The Logic-Automaton Connection (cont.)

Logical descriptions of regular languages often more succinct than corresponding regular expressions.

Monadic 2nd-order logic of one successor (WS1S)

Variables.

- $FO$ variables $p, q, \ldots$ interpreted over integers
- $MSO$ variables $X, Y, \ldots$ interpreted over finite set of integers

Syntax.

$$\varphi ::= \text{succ}(p, q) \mid X(p) \mid \neg \varphi \mid \varphi \land \varphi \mid \exists p \varphi \mid \exists X \varphi$$
WS1S semantics

Structure. Natural numbers \( N \) with successor relation

- \( I : \mathbb{N} \rightarrow n \in N \)
- \( I : X \rightarrow N \in \mathbb{F}(N) \)

Interpretation. \( \mathcal{I} : X \rightarrow N \in \mathbb{F}(N) \)

Truth value. (of a formula wrt interpretation \( \mathcal{I} \))

\[
\begin{align*}
\mathcal{I} \models Y(x) & \quad \text{iff} \quad \mathcal{I}(x) \in \mathcal{I}(Y) \\
\mathcal{I} \models \text{succ}(x, y) & \quad \text{iff} \quad \mathcal{I}(x) + 1 = \mathcal{I}(y) \\
\mathcal{I} \models \neg \varphi & \quad \text{iff} \quad \mathcal{I} \not\models \varphi \\
\mathcal{I} \models \varphi \land \psi & \quad \text{iff} \quad \mathcal{I} \models \varphi \quad \text{and} \quad \mathcal{I} \models \psi \\
\mathcal{I} \models \exists x \varphi & \quad \text{iff} \quad \mathcal{I}[n/x] \models \varphi, \text{ for some } n \in N \\
\mathcal{I} \models \exists X \varphi & \quad \text{iff} \quad \mathcal{I}[N/X] \models \varphi, \text{ for some } N \in \mathbb{F}(N)
\end{align*}
\]

Validity.

\( \models \varphi \) \quad \text{iff} \quad \mathcal{I} \models \varphi, \text{ for all interpretations } \mathcal{I} \)

Syntactic Sugar (cont.)

Intersection.

\( X \cap Y = Z \) \( : = \forall x(Z(x) \leftrightarrow X(x) \land Y(x)) \)

Subset.

\( Y \subseteq X \) \( : = \forall x(Y(x) \rightarrow X(x)) \)

Set equality.

\( Y = X \) \( : = Y \subseteq X \land X \subseteq Y \)

Emptiness.

\( X = \emptyset \) \( : = \forall Y(Y \subseteq X \rightarrow Y = X) \)

Singleton.

\( \text{Sing}(X) \) \( : = X \neq \emptyset \land \forall Y(Y \subseteq X \rightarrow (Y = X \lor Y = \emptyset)) \)

Syntactic Sugar

Standard connectives and quantifiers

\[
\begin{align*}
\varphi \lor \psi & \quad \text{for} \quad \neg(\neg \varphi \land \neg \psi) \\
\forall x \varphi & \quad \text{for} \quad \neg \exists x \neg \varphi \\
\vdots & \quad \vdots
\end{align*}
\]

Some definitions (\( n \in N \) fixed!)

\[
\begin{align*}
x = 0 & \quad : = \neg \exists z \text{succ}(z, x) \\
x = y & \quad : = \forall z \bigl(Z(x) \leftrightarrow Z(y)\bigr) \\
x = y + n & \quad : = \exists z_0 \ldots \exists z_n \bigl(z_0 = y \land x = z_n \land \bigwedge_{0 \leq i \leq n} \text{succ}(z_i, z_{i+1})\bigr) \\
X(x + n) & \quad : = \exists z \bigl(z = x + n \land X(z)\bigr) \\
x \leq y & \quad : = \forall U \bigl(U(y) \land \forall z \bigl(U(z + 1) \rightarrow U(z)\bigr) \rightarrow U(x)\bigr) \\
x < y & \quad : = x \leq y \land \neg x = y \\
\vdots & \quad \vdots
\end{align*}
\]

Minimal syntax

Syntax (and semantics): only MSO variables \( X,Y,… \)

\[
\psi : = X \subseteq Y \mid \text{Succ}(X,Y) \mid \neg \psi \mid \psi \land \psi \mid \exists X \psi
\]

