Monadic Second Order Logic

Hossein Hojjat

LARA

November 26, 2010
<table>
<thead>
<tr>
<th>BL</th>
<th>QBF</th>
<th>LTL</th>
<th>WS1S</th>
<th>S1S</th>
<th>WS2S</th>
<th>FOL</th>
<th>HOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(2^n)$</td>
<td>$O(2^n)$</td>
<td>$O(</td>
<td>M</td>
<td>.2^{</td>
<td>P</td>
<td>})$</td>
<td>non-elementary: $2^2 \cdot 2^n$</td>
</tr>
</tbody>
</table>

$n$: #vars  
$n$: ($\forall | + |\exists |$)  
$M$: Model  
$P$: Property  
n: length of the formula

Decidable Logics

Undecidable Logics
Second Order Logic

- **FOL** is the logic of quantification over the elements of a type
  - $\forall x.\Phi(x)$
    - For every individual $x$, $\Phi(x)$

- **Second order logic** is the logic of quantification over the predicates
  - $\forall P.\forall x. P(x)$
    - For every set of individuals $P$ and for every individual $x$, $x \in P$
  - $\exists R.\forall x. R(x, x)$
    - There exists a relation $R$ such that for every individual $x$, $R(x, x)$

- **Monadic Second Order (MSO)**: The fragment of the second order logic which allows only quantification over sets
S1S

- S1S: Monadic second order logic of one successor
- The fragment of MSO interpreted on discrete linear orders ($\leq$)

Let $\{x_1, \cdots, x_n\}$ be a family of first-order variables and $\{X_1, \cdots, X_n\}$ a family of second-order monadic variables.

S1S is defined on the signature $(\mathbb{N}, S)$ as the following:

- $t := 0 | x_i$
- $f := S(t, t) | X_i(t) | \neg f | f \land f | \exists x_i . f | \exists X_i . f$

$S$ is the successor predicate.

The predicate $S$ and $\leq$ can be defined from each other.

WS1S: The fragment of S1S which allows only quantification over finite sets.
WS1S Semantics

- **Signature:** Natural numbers $\langle \mathbb{N}, S \rangle$
- **Interpretation:** $x \xrightarrow{I} n \in \mathbb{N}$ and $X \xrightarrow{I} N \in 2^\mathbb{N}$ such that $N$ is finite
- **Truth value of a formula with respect to interpretation $I$**

\[
egin{align*}
I \models Y(x) & \iff I(x) \in I(Y) \\
I \models S(x, y) & \iff I(x) + 1 = I(y) \\
I \models \neg \Phi & \iff I \not\models \Phi \\
I \models \Phi_1 \land \Phi_2 & \iff I \models \Phi_1 \text{ and } I \models \Phi_2 \\
I \models \exists x. \Phi & \iff I[n/x] \models \Phi, \text{ for some } n \in \mathbb{N} \\
I \models \exists X. \Phi & \iff I[N/X] \models \Phi, \text{ for some finite } N \in 2^\mathbb{N}
\end{align*}
\]
Word Model

- Finite alphabet $\Sigma$ is given
- Word is defined as $\omega = a_0 \cdots a_{n-1}$ where $a_0, \cdots, a_{n-1} \in \Sigma$
- Domain of $\omega$: $\text{dom}(\omega) = \{0, \cdots, |\omega| - 1\}$
- A unary predicate $P_\alpha$ is defined for every $\alpha \in \Sigma$ such that $P_\alpha(i)$ if and only if $a_i = \alpha$
- The word $\omega$ defines a word model $\underline{\omega} = (\text{dom}(\omega), S^\omega, P_{a_0}, \cdots, P_{a_{n-1}})$

Example

Let $\Sigma = \{a, b\}$ and $\omega = aabba$

$\text{dom}(\omega) = \{0, 1, 2, 3, 4\}$

$P_a = \{0, 1, 4\}$

$P_b = \{2, 3\}$
Given an alphabet $\Sigma$, the logic S1S can also be defined on the signature of the words:  $\{\leq, (P_\alpha)_{\alpha \in \Sigma}\}$ or $\{S, (P_\alpha)_{\alpha \in \Sigma}\}$

