Second Order Logic

Mahesh Viswanathan

Fall 2018

Second order logic is an extension of first order logic that reasons about predicates. Recall that one of the main features of first order logic over propositional logic, was the ability to quantify over elements that are in the universe of the structure. Second order logic not only allows one quantify over elements of the universe, but in addition, also allows quantifying relations over the universe.

1 Syntax and Semantics

Like first order logic, second order logic is defined over a vocabulary or signature. The signature in this context is the same. Thus, a signature is $\tau = (\mathcal{C}, \mathcal{R})$, where $\mathcal{C}$ is a set of constant symbols, and $\mathcal{R}$ is a collection of relation symbols with a specified arity. Formulas in second order logic will be over the same collection of symbols as first order logic, except that we can, in addition, use relational variables. Thus, a formula in second order logic is a sequence of symbols where each symbol is one of the following.

1. The symbol $\perp$ called *false* and the symbol $=$;
2. An element of the infinite set $\mathcal{V}_1 = \{x_1, x_2, x_3, \ldots\}$ of *variables*;
3. An element of the infinite set $\mathcal{V}_2 = \{X_1^1, x_2^1, \ldots X_j^k, \ldots\}$ of *relational variables*, where the superscript indicates the arity of the variable;
4. Constant symbols and relation symbols in τ;
5. The symbol $\rightarrow$ called *implication*;
6. The symbol $\forall$ called the *universal quantifier*;
7. The symbols (and) called *parenthesis*.

As always, not all such sequences are formulas; only *well formed* sequences are formulas in the logic. This is defined as follows.

Definition 1. A *term* t over signature τ is either a (first order) variable or a constant symbol in τ.
A *well formed formula (wff)* over signature τ is inductively defined as follows.

1. $\perp$ is a wff.
2. If t_1, t_2 are terms then $t_1 = t_2$ is a wff.
3. If t_i is a term for $1 \leq i \leq k$ and R is a k-ary relation symbol in τ then $Rt_1t_2\cdots t_k$ is a wff.
4. If t_i is a term for $1 \leq i \leq k$ and X^k is a k-ary relational variable then $X^kt_1t_2\cdots t_k$ is a wff.
5. If φ and ψ are wffs then $(\varphi \rightarrow \psi)$ is a wff.
6. If φ is a wff and x is a variable then $(\forall x\varphi)$ is a wff.

7. If φ is a wff and X^k is a relational variable then $(\forall X^k \varphi)$ is a wff.

As in the case of first order logic, it is convenient to introduce other standard logical operators that can be defined syntactically defined in terms of the formulas given in Definition 1. In addition, to boolean connectives, and first order existential quantification (which are defined exactly as for first order logic), we can also define existential second-order quantification as

$$\exists X^k \varphi = \neg(\forall X^k \neg\varphi).$$

Again, in order to avoid writing to many parenthesis, we will assume the following precedence of operators (from increasing to decreasing): $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\forall$ (both first-order and second-order).

Example 2. Let us look at some example formulas in second order logic. Let us start by looking at the vocabulary of graphs, which is $\tau_G = \{E\}$ and consists of a single binary relation which denotes the edge relation in the graph. We could write a formula that expresses the fact that the edge relation E encodes a graph that is 3-colorable as follows.

$$\begin{aligned}\exists X_1^1 \exists X_2^1 \exists X_3^1 (&\forall x (X_1^1 x \vee X_2^1 x \vee X_3^1 x) \wedge \\ &\forall x \forall y (Exy \rightarrow (\neg(X_1^1 x \wedge X_1^1 y) \wedge \neg(X_2^1 x \wedge X_2^1 y) \wedge \neg(X_3^1 x \wedge X_3^1 y))))\end{aligned}$$

The formula says that there are sets X_1^1, X_2^1, and X_3^1 (i.e., unary relations) that correspond to vertices colored 1, 2, and 3, respectively, such that every vertex gets one of the 3 colors, and adjacent vertices get different colors.

Consider writing a formula to express the fact that the graph has a path from s to t. One way to express this is by saying that s and t belong to the transitive closure of E. But how do we denote the transitive closure of E? Well, it turns out that any binary relation that contains E, is reflexive and "closed" with respect to composition with E, contains the transitive closure! Therefore, we express the existence of a path from s to t as

$$\forall P^2 (\mathsf{Closed}(P^2) \rightarrow P^2 st)$$

where $\mathsf{Closed}(P^2)$ is the formula $\forall x \forall y (((((x = y) \vee Exy) \rightarrow P^2 xy) \wedge ((Exy \wedge P^2 yz) \rightarrow P^2 xz))$.

