obvious way (A formal definition is given in the appendix). Thus, if $\mathrm{free}(\phi) \subseteq V$ then ϕ defines a language of V-words. (In particular, when $\mathrm{free}(\phi)$ is empty, the ϕ defines a language over the set of words over Σ, the set of all words that satisfy ϕ.)

How do we extend this to MSO formulas? Notice that in order to evaluate a formula with a free second order variable X we need to associate a set of positions with X. We could use the decorating technique and simply decorate each position that belongs to the set associated with X by X. Of course, there might be no positions decorated with X, indicating that X is the empty set.

For instance, consider the formula $a(x) \wedge (x \in X) \wedge (y \in X)$. The following decorated word, where x is assigned position 2, y is assigned position 5 and X is assigned the set $\{2, 3, 5\}$ of positions, satisfies this formula.

$$\begin{array}{cccccc} b & a & c & a & b & c \\ & x & & & y & \\ & X & X & & X & \end{array}$$

On the other hand,

$$\begin{array}{cccccc} b & a & c & a & b & c \\ & x & & & y & \\ & & X & & X & \end{array}$$

does not.

Extending the idea used for FO, we define (V, W)-words to be words over the alphabet $\Sigma \times 2^V \times 2^W$ that describe positions for the variables in V and sets of positions for the variables in W. Formally, a word $(a_1, U_1, W_1)(a_2, U_2, W_2) \ldots (a_k, U_k, W_k)$ is a (V, W)-word if it satisfies:

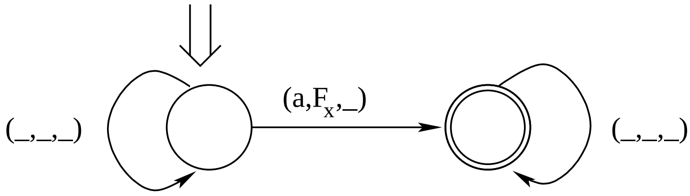
1. $U_i \cap U_j = \emptyset$ for $i \neq j$.
2. $\bigcup_{1 \leq i \leq k} U_i = V$.

Thus, given a formula ϕ whose set of first order free variables $\mathrm{free}_1(\phi)$ is contained in V and whose set of second order free variables $\mathrm{free}_2(\phi)$ is contained in W, and a (V, W)-word w we can define, in the obvious way, whether w satisfies ϕ (a formal definition is given in the appendix). Thus, such a formula, defines a language of (V, W)-words.

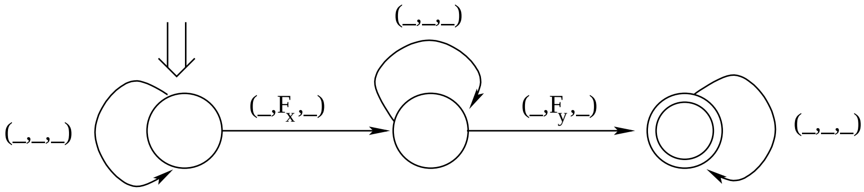
We shall show, by induction on the structure of the formula ϕ that the language (V, W)-words defined by ϕ for any V, W with $\mathrm{free}_1(\phi) \subseteq V$ and $\mathrm{free}_2(\phi) \subseteq W$ is a regular language over $\Sigma \times 2^V \times 2^W$.

It is quite easy to write down a finite automaton that accepts the language of all (V, W)-words. Since regular languages are closed under intersection, in what follows we will assume that only valid (V, W)-words are considered as valid input.

