Monadic Second-Order Logic

Klaus Sutner
Carnegie Mellon University

Fall 2014
Second-Order Logic

- Words as Structures
Intuitively, FOL allows us to ask “local” queries about an automatic structure.

For example, given an automatic graph \( \langle V, E \rangle \) we can describe the local neighborhood of any vertex. We can also assert that there are paths of a certain, fixed length. But we cannot express more global properties such as connectivity: we cannot handle paths of arbitrary length.

This and other limitations makes it natural to try to look for stronger logics and hope they might still admit algorithms for model checking.

**Exercise**

*Use the compactness theorem for FOL to show that connectivity is not first-order definable.*
For example, suppose you are doing calculus and want to say something along the lines of

For every differentiable function $f$, yarglebargle $f$ bargleblargh.

Even if we assume that the underlying structure is $\mathbb{R}$, this is not expressible. Things get worse when we try to make statements about some class of higher order functions, say, measures on some space. Things get worse yet if we try to have this discussion over a tame structure like $\mathcal{N}$.

We simply need more expressive power.
In first-order logic quantification takes place only over individuals.

Sets of individuals, and more generally relations and functions on the domain, are given by an appropriate first-order structure but cannot be quantified at the syntactical level.

And, of course, there is no way to quantify over, higher type objects such as families of functions.

This suggests a generalization: how about allowing quantification over all these objects? For example, rewritten in terms of set theory we would like to be able to make assertions

\[ \forall x \in \mathcal{P}(\mathcal{P}(A)) \ldots x \ldots \]

When \( A = \mathbb{N} \), this would allow us to talk about sets of reals, a perfectly natural thing to do.
J. Harrison has developed an HOL theorem prover, written in OCaml, that is surprisingly powerful and freely available.

http://www.cl.cam.ac.uk/~jrh13/hol-light/

As we see in a moment, no such prover can be complete. However, this one is capable of proving lots of interesting theorems and has had considerable success in the verification of fairly complicated fragments of math. It has also been used to verify floating-point algorithms for Intel.
1994: Pentium FDIV Bug
4195835.0/3145727.0 = 1.3338204491362410025 correct
4195835.0/3145727.0 = 1.3337390689020375894 pentium

Alternatively

4195835.0 − 3145727.0 * (4195835.0/3145727.0) = 0 correct
4195835.0 − 3145727.0 * (4195835.0/3145727.0) = 256 pentium

Discovered in October 1994 by number theorist Thomas R. Nicely, doing research in pure math.
HOL is a huge sledge-hammer. As a first step towards more powerful quantification, consider so-called second-order logic (SOL) where one can only quantify over

- individuals,
- sets of individuals (monadic fragment),
- \( k \)-ary relations on individuals.

This is called second-order logic (SOL) and turns out to be in essence just as powerful as full HOL.

Incidentally, both FOL and SOL were introduced more or less by Frege in his *Begriffsschrift* in 1879.
In 1879 Frege published his *Begriffsschrift*, which translates roughly as “concept script.” He establishes a language and a (very powerful) logic, the groundwork for the Grundgesetze.

Unfortunately, Frege developed a horrible, two-dimensional notation system.

```
  | B
  |   A
  |   A
  |   a A

A \Rightarrow B
\neg A
\forall a A
```

This may seem harmless, but if one constructs larger formulae from these primitives, things start to look quite ominous.
In modern notation (arguably, a huge improvement):

\[
[\forall a \ (P(x, a) \Rightarrow Q(a)) \Rightarrow (P(x, y) \Rightarrow Q(y))] \Rightarrow \\
[\forall b \ (Q(b) \Rightarrow \forall a \ (P(b, a) \Rightarrow Q(a)))] \Rightarrow \\
Q(x) \Rightarrow \\
P(x, y) \Rightarrow Q(y)
\]

There are 4 premises, and altogether they imply \(Q(y)\).

**Notation matters!**
When writing SOL formulae (in modern notation), it is sometimes helpful to indicate the arity of relation and function symbols by a superscript. So, for example,

$$\forall R^2 (\ldots R(x, y)\ldots)$$

is a statement about all binary relations.