Let us now consider the signature of orders $\tau_O = \{\leq\}$ that has one binary relation namely, the ordering $\leq$. Let us write the sentence "every bounded, non-empty set has a least upper bound"; the formula below is direct translation of each of the adjectives in this property.

$$\forall X^1 (((\exists x X^1 x) \wedge (\exists y \forall x (X^1 x \rightarrow x \leq y))) \rightarrow (\exists z \forall y (\forall x (X^1 x \rightarrow x \leq y)) \leftrightarrow (z \leq y)))$$

The semantics of formulas in second-order logic, like in the case of first order logic, are defined in a structure. Recall that a structure is a universe along with an interpretation of the constant symbols and relation symbols in the signature. To define whether a formula holds in a structure, we also need an assignment, which interprets the free variables. For first order logic, an assignment was simply a mapping of variables to elements in the universe of the structure. For second order logic, however, the presence of relational variables, means that an assignment must also give an interpretation of these variables as relations (of the appropriate arity) over the structure. This is formally defined as follows.

Definition 3. For a τ-structure $\mathcal{A}$, an *assignment* α over $\mathcal{A}$ is a pair of functions (α_1, α_2) where $\alpha_1 : \mathcal{V}_1 \rightarrow u(\mathcal{A})$ and $\alpha_2 : \mathcal{V}_2 \rightarrow \cup_k [u(\mathcal{A})]^k$ that assigns to every k-ary relational variable X^k a relation $\alpha_2(X^k) \subseteq [u(\mathcal{A})]^k$. If t is a constant symbol c, we will take $\alpha(t)$ to be $c^{\mathcal{A}}$.

For an assignment $\alpha = (\alpha_1, \alpha_2)$ over $\mathcal{A}$, $\alpha[x \mapsto a]$ is the assignment where α_1 is changed as follows.

$$\alpha_i[x \mapsto a](y) = \begin{cases} \alpha_i(y) & \text{for } y \neq x \\ a & \text{when } x = y \end{cases}$$

We are now ready to define the semantics of a (second order) formula in a structure under an assignment. The definition is similar to the one for first order logic, and is also defined inductively on the structure of the formula.

Definition 4. The relation $\mathcal{A} \models \varphi[\alpha]$ is inductively defined as follows.

- $\mathcal{A} \not\models \bot[\alpha]$ for all $\mathcal{A}$ and α
- $\mathcal{A} \models t_1 = t_2[\alpha]$ iff $\alpha_1(t_1) = \alpha_1(t_2)$
- $\mathcal{A} \models Rt_1 \cdots t_n[\alpha]$ iff $(\alpha_1(t_1), \alpha_1(t_2), \ldots \alpha_1(t_n)) \in R^{\mathcal{A}}$
- $\mathcal{A} \models X^k t_1 \cdots t_n[\alpha]$ iff $(\alpha_1(t_1), \alpha_1(t_2), \ldots \alpha_1(t_n)) \in \alpha_2(X)$
- $\mathcal{A} \models (\varphi \to \psi)[\alpha]$ iff $\mathcal{A} \not\models \varphi[\alpha]$ or $\mathcal{A} \models \psi[\alpha]$
- $\mathcal{A} \models (\forall x \varphi)[\alpha]$ iff for every $a \in u(\mathcal{A})$, $\mathcal{A} \models \varphi[\alpha[x \mapsto a]]$
- $\mathcal{A} \models (\forall X^k \varphi)[\alpha]$ iff for every $S \in [u(\mathcal{A})]^k$, $\mathcal{A} \models \varphi[\alpha[X \mapsto S]]$

As in first order logic, Definition 4 suggests that the assignment of values to variables (first order and relational) only matters for the "free" variables, and not those that are quantified. We extend the definition of free and bound occurences to relational variables as well.

Definition 5. In wffs $\forall x \psi$ and $\forall X^k \psi'$, ψ and ψ' is said to be in the scope of the quantifiers $\forall x$ and $\forall X^k$, respectively.

Every occurrence of variables x and X^k in $\forall x \psi$ and $\forall X^k \psi$, respectively, are said to be *bound*. Any occurrence of variables (first order or relational) that is not bound is said to be *free*.

As in first order logic, the semantics given in Definition 4 ensures that the satisfaction relation only depends on the values assigned to free variables.

Theorem 6. *For a formula φ and assignments α and β that agree on all the free variables and free relational variables of φ, $\mathcal{A} \models \varphi[\alpha]$ iff $\mathcal{A} \models \varphi[\beta]$.*

Sentences are, as always, formulas with no free (first order) variables or relational variables. Thanks to Theorem 6, this means that for sentences, the assignment does not determine its satisfaction.