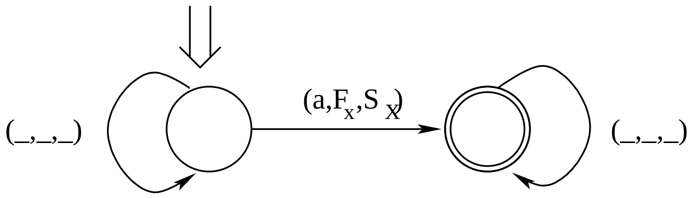
For the basis, we consider the atomic formulas. There are three choices $a(x)$, $x < y$ and $x \in X$. Here is an automaton that accepts (V, W)-words that satisfy $a(x)$.



where a _ stands for "any" and F_x is any subset of V that contains x. Similarly, an automaton that accepts $x < y$ is the following:



where F_y is any subset of V that contains y. Finally, here is an automaton that accepts any (V, W)-word that satisfies $x \in X$,



where S_X is any subset of W that contains X. Note, that the correctness of these three automata relies on the fact that the input consists only of (V, W)-words, but we can always ensure this by taking the product of this automaton with any finite automaton accepting the set of (V, W)-words.

For the induction step, we need to consider the various choices of logical operators. If $\phi = \alpha \wedge \beta$ then, by induction hypothesis we have automata for the languages accepted by α and β and we know that finite automata are closed under language intersection. Similarly if $\phi = \neg\alpha$, we can complement the automaton recognising the set of (V, W)-words satisfying α (and intersect it with the set of valid (V, W)-words). The other operators like $\vee$ and $\Rightarrow$ can be expressed using $\wedge$ and $\neg$. Thus we are left with the quantifiers. Note that $\forall x.\phi(x)$ is $\neg(\exists x.\neg\phi(x))$ and thus it suffices to consider the first order and second order existential quantifiers.

Suppose $\mathrm{free}_1(\exists x.\phi) \subseteq V$ and $\mathrm{free}_2(\exists x.\phi) \subseteq W$. Then, $\mathrm{free}_1(\phi) \subseteq V \cup \{x\}$ and $\mathrm{free}_2(\phi) \subseteq W$. Therefore, by the induction hypothesis, the set of $(V \cup \{x\}, W)$-words that satisfy ϕ is a regular language. Further, it is quite easy to see that if

$$\begin{array}{llllll} a_1 & a_2 & \ldots & a_i & \ldots & a_k \\ F_1 & F_2 & \ldots & F_i \cup \{x\} & \ldots & F_k \\ S_1 & S_2 & \ldots & S_i & \ldots & S_k \end{array}$$

satisfies ϕ then

$$\begin{array}{cccccc} a_1 & a_2 & \dots & a_i & \dots & a_k \\ F_1 & F_2 & \dots & F_i & \dots & F_k \\ S_1 & S_2 & \dots & S_i & \dots & S_k \end{array}$$

satisfied $\exists x.\phi$ (simply choose x to be the position i). Conversely, when

$$\begin{array}{cccc} a_1 & a_2 & \dots & a_k \\ F_1 & F_2 & \dots & F_k \\ S_1 & S_2 & \dots & S_k \end{array}$$

satisfies $\exists x.\phi$ then, then there is a choice of position, say i, for x such that ϕ is satisfied w.r.t. to this assignment and thus,

$$\begin{array}{cccccc} a_1 & a_2 & \dots & a_i & \dots & a_k \\ F_1 & F_2 & \dots & F_i \cup \{x\} & \dots & F_k \\ S_1 & S_2 & \dots & S_i & \dots & S_k \end{array}$$

satisfies ϕ. Thus, set of (V, W)-words that satisfy $\exists x.\phi$ are just the images of the set of $(V \cup \{x\}, W)$-words that satisfy ϕ under the homomorphism h defined by $h((a, F, S)) = (a, F \setminus \{x\}, S)$. Since, regular languages are closed under homomorphic images, we conclude that the set of (V, W)-words satisfying $\exists x.\phi$ is a regular language.

The proof in case of $\exists X.\phi$ is almost identical. In this case, the set of (V, W)-words that satisfy $\exists X.\phi$ are just the images of the set of $(V, W \cup \{X\})$-words that satisfy ϕ under the homomorphism h defined by $h((a, F, S)) = (a, F, S \setminus \{x\})$. Thus the set of words satisfying $\exists X.\phi$ forms a regular language.