Note that one can get by without function symbols. For example, we can fake unary functions like so. Instead of $\exists f^1 \ldots$ we write

$$\exists R^2 (\forall x \exists y R(x, y) \land \forall x, y, z (R(x, y) \land R(x, z) \Rightarrow y = z) \land \ldots)$$
We could also get rid of equality on individuals:

\[ x = y \iff \forall R^1 (R(x) \iff R(y)) \]

The importance of this idea was first recognized by Leibniz and is enshrined in his *principlum identitatis indiscernibilium*: if we cannot tell two entities apart, they are identical.
With second-order quantification one can avoid the effects of the compactness theorem for first-order.

For example, suppose we use the standard Peano axioms to describe arithmetic. Then add the following axiom:

\[ \forall x \forall X (X(0) \land \forall z (X(z) \Rightarrow X(z+1)) \Rightarrow X(x)) \]

Here \( X \) is a subset of the universe.

In other words, every set containing 0 and closed under successors is already the whole universe: this rules out the non-standard, “infinite” elements.

In fact, SOL can be used to produce a finite axiomatization of arithmetic that has only one model (categoricity).
One standard application of axioms is to describe all examples of some particular kind of structure.

For example, the standard (first-order) axioms for groups describe the whole, large and very complex class of all groups.

The other standard application is to construct a set of axioms with the goal of pinning down one particular structure precisely.

For example, the Peano axioms form an attempt to give a precise and complete description of the natural numbers. Alas, if first-order this attempt fails, there are non-standard models.
Full SOL is quite powerful, but also quite unwieldy. In particular proof theory breaks down in the following sense: we cannot construct a proof system for SOL in a way that it satisfies the three standard requirements:

- **Sound**: only valid formulae are provable,
- **Complete**: all valid formulae are provable,
- **Effective**: the collection of all proofs is decidable.

This has lead some people like Willard Quine to deny that SOL is “a logic” at all. In their view it’s just a branch of set theory.
The reason there can be no adequate notion of proof for SOL is the following: for every sentence $\varphi$ of arithmetic in FOL, one can effectively construct a sentence $\hat{\varphi}$ of SOL such that

$$\hat{\varphi} \text{ valid } \iff \mathbb{N} \models \varphi.$$ 

Since proofs are always semidecidable (r.e.), the existence of a deduction system for SOL would immediately imply that the LHS is also semidecidable.

But Gödel and Tarski have shown that the RHS is far from being semidecidable.

In fact, validity of SOL as a decision problem is hideously complicated.
Often it suffices to consider a weak subsystem for full SOL where quantification is restricted to just two kinds:

- individuals, and
- sets of individuals.

Notation:

\[
\exists X \quad \forall X \\

x \in X \quad X(x)
\]
Assuming a binary total order $\leq$, we can express the assertion that every bounded set has a least upper bound:

$$\forall X \left( \exists z X(z) \wedge \exists x \forall z (X(z) \Rightarrow z \leq x) \Rightarrow \right.$$

$$\exists x \left( \forall z (X(z) \Rightarrow z \leq x) \wedge \forall y \left( \forall z (X(z) \Rightarrow z \leq y) \Rightarrow x \leq y \right) \right)$$

This is the critical property of the reals and cannot be expressed in FOL.
Again assume a binary total order $\leq$. We can express the assertion that we have a well-order in terms of the least-element principle: every non-empty set has a least element.

$$\forall X \left( \exists z X(z) \Rightarrow \exists x \left( X(x) \land \forall z \left( X(z) \Rightarrow x \leq z \right) \right) \right)$$

This is the critical property of the natural numbers with the standard order, and cannot be expressed in FOL.
Lastly, consider a digraph, a single binary edge relation $\rightarrow$.

We can express the assertion that there is a path from $s$ to $t$ as follows:

$$\forall X (X(s) \land \forall x, y (X(x) \land x \rightarrow y \Rightarrow X(y)) \Rightarrow X(t))$$

Again, FOL is not strong enough to express path existence in general (and thus other concepts like connectivity).
Second-Order Logic

Words as Structures
So far we have considered structures whose elements are words, and whose relations are synchronous.