- $\exists x \exists y. P_a(x) \land P_b(y) \land x \leq y \land \neg \exists z. (x < z \land z < y)$
- Word contains the substring $ab$

- $\exists x. P_a(x) \land \neg \exists y. (x < y)$
- The last symbol is $a$: $P_a(last)$

- $\exists X. (X(first) \land \forall y \forall z. (S(y, z) \rightarrow (X(y) \leftrightarrow \neg X(z)))) \land \neg X(last)$
- The length of the word is even
We can check a set $X$ to see if it is singleton

$$\text{Sing}(X) \equiv \exists Y. Y \subseteq X \land Y \neq X \land \neg(\exists Z. Z \subseteq Y \land Z \neq Y)$$

$$(X = Y) \equiv X \subseteq Y \land Y \subseteq X$$

We can remove all the first-order variables if we allow $S$ and $\leq$ to be applied to singleton sets

The result belongs to MSO$_0$

$$\Phi ::= X \subseteq Y \mid S(X, Y) \mid \exists X. \Phi \mid \neg \Phi \mid \Phi_1 \land \Phi_2$$

$$\Phi_{MSO} = \forall x \forall y. (P_a(x) \land x < y \rightarrow P_b(y))$$

$$\Phi_{MSO_0} = \forall X \forall Y. (\text{Sing}(X) \land \text{Sing}(Y) \land X \subseteq P_a \land X < Y \rightarrow Y \subseteq P_b)$$
A language $L \subseteq \Sigma^*$ is regular if and only if it is expressible in weak monadic second-order logic on words.

A language $L \subseteq \Sigma^\omega$ is $\omega$-regular if and only if it is expressible in monadic second-order logic on words.
Proof

A language $L \subseteq \Sigma^*$ is regular if and only if it is expressible in weak monadic second-order logic on words.

**Automata $\Rightarrow$ Logic**

- Code the execution of an automaton.
- A formula with a structure similar to the following:
  $$\exists X_0 \cdots \exists X_n. \Phi_{\text{partition}} \land \Phi_{\text{start}} \land \Phi_{\text{transitions}} \Phi_{\text{accept}}$$

**Logic $\Rightarrow$ Automata**

- Construction based on induction on the structure of $\Phi$.
- $X_1 \subseteq X_2$, $X_1 \subseteq P_a$, $\text{Sing}(X_1)$, $S(X_1, X_2)$, $X_1 < X_2$
WS1S decidability

Decision Procedure

- Given a WS1S formula $\Phi$
- Translate $\neg \Phi$ to an automaton $A_{\neg \Phi} = (Q, \Sigma, \delta, q_0, F)$ accepting $\omega$ iff $\omega \models \neg \Phi$
- Output
  - $\Phi$ is valid when $A_{\neg \Phi}$ accepts the empty string
  - Return a counter model $\omega$ which belongs to the automaton
The MSO$_0$ formula $\Phi(X_1, \cdots, X_n)$ is interpreted in the word model of $\omega$ and the sets $K_1, \cdots, K_n$

- $K_i \in \text{dom}(\omega)$ represents a set of positions
- To code the models we use an alphabet $\Sigma \times \{0, 1\}^n$

Let $\Sigma = \{a, b\}$

<table>
<thead>
<tr>
<th>$\omega$</th>
<th>$K_1$</th>
<th>$K_2$</th>
<th>$K_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

$\omega = abba$

- $K_1 = \{2, 3\}$
- $K_2 = \emptyset$
- $K_3 = \{0, 3\}$
From Logic to Automaton: Base case

- $\# \in \Sigma$ is an arbitrary symbol
- $\ast \in \{0, 1\}^{n-2}$ is an arbitrary vector

$X_1 \subseteq X_2$

$X_1 \subseteq P_a$
From Logic to Automaton: Base case

$\text{Sing}(X_1)$

$\text{S}(X_1, X_2)$
From Logic to Automaton: Base case

\((X_1 < X_2)\)
From Logic to Automaton: Step case

¬\(\Phi\)