Thus we have proved that whenever $\text{free}_1(\phi) \subseteq V$ and $\text{free}_2(\phi) \subseteq W$, the set of (V, W)-words that satisfy ϕ is a regular language over $\Sigma \times 2^V \times 2^W$. Thus, if ϕ is a sentence then the set of $(\emptyset, \emptyset)$-words that satisfy ϕ is a regular language over $\Sigma \times 2^\emptyset \times 2^\emptyset$ (and this is the same as the language over Σ, via the bijection that sends $(a, \emptyset, \emptyset)$ to a).

Thus, we have established both directions of Büchi's theorem.

1 Stratification of FO formulas

We now turn our attention to showing that the language of words with even number of as is not definable in the first-order logic of words. We define the quantifier depth of a f.o. formula ϕ as follows: if ϕ is quantifier-free than $\mathsf{qd}(\phi) = 0$. Otherwise, $\mathsf{qd}(\phi \wedge \phi') = \mathsf{maximum}(\mathsf{qd}(\phi, \phi')$, $\mathsf{qd}(\neg\phi) = \mathsf{qd}(\phi)$ and $\mathsf{qd}(\exists x.\phi) = \mathsf{qd}(\phi) + 1$.

Now, if we fix a finite set F of variables, there are only finitely many quantifier-free formulas over F upto logical equivalence. Simply rewrite the formula into its equivalent disjunctive normal form (i.e. a formula of the form $P_1 \vee P_2 \vee \dots P_k$ where each $P_i = A_1 \wedge A_2 \wedge \dots A_{k_i}$ is a conjunction of literals (i.e. each A_i is either a atomic formula or the negation of an atomic formula) and use the fact that $\phi \wedge \phi = \phi$ and $\phi \vee \phi = \phi$.

Next, observe that $\mathsf{qd}(i+1)$ formulas are just boolean combinations of formulas with quantifier depth $<= i$ and formulas of the form $\exists x.\phi$ where $\mathsf{qd}(\phi) \leq i$. Thus, if the number of formulas (upto logical equivalence) of quantifier depth i or less is finite then the number of formulas with quantifier depth $i+1$ or less is also finite.

The previous two paragraphs (when formalise appropriately!) yields the following theorem.

Theorem 1 *For any i there are only finitely many formulas of quantifier depth i or less (upto logical equivalence).*

Thus, we can stratify first order definable languages via the quantifier depth necessary to define a language. One method to show that a particular to language is not first order definable is to show that for each k, no sentence of quantifier depth k can define the language. This is the route we shall take in order to show that evenness is not first order definable.

This leads us to the natural question: How do we show that a language is not definable via sentences of quantifier depth k? Well, this is done by finding two words w and w', one in the language and another outside the language and show that these cannot be *distinguished* by sentences of quantifier depth k or less. That is, we show that for each sentence ϕ of quantifier depth k or less, either both w and w' satisfy ϕ or neither satisfies ϕ.

Thus we move on to the following question: Given two words w and w' how do we decide whether there is a sentence of quantifer depth k (or less) that distinguishes w from w'? It is here that *Ehrenfeucht-Fraisse games* play their role. Given w,w' and k we set up a game between two players 0 and 1 (the *cynic* and the *believer*) such that w is distinguishable from w' by some sentence of quantifier depth k or less if and only the player 0 has a winning strategy in the game.

1.1 Ehrenfeucht-Fraisse Games

Let w and w' be two V-words and let k be some positive integer. There are k-rounds in the game. In each round, say round i, player 0 (who is trying to show that these two words are distinguishable) picks one of the two words and a position in that word and labels it with a new variable x_i. Player 1 must then pick the other word (the one not picked by player 0), and label one of its positions with x_i. Thus, at the end of k rounds we have two $V \cup \{x_1, \ldots x_k\}$-words. Player 0 wins the game if there is some quantifier-free formula (over $V \cup \{x_1, \ldots, x_k\}$) that distinguishes these two words. Otherwise player 1 wins the game. Notice that this forces player 1 to try and duplicate player 0's moves as closely as possible so that the labellings are indistinguishable via atomic propositions.