**Wild Idea:** Can be think of a single word as a structure?

And, of course, use logic to describe the properties of the structure? This may seem a bit weird, but bear with me. First, we need to fix an appropriate language for our logic. As always, we want at least propositional logic.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \bot )</td>
<td>false, true</td>
</tr>
<tr>
<td>( \neg )</td>
<td>not, negation</td>
</tr>
<tr>
<td>( \land )</td>
<td>and, conjunction</td>
</tr>
<tr>
<td>( \lor )</td>
<td>or, disjunction</td>
</tr>
<tr>
<td>( \Rightarrow )</td>
<td>implication</td>
</tr>
<tr>
<td>( \Leftrightarrow )</td>
<td>equivalence</td>
</tr>
</tbody>
</table>
We will have variables $x, y, z, \ldots$ that range over positions in a word, integers in the range 1 through $n$ where $n$ is the length of the word.

We allow the following basic predicates between variables:

\[ x < y \quad x = y \]

Of course, we can get, say, $x \geq y$ by Boolean operations.

Most importantly, we write

\[ Q_a(x) \]

for “there is a letter $a$ in position $x$. “
We allow quantification for position variables.

\[ \exists x \varphi \quad \forall x \varphi \]

For example, the formula

\[ \exists x, y \ (x < y \land Q_a(x) \land Q_b(y)) \]

intuitively means “somewhere there is an a and somewhere, to the right of it, there is a b.”

The formula

\[ \forall x, y \ (Q_a(x) \land Q_b(y) \Rightarrow x < y) \]

intuitively means “all the a’s come before all the b’s.”
We also have second-order variables $X, Y, Z, \ldots$ that range over sets of positions in a word.

\[ \exists X \varphi \quad \forall X \varphi \quad X(x) \]

Sets of positions are all there is; we do not have variables in our language for, say, binary relations on positions (we do not use full SOL).

This system is called monadic second-order logic (with less-than), written $\text{MSO}[<]$. 
We need some notion of satisfaction $w \models \varphi$ where $w$ is a word and $\varphi$ a sentence in MSO[$\prec$].

We won’t give a formal definition, but the basic idea is simple: Let $|w| = n$:

- the first order variables range over $[n] = \{1, 2, \ldots, n\}$,
- the second-order variables range over $\mathcal{P}([n])$.

The basic predicates $x < y$ and $x = y$ have their obvious meaning. For the $Q_a(x)$ predicate we let

$$Q_a(x) \iff w_x = a$$
Examples

\[ aaacbbb \models \forall x \left( Q_a(x) \lor Q_b(x) \lor Q_c(x) \right) \]

\[ aaabbb \models \exists x, y \left( x < y \land Q_a(x) \land Q_b(y) \right) \]

\[ bbbaaa \not\models \exists x, y \left( x < y \land Q_a(x) \land Q_b(y) \right) \]

\[ aaabbb \models \exists x, y \left( x < y \land \neg \exists z \left( x < z \land z < y \right) \land Q_a(x) \land Q_b(y) \right) \]

\[ aaacbbb \not\models \exists x, y \left( x < y \land \neg \exists z \left( x < z \land z < y \right) \land Q_a(x) \land Q_b(y) \right) \]

\[ aaacbbb \models \exists x \left( Q_c(x) \Rightarrow \forall y \left( x < y \Rightarrow Q_b(y) \right) \right) \]
In applications, the atomic relation $x < y$ is slightly more useful than $y = x + 1$, but either one would have the same expressiveness.

On the one hand

$$y = x + 1 \iff x < y \land \forall z \,(x < z \Rightarrow y \leq z)$$

On the other hand write $\text{closed}(X)$ for the formula $\forall z \,(X(z) \Rightarrow X(z + 1))$. Then

$$x < y \iff x \neq y \land \forall X \,(X(x) \land \text{closed}(X) \Rightarrow X(y))$$

This is sometimes written as $\text{MSO}[<] = \text{MSO}[+1]$. 
Example

We can hardwire factors. For example, to obtain a factor \( abc \) let

\[
\varphi \equiv \exists x, y, z \ (y = x + 1 \land z = y + 1 \land Q_a(x) \land Q_b(y) \land Q_c(z))
\]

Then \( w \mid= \varphi \) iff \( w \in \Sigma^* abc \Sigma^* \).