Proposition 7. *For a sentence φ, and any two assignments α_1 and α_2, $\mathcal{A} \models \varphi[\alpha_1]$ iff $\mathcal{A} \models \varphi[\alpha_2]$. Therefore, we write $\mathcal{A} \models \varphi$ if for any assignment α, $\mathcal{A} \models \varphi[\alpha]$.*

Satisfiability and validity of second order formulas is defined in the same way as for first order logic — φ is *satisfiable* if there is a structure $\mathcal{A}$ and assignment α such that $\mathcal{A} \models \varphi[\alpha]$; and φ is *valid* if for every structure $\mathcal{A}$ and assignment α, $\mathcal{A} \models \varphi[\alpha]$.

For any logic, the fundamental computational questions are satisfiability and validity. We observed that, in the case of propositional logic, satisfiability is NP-complete while validity if co-NP-complete. In contrast, for first order logic, validity if RE-complete, and (therefore) satisfiabiity is not recursively enumerable. What does the landscape look like for second order logic? Clearly, the satisfiability problem for second order logic is not recursively enumerable — since first order logic formulas are special second order logic formulas, the non-recursive enumerability proof can be extended. But what about validity? Is there a sound and complete proof system that demonstrates the recursive enumerability of the validity problem? It turns out that there is no such proof system. The "incompleteness" of second order logic can be seen as a consequence of Gödel's incompleteness theorem — checking membership in $\mathrm{Th}(\mathbb{N}, 0, 1, +, \cdot)$ can be reduced to the validity problem of second order logic. Here we present a simpler proof that doesn't establish incompleteness, but rather the non-recursive enumerability of validty.

Theorem 8. *The validity problem for second order logic — given a formula φ, determine if φ is valid — is not recursively enumerable.*

Proof. Recall that the satisfiability problem of first order logic is not recursively enumerable; this holds even for formulas over a finite vocabulary. We will reduce this to the validity problem of second order logic, and thus complete the proof. Consider a finite vocabulary τ and a formula φ over τ. Consider the formula

$$\psi = \exists X_R^k \cdots \exists x_c \cdots \varphi[R \mapsto X_R^k, c \mapsto x_c]$$

In ψ, for each relation symbol R of arity k, we have a new relational variable X_R^k that is existentially quantified, and for each constant symbol c, we have anew (first order) variable x_c that is existentially quantified. Within the scope of these quantifiers, we have the formula φ where every relation symbol $R \in \tau$ has been syntactically replaced by its corresponding variable X_R^k, and every constant $c \in \tau$ has been replaced by x_c.

It is easy to see that φ is satisfiable if and only if ψ is valid. This completes the proof. □

2 Monadic Second Order Logic

In addition to first order logic, another important fragment of second order logic is *monadic second order logic* (MSO). While first order logic can be seen as the fragment that does not allow any relational variables, MSO is the fragment where all relational variable are "monadic", i.e., of arity 1. Thus, in MSO one can only quantify over sets, as opposed to more general relations.

Definition 9. *Monadic second order logic* is the fragment of second order logic where all relational variables have arity 1. Since all relational variables are unary, by convention, we will drop the superscript that indicates the variables arity.

Example 10. Consider the signature of graphs $\tau_G = \{E\}$. Observe that sentence characterizing 3-colorable graphs in Example 2 is monadic. The formula given to express the fact the graph has an s-t path is in Example 2 however not monadic; it uses a binary relational variable P^2. However, this property can be expressed in MSO. Instead of saying that s and t are in the transtive closure of the relation E, we will instead say that t belongs to any set that contains s and is closed under taking edges in the graph. Here is the precise definition.

$$\forall R((Rs \wedge \mathsf{Closed}(R)) \rightarrow Rt)$$

where $\mathsf{Closed}(R)$ is the formula $\forall x \forall y((Rx \wedge Exy) \rightarrow Ry)$.

Consider now trying to express the fact that a graph has an independent set of size at leat k. This is equivalent to saying that there is a set of vertices that has at least k distinct elements such that no pair in the set is connected by an edge. This can be written in MSO as follows.

$$\exists I(\forall x \forall y((\neg(x = y) \wedge Ix \wedge Iy) \rightarrow \neg Exy) \wedge \exists x_1 \cdots \exists x_k(\bigwedge_{i \neq j} \neg(x_i = x_j) \wedge \bigwedge_i Xx_i))$$

2.1 Monadic Second Order Logic on Words

We will study MSO on restricted class of structures. In particular, of particular interest is the study of MSO on words. What this means is, for alphabet Σ, the signature is fixed to be $\tau_W = \{<, S, \{Q_a\}_{a \in \Sigma}\}$ where $<, S$ are binary relation symbols, and Q_a is unary relation symbol for every a in alphabet Σ. Further, any structure is of the form, where the universe is the set of positions in the word, and the relations $S, <$, and Q_a are interpreted the way "standard" manner.