We say that two V-words w and w' are *k-equivalent* if player 1 has a winning strategy to win the k round game on w and w'. We write $w \equiv_k w'$ to indicate this. On the logical side, we say that two V-words w and w' are *k-indistinguishable* if no quantifier depth k formula with free variables in the set V can distinguish between these two words. We write $w \sim_k w'$ to indicate this.

Here is a 2 round game played on the words $w = abbabbab$ and $w' = ababbabb$. Player 0 picks w' and labels position 7 with x_1.

$$\begin{array}{cccccccc} \mathrm{a} & \mathrm{b} & \mathrm{b} & \mathrm{a} & \mathrm{b} & \mathrm{b} & \mathrm{a} & \mathrm{b} \\ & & & & & & & x_1 \end{array}$$

$$\begin{array}{cccccccc} \mathrm{a} & \mathrm{b} & \mathrm{a} & \mathrm{b} & \mathrm{b} & \mathrm{a} & \mathrm{b} & \mathrm{b} \\ & & & & & & x_1 & x_2 \end{array}$$

Now, player 1 must pick some position, with a b, and to represent the "equivalent" in w of position 7 in w'. But this is doomed to fail.

If player 1 picks position 8 then in the second round player 0 would pick position 8 in w' and this leaves us at the following situation: Now, no matter where player 1 places x_2 it would violate atomic formula $x_1 < x_2$ satisfied by w'.

On the other hand, if player 1 picks any position other than 8, then player 0 would pick w in the second round and label position 7 with x_2. Here is the result (where player 1 played position 6 in the first round):

$$\begin{array}{cccccccc} \mathrm{a} & \mathrm{b} & \mathrm{b} & \mathrm{a} & \mathrm{b} & \mathrm{b} & \mathrm{a} & \mathrm{b} \\ & & & & & x_1 & x_2 & \end{array}$$

$$\begin{array}{cccccccc} \mathrm{a} & \mathrm{b} & \mathrm{a} & \mathrm{b} & \mathrm{b} & \mathrm{a} & \mathrm{b} & \mathrm{b} \\ & & & & & & x_1 & \end{array}$$

Once again, no matter where player 1 places x_2 it would violate the formula $a(x_2) \wedge (x_1 < x_2)$ satisfied by w. Here is a formlula of quantifier depth 2 that distinguishes these two words: $\exists x_1.\ b(x_1) \wedge (\exists x_2.\ x_1 < x_2) \wedge \forall x_2.(x_2 > x_1) \Rightarrow \neg a(x_2)$. The word w' satisfies this formula with x_1 instantiated as position 7. Further note that $\exists x_2.\ x_1 < x_2$ translates the strategy against player 1 playing position 8 in round 1 and $\forall x_2.(x_2 > x_1) \Rightarrow \neg a(x_2)$ translates the strategy against player 1 playing any other position in round 1. This ability to translate a k round winning strategy to a distinguishing formula of quantfier depth k is not a coincidence.

Lemma 2 *Let w and w' be two V-words such that player 0 has a winning strategy in the k round game. Then, there is a formula ϕ (with free variables in V) with quantifier depth bounded by k that is satisfied by w and not by w'.*

Proof: Let $w = (a_1, V_1)(a_2, V_2) \ldots (a_m, V_m)$ and $w' = (a'_1, V'_1)(a'_2, V'_2) \ldots (a'_n, V'_n)$. The proof proceeds by induction on k. If $k = 0$ then, by definition there is a quantifier-free formula that distinguishes w and w' and this serves as the base case. Suppose the results holds if the number of rounds is less than k.