Example

Scattered subwords are very similar in this setting:

\[
\varphi \equiv \exists x, y, z \ (x < y \land y < z \land Q_a(x) \land Q_b(y) \land Q_c(z))
\]

Then \( w \mid= \varphi \) iff \( w \in \Sigma^* a \Sigma^* b \Sigma^* c \Sigma^* \).
Example

We can split a word into two parts as in

$$\varphi \equiv \exists x \forall y ( (y \leq x \Rightarrow Q_a(y)) \land (y > x \Rightarrow Q_b(y))) \lor \forall x (Q_b(x))$$

Then $w \models \varphi$ iff $w \in a^*b^*$. 

Example

Let first$(x)$ be shorthand for $\forall z (x \leq z)$, and last$(x)$ shorthand for $\forall z (x \geq z)$. Then

$$\varphi \equiv \exists x, y \ (\text{first}(x) \land Q_a(x) \land \text{last}(y) \land Q_b(y))$$

Then $w \models \varphi$ iff $w \in a\Sigma^*b$. 
The examples suggest that, for any sentence $\varphi$, we should consider the collection of all words that satisfy $\varphi$:

$$L(\varphi) = \{ w \in \Sigma^* \mid w \models \varphi \}.$$ 

One cannot fail to notice that, in the examples so far, $L(\varphi)$ is always regular. Needless to say, this is no coincidence.

Also note that we have not used the second-order part of our language yet.
Example

Write even($X$) to mean that $X$ has even cardinality and consider

$$\varphi \equiv \exists X \left( \forall x (Q_a(x) \iff X(x)) \land \text{even}(X) \right)$$

Then $w \models \varphi$ iff the number of $a$’s in $w$ is even.

We’re cheating, of course; we need to show that the predicate even($X$) is definable in our setting. This is tedious but not really hard:

$$\text{even}(X) \iff \exists Y, Z (X = Y \cup Z \land \emptyset = Y \cap Z \land \text{alt}(Y, Z))$$

Here alt($Y, Z$) is supposed to express that the elements of $Y$ and $Z$ strictly alternate as in

$$y_1 < z_1 < y_2 < z_2 < \ldots < y_k < z_k$$
\[ X = Y \cup Z \iff \forall u \ (X(u) \leftrightarrow Y(u) \vee Z(u)) \]
\[ \emptyset = Y \cap Z \iff \neg \exists u \ (Y(u) \land Z(u)) \]
\[ \text{alt}(Y, Z) \iff \exists y \in Y \ \forall x < y \ (\neg Z(x)) \land \exists z \in Z \ \forall x > z \ (\neg Y(x)) \land \forall y \in Y \ \exists z \in Z \ (y < z \land \forall x \ (y < x < z \Rightarrow \neg Y(x) \land \neg Z(x))) \land \forall z \in Z \ \exists y \in Y \ (y < z \land \forall x \ (y < x < z \Rightarrow \neg Y(x) \land \neg Z(x))) \]

**Exercise**

*The alt formula above does not handle the case where \( Y \) and \( Z \) are empty; fix this.*

*Show that one can check if the number of \( a \)'s is a multiple of \( k \), for any fixed \( k \).*
Definition

A language $L$ is $\text{MSO}[<]$ definable (or simply $\text{MSO}[<]$) if there is some sentence $\varphi$ such that

$$L = \mathcal{L}(\varphi) = \{w \in \Sigma^* \mid w \models \varphi\}.$$ 

Our examples suggest the following theorem.

**Theorem (Buechi 1960, Elgot 1961)**

A language is regular if, and only if, it is $\text{MSO}[<]$ definable.

The theorem connects complexity with definability: we can recognize a set of strings in constant space if, and only if, the set can be described by a formula in our logic.
Obviously, the proof comes in two parts:

- For every regular language $L$ we need to construct a sentence $\varphi$ such that $L = \mathcal{L}(\varphi)$.

- For every sentence $\varphi$ we have to show that the language $\mathcal{L}(\varphi)$ is regular.

We should expect part (1) to be harder since there is no good inductive structure to exploit.