- Complement the automaton \(A_{\Phi}\) by flipping the final and non-final states
- \(L(\neg\Phi) = \overline{L(\Phi)} = \overline{L(A_{\Phi})} = L(A_{\neg\Phi})\)

\(\Phi_1 \land \Phi_2\)

- Product construction of \(A_{\Phi_1}\) and \(A_{\Phi_2}\)
- \(L(\Phi_1 \land \Phi_2) = L(\Phi_1) \cap L(\Phi_2) = L(A_{\Phi_1}) \cap L(A_{\Phi_2}) = L(A_{\Phi_1 \land \Phi_2})\)
From Logic to Automaton: Step case

\[ \exists X_i. \Phi \]

- \( A_{\exists X_i. \Phi} \) acts as \( A\Phi \) except that it guesses the values in the set \( X_i \)
- Projection on \( X_i \) by simply removing its track from the automaton

\[
\Phi(X_1, X_2) = X_1 < X_2
\]

\[
\exists X_1. \Phi(X_1, X_2)
\]
From Logic to Automaton: Step case

\[ \exists x_i. \Phi \]

- We should be careful of \((0^{n-1})^*\) suffix after projection from \(\Phi(x_1, \cdots, x_n)\) to \(\exists x_i. \Phi(x_1, \cdots, x_n)\)

\[ \Phi(x_1, x_2) = x_1(1) \land x_2(2) \]

\[ \exists x_2. \Phi(x_1, x_2) \]

Doesn’t include \(x_1 = \{1\}\)
For Logic to Automaton: Step case

$\exists X_i. \Phi$

- Right quotient of $L \subseteq \Sigma^*$ with $L' \subseteq \Sigma^*$
  
  $L / L' = \{ \omega \in \Sigma^* \mid \exists u \in L'. \omega u \in L \}$

- Define the projection function $\Pi_i : (\{0, 1\}^n)^* \to (\{0, 1\}^{n-1})^*$ such that $\Pi_i(\begin{pmatrix} b_1 \\ \vdots \\ b_{i-1} \\ b_i \\ b_{i+1} \\ \vdots \\ b_n \end{pmatrix}) = \begin{pmatrix} b_1 \\ \vdots \\ b_{i-1} \\ b_i+1 \\ \vdots \\ b_n \end{pmatrix}$

- $L(\exists X_i \Phi) = \Pi_i(L(\Phi))/(\{0\}^{n-1}) = \Pi_i(L(A\Phi))/(\{0\}^{n-1}) = L(A_\exists X_i \Phi)$
Correspondence between logical operators and basic automata
Constructive proof using induction on the formula structure
Construction results in a trivial DFA that accepts all the acceptable words
Shows that WS1S formulas define regular languages
Give a decision procedure for WS1S
State Explosion

- Negation requires determinization
- Existential quantification introduces non-determinism
- Quantifier alternation results in exponential blow-ups
  - $\forall X. \exists Y. \phi \equiv \neg \exists X. \neg \exists Y. \phi$
  - If $|A_{\phi}| = n$ then $|A_{\neg \exists Y. \phi}| = O(2^{|n|})$
  - $|A_{\neg \exists X. \neg \exists Y. \phi}| = O(2^{2^{|n|}})$
Corollary

- Presburger arithmetic is decidable
- We can translate a given formula in Presburger arithmetic to its equivalent in MSO logic
- Idea of encoding:
  - Encode $n \in \mathbb{N}$ as the set of positions in which there is a 1 in its binary representation
    \[ 17 = (10001)_2 \iff \{0, 4\} \]
  - Encode the addition of $x_1 \in \mathbb{N}$ and $x_2 \in \mathbb{N}$ as the MSO formula
    \[
    \exists X_{Result} \exists X_{Carry} \Phi(X_1, X_2, X_{Result}, X_{Carry})
    \]
  - $X_1, X_2, X_{Result}$ and $X_{Carry}$ represents the bits of $x_1, x_2$, result and carry during addition
- Interpretation with respect to \( k \)
- Domain is \([k] = \{0, \cdots, k\}\)
- Successor relation \( S \) restricted to \([k] \times [k]\)