- \( X \subseteq Y \) says "\( X \) is a subset of \( Y \)"
- \( \text{Succ}(X,Y) \) says "\( X = \{n\} \) and \( Y = \{n+1\} \) for some \( n \in \mathbb{N} \)"

Introduce for each FO variable \( x \) a fresh MSO variable \( \hat{x} \)

Translate formulas inductively to "minimal" syntax, e.g.,

\[
\begin{align*}
\forall x \exists y \bigl(\text{succ}(x, y) \land Z(y)\bigr) & \quad \rightarrow \quad \forall \hat{x} \bigl(\text{Sing}(\hat{x}) \rightarrow \exists \hat{y} \bigl(\text{Sing}(\hat{y}) \land \text{Succ}(\hat{x}, \hat{y}) \land \hat{y} \subseteq Z\bigr)\bigr)
\end{align*}
\]
Circuits in WS1S

Encode quantified Boolean logic in WS1S
\[ \forall x \exists y (x \leftrightarrow y) \sim \forall X \exists Y (X(0) \leftrightarrow Y(0)) \]

Logical gates as Boolean relations

\[
\begin{align*}
\text{not} & (a, o) := -a \leftrightarrow o \\
\text{and} & (a, b, o) := a \land b \leftrightarrow o \\
\text{or} & (a, b, o) := a \lor b \leftrightarrow o \\
\text{xor} & (a, b, o) := a \land \neg b \lor \neg a \land b \leftrightarrow o
\end{align*}
\]

Combine circuits with \(\land\) and \(\exists\)

\[
C(x, y, q, r, s) := \exists w \left( C_1(x, y, w, q) \land C_2(y, w, r, s) \right)
\]

Family of adders: structural model

- General case \((n\text{-bit ripple-carry adder})\):
  1. wire together \(n\) full adders where \(i\)th carry-out is \((i + 1)\)st carry-in
  2. first carry is \(cin\) and last carry is \(cout\)
- In WS1S:
  \[
  \text{adder}(n, A, B, S, \text{cin}, \text{cout}) := \\
  \exists C \left( \forall x < n \rightarrow \text{full_adder}(A(x), B(x), C(x), S(x), C(x + 1)) \right) \land \\
  \left( C(0) \leftrightarrow \text{cin} \land \left( C(n) \leftrightarrow \text{cout} \right) \right)
  \]

Modeling with Boolean logic: a full adder

<table>
<thead>
<tr>
<th>(a)</th>
<th>(b)</th>
<th>(\text{cin})</th>
<th>(o)</th>
<th>(\text{cout})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\text{spec}(a, b, \text{cin}, o, \text{cout}) := \\
\left( o \leftrightarrow (-a \land b \land \text{cin}) \lor \ldots \right) \land \\
\left( \text{cout} \leftrightarrow (-a \land -b \land \text{cin}) \lor \ldots \right)
\]

\[
\text{full_adder}(a, b, \text{cin}, o, \text{cout}) := \\
\exists w_1 \exists w_2 \exists w_3 \left( \text{xor}(a, b, w_1) \land \text{xor}(w_1, \text{cin}, o) \land \\
\text{and}(a, b, w_2) \land \text{and}(\text{cin}, w_1, w_3) \land \\
\text{or}(w_3, w_2, \text{cout}) \right)
\]

Correctness

\[
\text{spec}(a, b, \text{cin}, o, \text{cout}) \leftrightarrow \text{full_adder}(a, b, \text{cin}, o, \text{cout})
\]

Addition in WS1S

Behavioral Specification

\[
\text{val}(n, S) + 2^n \cdot \text{vl} = \text{val}(cin) + \text{val}(n, A) + \text{val}(n, B)
\]

Encode functions as relations

\[
\text{mod}2(a, b, c, d) := a \leftrightarrow b \leftrightarrow c \leftrightarrow d
\]

\[
\text{atLeast}2(a, b, c, d) := d \leftrightarrow (a \land b) \lor (b \land c) \lor (a \land c)
\]

\[
\text{add}(A, B, S) := \exists C \left( \neg C(0) \land \forall p \left( \text{mod}2(A(p), B(p), C(p), S(p)) \land \\
\text{atLeast}2(A(p), B(p), C(p), C(p + 1)) \right) \right)
\]

\[
\text{val}(n, X, Y) := \forall p \left( Y(p) \rightarrow p < n \land X(p) \right)
\]

\[
\text{powof}(n, b, X) := \forall p \left( X(p) \rightarrow p = n \land b \right)
\]