Now, consider the winning strategy for player 0 that wins the k round game. Suppose this move picks the position i in word w and labels it with variable x. Therefore, any position j in w' as the choice for player 1's move is a loosing move (i.e. player 0 can continue the game so as to win it.) This is equivalent to saying that player 0 has a winning strategy in the $k-1$ round game played on the words $u = (a_1, V_1) \ldots (a_i, V_i \cup \{x\}) \ldots (a_m, V_m)$ and $u'_j = (a'_1, V'_1) \ldots (a'_j, V'_j \cup \{x\}) \ldots (a'_n, V'_n)$ for each j. Thus, by the induction hypothesis there is a formula ϕ_j, with quantifier depth bounded by $k-1$, such that $u \models \phi_j$ and $u'_j \not\models \phi_j$.

Thus, $w \models \exists x. \bigwedge_{1 \leq j \leq n} \phi_j$ (Simply set x to be i). On the other hand $w' \not\models \exists x. \bigwedge_{1 \leq j \leq n} \phi_j$. Thus, we have constructed a formula of quantifier depth bounded by k that is satisfied by w and not by w'. ∎

Notes: Our presentation has followed the notation used in Straubing [2]. Another book that presents these results is Pippinger [1].

References

[1] Nick Pippenger: *Theories of Computability*, Cambridge University Press, 1997.

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

Appendix:

Given a word $w = a_1 a_2 \ldots a_n$ a (V, W)-valuation over w is a function σ that maps V to $\{1, 2, \ldots n\}$ and W to $2^{\{1,2,\ldots,n\}}$. Given a word $w = a_1 a_2 \ldots a_n$ and a (V, W)-valuation σ with $\mathrm{free}_1(\phi) \subseteq V$ and $\mathrm{free}_2(\phi) \subseteq W$, we define when $(a_1 a_2 \ldots a_n, \sigma)$ satisfies a formula ϕ, written $a_1 a_2 \ldots a_n, \sigma \models \phi$, as follows:

$$
\begin{array}{lcll}
a_1 a_2 \ldots a_n, \sigma & \models & a(x) & \text{if } a_{\sigma(x)} = a \\
a_1 a_2 \ldots a_n, \sigma & \models & x < y & \text{if } \sigma(x) < \sigma(y) \\
a_1 a_2 \ldots a_n, \sigma & \models & x \in X & \text{if } \sigma(x) \in \sigma(X) \\
a_1 a_2 \ldots a_n \sigma & \models & \phi \wedge \phi' & \text{if } (a_1 a_2 \ldots a_n, \sigma \models \phi) \text{ and } (a_1 a_2 \ldots a_n, \sigma \models \phi') \\
a_1 a_2 \ldots a_n, \sigma & \models & \neg \phi & \text{if } (a_1 a_2 \ldots a_n, \sigma \not\models \phi) \\
a_1 a_2 \ldots a_n, \sigma & \models & \exists x. \phi & \text{if there is an } i \in \{1, 2, 3, \ldots n\} \text{ such } a_1 a_2 \ldots a_n, \sigma[x : i] \models \phi \\
a_1 a_2 \ldots a_n, \sigma & \models & \exists X. \phi & \text{if there is } S \subseteq \{1, 2, 3, \ldots n\} \text{ such } a_1 a_2 \ldots a_n, \sigma[X : S] \models \phi
\end{array}
$$

where $\sigma[v : y](u) = \sigma(u)$ if $u \neq v$ and $\sigma[v : y](v) = y$.

Given a (V, W)-word $(a_1, F_1, S_1)(a_2, F_2, S_2) \ldots (a_n, F_n, S_n)$ we can construct a word–valuation pair (w, σ) by setting $w = a_1 a_2 \ldots a_n$ and $\sigma(x) = i$ if $x \in F_i$ for any $x \in V$ and $\sigma(X) = \{i \mid X \in S_i\}$ for any $X \in W$. It is easy to check that this is a bijective correspondance between (V, W)-words and word-valuation pairs. We say that a (V, W)-word satisfies a formula ϕ if and only if the corresponding word–valuation pair satisfies the formula ϕ.