Part (2) is by straightforward induction on $\varphi$, but there is the usual technical twist: we need to deal not just with sentences but also with free variables. Since we don’t have a formal semantics we will not give details of this construction, but see the next section for a very similar argument.
We may safely assume that the regular language \( L \) is given by a DFA
\[ M = \langle Q, \Sigma, \delta; q_0, F \rangle. \]
For simplicity assume \( Q = [n] \) and \( q_0 = 1 \).
We have to construct a formula \( \varphi \) such that \( w \models \varphi \) iff \( M \) accepts \( w \).
Consider a trace of \( M \) on input \( w \)
\[ q_0 w_1 q_1 w_2 q_2 \cdots q_{m-1} w_m q_m. \]
Here \( m \) can be arbitrarily large.

We can think of states as being associated with the letters of the word as in
\[ w_1 \ w_2 \ w_3 \ \cdots \ w_m \]
\[ q_0 \ q_1 \ q_2 \ q_3 \ \cdots \ q_m \]
Thus, position \( x = 1, \ldots, m \) in the word is associated with state
\( \delta(q_0, w_1 \ldots w_x) \).
In order to express this in a MSO[<] formula, we partition the set positions $[m]$ into $n = |Q|$ blocks $X_1, X_2, \ldots, X_n$ such that

$$X_p(x) \iff \delta(q_0, w_1 \ldots w_x) = p$$

Some of these blocks may be empty, but note that the number of blocks is always exactly $n$ (which we can express as a formula).

But given state $p$ in position $x$ we can determine the state in position $x + 1$ given $w_{x+1}$ by a table lookup – which table lookup can be hardwired in a formula.
Expressing Transitions

Technically, this is done by a formula

$$\Phi_{p,a} \equiv \forall x \left( X_p(x) \land Q_a(x+1) \Rightarrow X_{\delta(p,a)}(x+1) \right)$$

meaning “if at position $x$ we are in state $p$ and the next letter is an $a$, then the state in position $x + 1$ is $\delta(p,a)$.

Note that this is not quite right, we really need a non-existing position 0 corresponding to state $q_0$.

Exercise

*Figure out how to fix this little glitch. Also figure out how to express “the last state is final.”*
Now consider the big conjunction of $\Phi_{p,a}$ where $p \in Q$ and $a \in \Sigma$. Add formulae that pin down the first and last state to arrive at a formula of the form

$$\varphi \equiv \exists X_1, \ldots, X_n \Psi$$

where $\Psi$ is first-order as indicated above. \hfill $\square$

Note that in conjunction with the opposite direction of Büchi’s theorem, this result has the surprising consequence that every $\text{MSO}[<]$ formula is equivalent to a $\text{MSO}[<]$ formula containing only one block of existential second-order quantifiers.

**Exercise**

*Fill in all the details in the last proof.*
It is natural to ask whether the languages defined by the first-order fragment of MSO[<] have some natural characterization.

A language $L \subseteq \Sigma^*$ is star-free iff it can be generated from $\emptyset$ and the singletons $\{a\}$, $a \in \Sigma$, using only operations union, concatenation and complement (but not Kleene star).

Note well: $a^*b^*a^*$ is star-free.

**Theorem**

A language $L \subseteq \Sigma^*$ is FOL[<] definable if, and only if, $L$ is star-free.
While we’re at it: star-free languages are quite interesting since they admit a purely algebraic characterization in terms of their syntactic semigroups.

A semigroup is aperiodic if it contains only trivial subgroups (the idempotents of the semigroup).

**Theorem (Schützenberger 1965)**

*A regular language is star-free if, and only if, its syntactic semigroup is aperiodic.*
The Büchi/Elgot theorem establishes a connection between a very low complexity class (constant space) and MSO[+1]. In fact, there is the whole area of descriptive complexity that characterizes complexity classes in terms of logic and finite structures:

- **NP** corresponds to existential SOL (Fagin 1974).
- **PH** corresponds to SOL.
- **PSPACE** corresponds to SOL plus a transitive closure operator.

**Exercise**

*Figure out the details of Fagin’s theorem.*
We have seen that there is a deep connection between regular languages and certain weak logics, notably monadic second-order logic.

Büchi’s result may seem a bit odd and of no practical value.

However, it also holds for infinite words and in that version it is extremely useful.

Next goal: push our theory to infinite words.