MSO logic & Automata

Elli Anastasiadi

Descriptive Complexity 2020 - ALMA

June 9, 2020

Outline

1 Intro

2 Graphs, Strings and Regular Languages

3 Tree Automata

4 Complexity

We'll get there when we get there.

What is Second Order Logic?

Definition

Second order Logic = FO + variables ranging over predicates (and quantification over them)

Semantics

= semantics of FO and second order variables are all the functions (or sets) of the appropriate sort. Once the domain of the first order variables is set, the second order elements are defined too.

Expressive Power A lot.

We'll get there when we get there.

What is Second Order Logic?

Definition

Second order Logic = FO + variables ranging over predicates (and quantification over them)

Semantics

= semantics of FO and second order variables are all the functions (or sets) of the appropriate sort. Once the domain of the first order variables is set, the second order elements are defined too.

Expressive Power A lot. We have formal sentences which say "the domain is finite" or "the domain is of countable cardinality." (finite = every surjective function from the domain to itself is injective.)

Normalization Rules

Every SO formula can be written as a sequence of first- and second-order quantifiers, followed by a quantifier-free formula (This can be done by following the normalization procedure of first order logic).
Aaaaaand!

$$\exists x \, \mathbf{Q} \, \phi(x, \cdot) \leftrightarrow \exists X \, \mathbf{Q} \, \exists x \, (X(x) \wedge \phi(x, \cdot))) \tag{1}$$

$$\forall x \, \mathbf{Q} \, \phi(x, \cdot) \leftrightarrow \forall X \, \mathbf{Q}(\exists! x \, X(x) \rightarrow \forall x(X(x) \rightarrow \phi(x, \cdot))) \tag{2}$$

Repeat.

Normalization Rules

Every SO formula can be written as a sequence of first- and second-order quantifiers, followed by a quantifier-free formula (This can be done by following the normalization procedure of first order logic).
Aaaaaand!

$$\exists x \ \mathbf{Q} \ \phi(x, \cdot) \leftrightarrow \exists X \ \mathbf{Q} \ \exists x \ (X(x) \wedge \phi(x, \cdot))) \quad (1)$$

$$\forall x \ \mathbf{Q} \ \phi(x, \cdot) \leftrightarrow \forall X \ \mathbf{Q}(\exists ! x \ X(x) \rightarrow \forall x(X(x) \rightarrow \phi(x, \cdot))) \quad (2)$$

Repeat.
We are making sure that the formula looks like:
[SO quantifications],[FO quantifications],[Quantifier free formula].
It will come in handy later.

Descriptive Complexity (Lets not forget what this course is called)

- *NP* is the set of languages definable by existential, second-order formulas (Fagin's theorem < 3 , 1974).
- $co - NP \rightsquigarrow$ universal, second-order formulas.
- $PH \rightsquigarrow$ second-order formulas.
- $PSPACE \rightsquigarrow$ second-order formulas with an added transitive closure operator.
- $EXPTIME \rightsquigarrow$ second-order formulas with an added least fixed point operator.
- Bonus (Today, with us! Specifically for MSO!!!1!).

Monadic - Finally!

Definition

MSO = SO but only second order variables of arity 1 .

Easy right? Normalization still applies. (Rules 1 and 2 only added SO variables of arrity 1 ;).)

Now the vocabulary of the model might actually play a role on the expressiveness (it is giving us indirect access to predicates of larger arrity).

Games

Definition

MSO game Spoiler and duplicator, on two structures A and B of the same vocabulary σ. The game has two different (not really) kinds of moves:

- **Point move**: This is the same move as in the Ehrenfeucht-Fraisse game for FO: the spoiler chooses a structure, A or B, and an element of that structure; the duplicator responds with an element in the other structure.
- **Set move**: The spoiler chooses a structure, A or B, and a subset of that structure. The duplicator responds with a subset of the other structure.

Up-Down its almost like FO.

More games

A k round game gives the expressibility class of MSO properties of quantifier rank $[k]$.

Theorem (Proposition 7.9 - Libkin)

A property P of σ-structures is expressible in MSO iff there is a number k such that for every two σ-structures A, B, if A has the property P and B does not, then the spoiler wins the k-round MSO game on A and B.

More games

A k round game gives the expressibility class of MSO properties of quantifier rank $[k]$.

Theorem (Proposition 7.9 - Libkin)

A property P of σ-structures is expressible in MSO iff there is a number k such that for every two σ-structures A, B, if A has the property P and B does not, then the spoiler wins the k-round MSO game on A and B.

Proof.

No. (we have seen many similar proofs and there are way more interesting -and harder ones < 3- later!) □

Properties

- For $\sigma = \emptyset$, *EVEN* is not expressible in MSO.
  - proof?
- For $\sigma = \{<\}$ (a linear ordering), *EVEN* **is** expressible in MSO.
  - proof!