**Semantics**

\[
\begin{align*}
k, I \models Y(x) &\iff I(x) \in I(Y), I(x) \in [k] \text{ and } I(X) \subseteq [k] \\
k, I \models S(x, y) &\iff I(x) + 1 = I(y), I(y) \in [k] \\
k, I \models \neg \Phi &\iff I \nvdash \Phi \\
k, I \models \Phi_1 \land \Phi_2 &\iff I \vdash \Phi_1 \text{ and } I \vdash \Phi_2 \\
k, I \models \exists x. \Phi &\iff I[n/x] \vdash \Phi, \text{ for some } n \in [k] \\
k, I \models \exists X. \Phi &\iff I[N/X] \vdash \Phi, \text{ for some } N \subseteq [k]
\end{align*}
\]

**Validity**

\( \models \Phi \) if and only if \( k, I \models \Phi \), for all \( I \) and \( k \in \mathbb{N} \)
## Satisfiability Examples

<table>
<thead>
<tr>
<th></th>
<th>WS1S</th>
<th>M2L-STR</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X \subseteq Y$</td>
<td>$X \mapsto {1}, Y \mapsto {1, 2}$</td>
<td>$k = 3, X \mapsto {1}, Y \mapsto {1, 2}$</td>
</tr>
<tr>
<td>$\exists X. \forall p. p \in X$</td>
<td>unsatisfiable</td>
<td>valid</td>
</tr>
<tr>
<td>$\forall k \in \mathbb{N}. X \mapsto {0, \ldots, k - 1}$</td>
<td>satisfiable for $k &gt; 0$</td>
<td>unsatisfiable for $k = 0$</td>
</tr>
<tr>
<td>$\exists X. \exists p. p \in X$</td>
<td>valid</td>
<td></td>
</tr>
</tbody>
</table>
**INSTANCE**: A formula $\Phi$ and $k \in \mathbb{N}$

**QUESTION**: Is there $\omega$ such that $|\omega| = k$ and $\omega$ satisfies $\Phi$?
BMC for M2L-STR

\[
\Phi ::= X \subseteq Y | S(X, Y) | \exists X. \Phi | \neg \Phi | \Phi_1 \land \Phi_2
\]

- Encode \( M \subseteq [k] \) by the Booleans \( b_0, \cdots, b_{k-1} \) so that \( i \in M \) iff \( b_i \) is true
- Translation from MSO to QBF with \( \lfloor . \rfloor_k : \text{MSO} \rightarrow \text{QBF} \)

\[
\begin{align*}
\lfloor X \subseteq Y \rfloor_k &= \land_{0 \leq i \leq k-1} (x_i \rightarrow y_i) \\
\lfloor S(X, Y) \rfloor_k &= Sing(x_0, \cdots, x_{k-1}) \land Sing(y_0, \cdots, y_{k-1}) \land \\
& \lor_{0 \leq i \leq k-1} (x_i \rightarrow y_{i+1}) \\
\lfloor \Phi_1 \land \Phi_2 \rfloor_k &= \lfloor \Phi_1 \rfloor_k \land \lfloor \Phi_2 \rfloor_k \\
\lfloor \neg \Phi \rfloor_k &= \neg \lfloor \Phi \rfloor_k \\
\lfloor \exists X. \Phi \rfloor_k &= \exists x_0 \cdots x_{k-1}. \lfloor \Phi \rfloor_k
\end{align*}
\]
BMC for WS1S

Theorem

- BMC for WS1S is non-elementary
- Proof.
  - Closed formulas are either valid or unsatisfiable
  - Closed formula $\Phi$ has a model of length $k$ iff $\Phi$ is valid
  - Validity in WS1S is non-elementary
Reference

- “Languages, Automata, and Logic”; Wolfgang Thomas, Chapter 7 of Handbook of Formal Languages vol. 3, 1997