Encode behavioral specification

\[
\text{adder}_\text{beh}(n, A, B, S, \text{cin}, \text{cout}) := \exists S' \exists C \exists C1 \exists A' \exists B' \exists X \exists Y \exists Z \\
\left( \text{val}(n, S, S') \land \text{powof}(n, \text{cout}, CO) \land \text{add}(S', CO, X) \land \\
\text{powof}(0, \text{cin}, CI) \land \text{val}(n, A, A') \land \text{val}(n, B, B') \land \\
\text{add}(CI, A', Y) \land \text{add}(Y, B', Z) \land X = Z \right)
\]
**Verification**

Equivalence of structural model and behavioral model
\[
\forall n \forall A \forall B \forall S \forall c \forall y \,(\text{adder}(n, A, B, S, c, y) \iff
\text{adder}_{\text{beh}}(n, A, B, S, c, y))
\]

Functional behavior
\[
\forall n \forall A \forall B \forall c \forall x \,(\forall x(A(x) \to x < n) \land \forall x(B(x) \to x < n) \implies
\exists S \exists c \forall x(S(x) \to x < n) \land \text{adder}(n, A, B, S, c, x) \land
\forall S' \forall c'(\forall x(S'(x) \to x < n) \land \text{adder}(n, A, B, S', c', x) \implies S = S' \land (c = c'))
\]

Algebraic properties, e.g., commutativity
\[
\forall n \forall A \forall B \forall S \forall c \forall x \,(\text{adder}(n, A, B, S, c, x) \iff \text{adder}(n, B, A, S, c, x))
\]

**Notice.** Induction built in!

---

**Exercise**

Exercise. Decide PA over \((Z, <, +, 0, 1)\) using a WS1S encoding.

Exercise. Demonstrate that, in the language of PA, there is no quantifier-free formula with variables in \(\{y\}\) equivalent to \(\exists x. 2 \cdot x = y\). \(\sim\) PA does not admit quantifier elimination.

Skolem's QE procedure particularly simple. Works relative to the augmented language containing rational multipliers \(q^*\) for all \(q \in Q\) and the floor function \([\cdot]\). Its main step is to eliminate bound variables in the scope of \([\cdot]\); e.g.
\[
\exists x. \left(\frac{2}{3} \cdot \frac{1}{5} + \left(\frac{1}{2} \cdot \frac{1}{4}\right)\right) > 15
\]

Exercise. Spell out the details of Skolem's QE procedure.

Exercise. Is \(\text{times}(X, Y, Z)\) expressible in WS1S?

---

**Presburger Arithmetic**

PA is first-order logic over the language \((N, <, +, 0, 1)\).

**Example.** \(\forall x. \exists y. y + y = x \lor y + y + 1 = x\)

PA decidable (at least nondeterministic \(2^{2^n}\)).

Quantifier elimination procedures due to Presburger, Skolem, and Cooper (deterministic \(2^{2^{2^n}}\)).

All relations and functions of PA definable in WS1S.

Encoding in WS1S by replacing PA variables with MSO variables and replacing \(x + y\) with existentially-bound \(Z\) together with constraint \(\text{add}(X, Y, Z)\).

Experimental results show that automata-based decision procedures compete with other decision procedures. *Often they are faster...*

Although the known upper bound is worse...
Automata-Based Decision Procedure for WS1S

Theorem. (Büchi, Elgot, Trakhtenbrot)

For every formula $\psi(X_1, \ldots, X_n)$, we can construct a DFA $A_\psi$ with $L(A_\psi) = L(\psi)$

Decision Procedure. For $\varphi(x_1, \ldots, x_m, X_1, \ldots, X_n)$
1. Eliminate FO variables in $\varphi \leadsto \psi(x_1, \ldots, x_m, X_1, \ldots, X_n)$
2. Construct DFA $A_\psi$ accepting $w$ iff $I_w \models \psi$
3. Output
   - "valid" if $L(A_{\text{Sing}(\widehat{x}_1) \land \cdots \land \text{Sing}(\widehat{x}_m) \rightarrow \varphi}) = \{0,1\}^{m+n}^*$
   - "unsatisfiable" if $L(A_{\text{Sing}(\widehat{x}_1) \land \cdots \land \text{Sing}(\widehat{x}_m) \land \psi}) = \emptyset$
   - otherwise: words $w, w' \in \{0,1\}^{m+n}^*$ with
     - $w \in L(A_{\text{Sing}(\widehat{x}_1) \land \cdots \land \text{Sing}(\widehat{x}_m) \rightarrow \varphi})$ and $w' \notin L(A_{\text{Sing}(\widehat{x}_1) \land \cdots \land \text{Sing}(\widehat{x}_m) \land \psi})$