For example, let us consider $\Sigma = \{0, 1\}$ and the word $w = 010110$. The w can be represented as a structure as follows. The signature (since $\sigma = \{0, 1\}$) is $\tau = \{<, S, Q_0, Q_1\}$. The structure for w is

$$\begin{aligned}
\mathcal{W} = (\quad & \{1,2,3,4,5,6\},\\
& <^{\mathcal{W}} = \{(1,2),(1,3),\ldots(1,6),(2,3),\ldots(2,6),(3,4),(3,5),(3,6),(4,5),(4,6),(5,6)\},\\
& S^{\mathcal{W}} = \{(1,2),(2,3),(3,4),(4,5),(5,6)\},\\
& Q_0 = \{1,3,6\},\\
& Q_1 = \{2,4,5\}).
\end{aligned}$$

Given the above example, and our intuition behind viewing words as structures, we make the following assumptions about the structure representation $\mathcal{W}$ of a word w.

- W will be taken to be the set $\{1, 2, \ldots n\}$ where n is the length of the string w.
- $<^{\mathcal{W}}, S^{\mathcal{W}}$ will be the standard ordering and successor relations on $\{1, 2, \ldots n\}$.
- We will drop the superscript $\mathcal{W}$ and refer to relations of $\mathcal{W}$ using the relation symbol. Thus, we will abuse notion and refer to $<^{\mathcal{W}}$ as $<$.

Notice these assumptions about our conventions are not restrictive. Further, there is a 1-to-1, onto mapping from words to the structures that represent them. Thus, we will interchangably use "word" to mean both the string and its structure representation.

Let us look at some example properties about words that can be expressed in MSO.

Example 11. Here are some simple example properties expressed in MSO.

- "x is the first position in the string."
$$\mathsf{first}(x) = \neg\exists y Syx$$
- "x is the last position in the string."
$$\mathsf{last}(x) = \neg\exists y Sxy$$
- The relation $<$ can be defined in terms of S; thus, for MSO over words, we may assume that the signature does not contain $<$.
$$\neg(x = y) \wedge \forall X((Xx \wedge \forall y_1 \forall y_2((Xy_1 \wedge Sy_1y_2) \rightarrow Xy_2)) \rightarrow Xy)$$
- "The length of the string is even."
$$\exists X((\forall x \forall y Sxy \rightarrow (Xx \leftrightarrow \neg Xy)) \wedge (\exists u \exists v(\mathsf{first}(u) \wedge Xu \wedge \mathsf{last}(v) \wedge \neg Xv)))$$

Example 12. Assume that words encode executions of a program with two threads. Label c_i denotes thread i is in the critical section, and w_i denotes that thread i is waiting to enter the critical section.

- *Mutual Exclusion:* Both threads are not in the critical section at the same time.
$$\forall x \neg(Q_{c_1}x \wedge Q_{c_2}x)$$
- *Starvation Freedom:* If thread 1 is waiting then it eventually enters the critical section.
$$\forall x(Q_{w_1}x \rightarrow \exists y((x < y) \wedge Q_{c_1}y))$$
- *Strong Starvation Freedom:* If thread 1 is waiting then it enters the critical section before thread 2 does for a second time.
$$\begin{aligned}\forall x(Q_{w_1}x \rightarrow (\exists y((x < y) \wedge Q_{c_1}y \wedge \\ \neg(\exists x_1 \exists x_2 \exists x_3((x \leq x_1) \wedge (x_1 < x_2) \wedge (x_2 < x_3) \wedge (x_3 < y) \\ \wedge Q_{c_2}x_1 \wedge \neg Q_{c_2}x_2 \wedge Q_{c_2}x_3)))))\end{aligned}$$

We will consider the classical decision problems for MSO, but when restricted to words. What that means is the following. An MSO sentence φ (over the signature of words) is said to be *satisfiable* if there some *word* structure $\mathcal{W}$ such that $\mathcal{W} \models \varphi$. On the other hand, φ is *valid* if for every *word* structure $\mathcal{W}$, $\mathcal{W} \models \varphi$. Thus, satisfiability and validity are interpreted over word structures.