Outline

1 Intro

2 Graphs, Strings and Regular Languages

3 Tree Automata

4 Complexity

Graphs

$\sigma = \{V, E\}$

- Graph connectivity is expressible in $\forall$MSO, but is not expressible in $\exists$MSO. (+ positive part of the proof - non connectivity is in $\exists$MSO, by identifying the non connected components.)
- For undirected graphs without loops, (s, t)-reachability is expressible in $\exists$MSO. (+ construction via $\exists X$ where X is the path.)
- Reachability for directed graphs is not expressible in $\exists$MSO.

Strings

$\sigma = \{<, P_a, P_b, \ldots P_g\}$:one predicate for each symbol in the alphabet of the strings.

- The linear ordering puts the elements on a line.
- the symbol predicates tell us when an element is of the type of the predicate.

Example

Example: *aaabcba* is encoded as:
$\{\{1,2,3,4,5,6,7\}, <, P_a, P_b, P_c\}$ where $P_a = \{1,2,3,7\}$,
$P_b = \{4,6\}$, $P_c = \{5\}$.

W.L.O.G. $\sigma = \{<, P_a, P_b\}$

More strings!

Theorem (Büchi - 1960)

A language $\mathcal{L}$ is definable in MSO (over strings) iff it is regular.

More strings!

Theorem (Büchi - 1960)

A language $\mathcal{L}$ is definable in MSO (over strings) iff it is regular.

Proof: $\mathcal{L}$ is regular $\Rightarrow$ $\mathcal{L}$ is expressible in MSO.

Start from the DFA of $\mathcal{L}$.

$\Phi := \exists X_0 \ldots \exists X_{m-1} \phi_{part} \wedge \phi_{start} \wedge \phi_{trans} \wedge \phi_{accept}$ □

$\mathcal{L}$ is expressible in MSO $\Rightarrow$ $\mathcal{L}$ is regular.

Make all rank-k types (what is a type Elli?) over the vocabulary of strings(finitely many) = states in the automaton. Transition function: update current type based on symbol. Initial state: type of empty sting formulas. Final states:types compatible with the original formula. □

Disclaimer: This is a very short version of the proof in the book.

MORE STRINGS!!

Note that via the previous proof: $\exists$MSO = MSO (over strings).
Corollary: Hamiltonian $\notin$ MSO (Over Graphs!)

MORE STRINGS!!

Note that via the previous proof: $\exists$MSO $=$ MSO (over strings).
Corollary: Hamiltonian $\notin$ MSO (Over Graphs!)
This was quite cool right? Let's do it again! This time for FO.

MORE STRINGS!!

Note that via the previous proof: $\exists$MSO = MSO (over strings).
Corollary: Hamiltonian $\notin$ MSO (Over Graphs!)
This was quite cool right? Let's do it again! This time for FO.

Star free languages

Star- free = Regular - Kleene star (Duh)
But we still have complement ($\overline{s}$)

(Basically automata without loops backwards)

Example (Star free languages)

$$(\sum_{a \in \Sigma} a)^* = \overline{\epsilon},$$

MORE STRINGS!!

Note that via the previous proof: $\exists$MSO $=$ MSO (over strings).
Corollary: Hamiltonian $\notin$ MSO (Over Graphs!)
This was quite cool right? Let's do it again! This time for FO.

Star free languages

Star- free $=$ Regular - Kleene star (Duh)
But we still have complement ($\overline{s}$)

(Basically automata without loops backwards)

Example (Star free languages)

$$(\sum_{a \in \Sigma} a)^* = \overline{\epsilon}, \qquad a^* b^* =$$

MORE STRINGS!!

Note that via the previous proof: $\exists$MSO = MSO (over strings).
Corollary: Hamiltonian $\notin$ MSO (Over Graphs!)
This was quite cool right? Let's do it again! This time for FO.

Star free languages

Star- free = Regular - Kleene star (Duh)
But we still have complement ($\overline{s}$)

(Basically automata without loops backwards)

Example (Star free languages)

$$(\sum_{a\in\Sigma} a)^* = \overline{\epsilon}, \qquad a^*b^* = \overline{\overline{\epsilon} \cdot b \cdot a \cdot \overline{\epsilon}}$$

MORE STRINGS!!

Note that via the previous proof: $\exists$MSO = MSO (over strings).
Corollary: Hamiltonian $\notin$ MSO (Over Graphs!)
This was quite cool right? Let's do it again! This time for FO.

Star free languages

Star- free = Regular - Kleene star (Duh)
But we still have complement ($\overline{s}$)

(Basically automata without loops backwards)

Example (Star free languages)

$$(\sum_{a\in\Sigma} a)^* = \overline{\epsilon}, \qquad a^*b^* = \overline{\overline{\epsilon}\cdot b\cdot a\cdot\overline{\epsilon}}$$

Theorem

Star Free Languages = FO on strings

MORE STRINGS!!

Note that via the previous proof: $\exists$MSO = MSO (over strings).
Corollary: Hamiltonian $\notin$ MSO (Over Graphs!)
This was quite cool right? Let's do it again! This time for FO.

