Lecture 2a: First-order Logic over words

Here is a formula describing the language of words over $\{a, b, c\}$ that contain ab as a subword:

$$\exists x.\ \exists y.\ (a(x) \wedge b(y) \wedge (x < y) \wedge \forall z.\ (x < z) \Rightarrow (y = z) \vee (y < z))$$

However, if the alphabet is $\{a, b\}$ then this can expressed for the following simpler formula:

$$\exists x.\ \exists y.\ a(x) \wedge b(y) \wedge (x < y)$$

The first formula uses 3 variables (x, y and z) while the second one uses only 2 and one of things that we hope to prove in this course is that if the alphabet has at least 3 letters then the above language cannot be expressed with FO($<$) formulas that use at most 2 variables.

Exercise: Write down a FO($<$) formula for the language $(ab)^*$.

Formal Semantics for FO($<$)

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 First(x) and $y = x + 1$ that were used in Lecture 2. 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 First(x). Instead, one has to give a word w and a position i in the word w and then one may ask if 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.

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

Thus, to evaluate the formula φ whose set of free variables is contained in a set V, we need a word $w = a_1 a_2 \ldots a_n$ along with a *valuation* σ which assigns to each variable in V a position from the set $\{1, 2, \ldots, n\}$. We write $w, \sigma \models \varphi$, to denote that the formula φ evaluates to true w.r.t. to the pair consisting of the word w and the valuation σ and this relation $\models$ is defined inductively as follows:

$$
\begin{array}{lcll}
a_1a_2\ldots a_n, \sigma & \models & a(x) & \text{if } a_{\sigma(x)} = a \\
a_1a_2\ldots a_n, \sigma & \models & x < y & \text{if } \sigma(x) < \sigma(y) \\
a_1a_2\ldots a_n\sigma & \models & \phi \wedge \phi' & \text{if } (a_1a_2\ldots a_n, \sigma \models \phi) \text{ and } (a_1a_2\ldots a_n, \sigma \models \phi') \\
a_1a_2\ldots a_n, \sigma & \models & \neg\phi & \text{if } (a_1a_2\ldots a_n, \sigma \not\models \phi) \\
a_1a_2\ldots a_n, \sigma & \models & \exists x.\phi & \text{if there is an } i \in \{1, 2, 3, \ldots n\} \text{ such } a_1a_2\ldots a_n, \sigma[x:i] \models \phi
\end{array}
$$

where $\sigma[v:y](u) = \sigma(u)$ if $u \neq v$ and $\sigma[v:y](v) = y$. Thus with this notation,

$$bacabc, [x \mapsto 2, y \mapsto 5] \models (x < y) \wedge a(x) \wedge b(y)$$

and

$$bacabc, [x \mapsto 5, y \mapsto 5] \not\models (x < y) \wedge a(x) \wedge b(y)$$

From the definition above, it is clear that if w is a word, φ is a formula and σ and σ' are valuations such that for all variables x that occur as free variables in φ we have $\sigma(x) = \sigma'(x)$ then $w, \sigma \models \varphi$ iff $w, \sigma' \models \varphi$. In particular, if the formula is a sentence, then the valuation is irrelevant and we may instead write $w \models \varphi$. The language defined by a sentence φ is the set $\{w \mid w \models \varphi\}$.

First-order Logic with 1 variable:

As a warm up to understanding FO over words, let us consider the simple fragment consisting all the formulas which use only one variable, say x. We write $\mathrm{FO}^1(<)$ to refer to this fragment.

Let $A \subseteq \Sigma$. Then the formula $\forall x. \bigvee_{a \in A} a(x)$ describes the language A^*. Further, since $\mathrm{FO}^1(<)$ has the logical operators $\neg$ and $\vee$, the class of languages definable using formulas in this logic is closed under boolean operations. Thus, every language that is a boolean combination of languages of the form A^* is definable in $\mathrm{FO}^1(<)$. What about the converse?

First of all we observe that in $\mathrm{FO}^1(<)$ the operators $=$ and $<$ are useless (i.e. they can be eliminated). The only formulas that may use these operators are $x < x$ which is always false and $x = x$ is always true. Thus we may assume that the atomic formulas are only of the form $a(x)$, $a \in \Sigma$. We will now show that if $u \equiv_\alpha v$ then for any sentence in $\varphi \in \mathrm{FO}^1(<)$, $u \models \varphi$ if and only if $v \models \varphi$. The obvious way to prove this is by induction on the size of the formula, but alas that will lead us to consider formulas with free variables and hence we have to strengthen the hypothesis of our inductive argument. The details are as follows:

Lemma 1 *Let φ be any formula in $FO^1(<)$ and let u and v be words with $u \equiv_\alpha v$. Then, if i and j are any two positions in u and v respectively, such that $u_i = v_j$, then $u, [x \mapsto i] \models \varphi$ if and only if $v, [x \mapsto j] \models \varphi$.*

Proof: The proof is by induction on the structure of the formula.

Case 1: $\varphi = a(x)$

$u, [x \mapsto i] \models a(x)$ iff $u_i = a$ iff $v_j = a$ iff $v, [x \mapsto j] \models a(x)$.

Case 2: $\varphi = \neg\psi$

$u, [x \mapsto i] \models \neg\psi$ iff $u, [x \mapsto i] \not\models \psi$ iff $v, [x \mapsto j] \not\models \psi$ iff $v, [x \mapsto j] \models \neg\psi$

Case 3: $\varphi = \varphi_1 \vee \varphi_2$

$u, [x \mapsto i] \models \varphi_1 \vee \varphi_2$ iff $u, [x \mapsto i] \models \varphi_1$ or $u, [x \mapsto i] \models \varphi_2$ and by induction hypothesis, $u, [x \mapsto i] \models \varphi_k$ iff $v, [x \mapsto j] \models \varphi_k$ for $k \in 1, 2$. The result follows.

Case 4: $\varphi = \exists x.\ \psi$

If $u, [x \mapsto i] \models \exists x.\ \psi$ then there is a position i' in u such that $u, [x \mapsto i'] \models \psi$. Further since $u \equiv_\alpha v$, there is a position j' in v such that $u_{i'} = v_{j'}$. Then, by induction hypothesis, $v, [x \mapsto j'] \models \psi$. Therefore, $v, [x \mapsto j] \models \exists x.\ \psi$. The converse is proved symmetrically and this completes the proof of this lemma. ■

As a consequence of the above lemma we have that if $u \equiv_\alpha v$ and φ is any formula in $\mathrm{FO}^1(<)$ then $u \models \varphi$ iff $v \models \varphi$.

But any equivalence class of $\equiv_a$ *lpha* is a boolean combination of languages of the form A^* (see Lecture 02a). Thus we have the following theorem:

Theorem 2 *Let Σ be any alphabet. For any language L over Σ the following are equivalent:*

1. *L is a boolean combination of languages of the form A^*, $A \subseteq \Sigma$.*
2. *L is recognised by a commutative, idempotent monoid.*
3. *L is definable in $FO^1(<)$.*

The equivalence of the first two is proved above and the equivalence of the last two can be found in Lecture 02a. As a consequence of these equivalences, checking if a given regular language is definable in $\mathrm{FO}^1(<)$, reduces to computing its syntactic monoid and checking that it is commutative and idempotent.

References

[1] V. Diekart, P. Gastin and M. Kufleitner: A Survey on Small Fragments of First-Order Logic over Finite Words, Int. J. Found. Comput. Sci., Vol 19, 2008.