Star free languages

Star- free = Regular - Kleene star (Duh)
But we still have complement ($\overline{s}$)

(Basically automata without loops backwards)

Example (Star free languages)

$$(\sum_{a \in \Sigma} a)^* = \overline{\epsilon}, \qquad a^* b^* = \overline{\overline{\epsilon} \cdot b \cdot a \cdot \overline{\epsilon}}$$

Theorem

Star Free Languages = FO on strings

BREAK

Outline

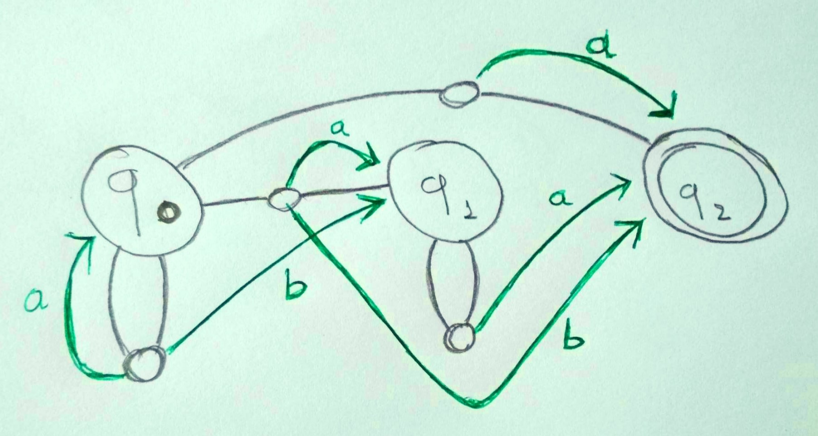
1. Intro
2. Graphs, Strings and Regular Languages
3. Tree Automata
4. Complexity

Formally

- Automata that read trees.
- Many variations (Deterministic/Not, Bottom-UP/Top-Down, Ranked/Unranked)
- We will see **Ranked Non-Deterministic Bottom-Up** (equal to all except deterministic Top-Down)
- Distinctive difference is that the transition function takes *tuples* of states and gives one state (going up on the tree).
- I'll explain the rest in the example.

Example

$\mathcal{L}$ = Binary trees with 2 total "b" labels.



Regular Tree Languages

Tree models: $M_T = \{D, <, P_{a,\ a\in\Sigma}, succ_1, succ_2\}$
D is a subset of $\{0,1\}^*$ that is prefix closed and if $s \in D$ then either both $s.0$, $s.1$ are in D or none of them.
$<$ is a partial ordering and the rest of the predicates are doing their obvious jobs.

Theorem

A set of trees is definable in MSO iff is regular (has a tree automaton).

Corollary

MSO over trees = $\exists$MSO over trees.
Non-deterministic tree automata = Deterministic tree automata.

Ranked and Unranked tree automata are equivalent.

Outline

1 Intro

2 Graphs, Strings and Regular Languages

3 Tree Automata

4 Complexity

MSO is quite slow in general

Theorem

For each level Σ_i^p or Π_i^p of the polynomial hierarchy, there exists a problem complete for that level which is expressible in MSO. :'(

MSO is quite slow in general

Theorem

For each level Σ_i^p or Π_i^p of the polynomial hierarchy, there exists a problem complete for that level which is expressible in MSO. :'(

Proof sketch: Start from the QBF problem (*PSPACE*-complete).
Take a formula ϕ restricted to i blocks of quantifier alterations (with the propositional part being in 3-CNF form).
Define as $i+1$ unary predicates the variables occurring in each block (model predicates).
Transform ϕ to ϕ' where instead of satisfying the clauses we have to satisfy one of the 4 predicates that enumerate the ways that 3 variables can occur in a clause $((x, y, z), (x, y, \neg z)$, etc):

$$\exists X_1 \subseteq E_1 \forall X_2 \subseteq E_2 \exists X_3 \subseteq E_3 ... \phi' \ .$$

ϕ' is in MSO ϕ is SAT iff ϕ' is.

So what did we do all of this for?

Corollary (7.36)

Over strings and trees (ranked and unranked), evaluating MSO sentences is fixed-parameter linear. In particular, over strings and trees, the data complexity of MSO is linear.

Proof: Just make the automaton and run it (time for making the automaton is not counted in the complexity)

Thats not good enough!

Thats not good enough!

Theorem (Courcelle)

Let $\mathcal{C}$ be a class of structures of bounded treewidth. Then evaluating MSO sentences over $\mathcal{C}$ is fixed-parameter linear. In particular, the data complexity of MSO over $\mathcal{C}$ is linear.

Proof: Bounded treewidth = Enumerate all graphs of treewidth k. Modify the formula by adding existential quantification over them to find a model that satisfies the original formula.

Questions?

References & cool links

- Elements of Finite Model Theory ch. 7
  Leonid Libkin
- Barry Cooper prize 2020 to Bruno Courcelle
  For the theorem we just proved!
- A Finite Model Theorem for the Propositional μ-Calculus
  A nice paper i would like us to look at

Elli Anastasiadi
elli19@ru.is