Proof of the theorem

Construct $A_\varphi$ recursively with $L(A_\varphi) = L(\varphi)$

Base case. $\psi = \text{Succ}(X, Y)$:

To prove: $L(A_{\text{Succ}(X, Y)}) = L(\text{Succ}(X, Y))$

Base case. $\psi = X \subseteq Y$:

Negation

$\neg \psi$: complementing $A_\psi$

Correctness:

$L(\neg \psi) = \overline{L(\psi)} = L(A_\neg \psi) = L(A_\psi)$

Complexity: linear

- By induction hypothesis, $A_\psi$ is deterministic

- Complementing $A_\psi$ can be done by flipping final and non-final states (assuming that $A_\psi$ is complete)

Conjunction

Step case. $\varphi \land \psi$: product construction of $A_\varphi$ and $A_\psi$

Correctness:

w.l.o.g. assume that the free variables of $\varphi$ and $\psi$ are $X_1, \ldots, X_n$

$L(\varphi \land \psi) = L(\varphi) \cap L(\psi) = L(A_\varphi) \cap L(A_\psi) = L(A_{\varphi \land \psi})$

Complexity: $O(m \cdot n)$

where $m$ is the size of $A_\varphi$ and $n$ is the size of $A_\psi$
Existential quantification $\exists X \psi$

Intuition: automaton guesses the interpretation for $X$

Try projection of the $X$-track in $A_\psi$

$X = 0$

$Y = 1$

$X = 0$

$Y = 0$

Projection of the $X$-track in $A_\psi$ does not do the job!

$\psi = X(1) \land Y \not\subseteq X$

does not accept $\lambda$, 0, and 1

Projection & “making states accepting if reachable by 0-paddings”

Complexity of the decision procedure

Quantifier alternation yields exponential blow-ups!

$$\forall X \exists Y \varphi \sim \neg \exists X \neg \exists Y \varphi$$

If $|A_\varphi| = n$ then $|A_{\neg \exists Y \varphi}| \leq 2^n$ and $|A_{\neg \exists X \neg \exists Y \varphi}| \leq 2^{2^n}$

Is the worst case really that bad? Yes

There is a family of formulas $(\varphi_n)_{n \geq 1}$ with $A_{\varphi_n}$ needs at least

$$|A_{\varphi_n}| \geq 2^{2^{2^{n-1}}}, \text{ tower of height } n - 1$$

states.

Is there a better decision procedure than the automata-based one? No

since WS1S is only non-elementary decidable.

Existential quantification (cont.)

Right quotient of $L \subseteq \Sigma^*$ with $L' \subseteq \Sigma^*$

$L / L' := \{ w \in \Sigma^* \mid \text{there is a } u \in L' \text{ with } wu \in L \}$

Correctness:

- Assume that the formula is $\exists X_1 \psi(X_1, \ldots, X_n)$

- $\pi$ means “delete $X_1$-track” in a word

$$\pi : (\{0,1\}^n)^* \to (\{0,1\}^{n-1})^* \text{ given by } \pi(\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}) := \begin{pmatrix} b_2 \\ \vdots \\ b_n \end{pmatrix}$$

$$L(\exists X_1 \psi) = \pi(L(\psi)) / (\{0\}^{n-1})^* = \pi(L(A_\psi)) / (\{0\}^{n-1})^* = L(A_{\exists X_1 \psi})$$

Complexity: exponential (result has to be deterministic)

Regular languages and WS1S

Any WS1S formula describes a regular language

Does the converse also hold?

$$L \subseteq \{0,1\}^n \text{ regular } \rightarrow \text{ there is a formula } \varphi(X_1, \ldots, X_n) \text{ with } L(\varphi) = L$$

For a WS1S formula $\varphi(X)$ it holds

$$w \in L(\varphi) \Rightarrow w0\ldots0 \in L(\varphi)$$

- Reason: $w$ and $w0\ldots0$ encode the same interpretation

- Make encoding unique by a parameter for $|w|$.

Theorem. For a regular language $L \subseteq (\{0,1\}^n)^*$ there is a formula $\varphi(\$, X_1, \ldots, X_n)$ with $w \in L$ iff $I_w[|w|/\$] = \varphi$. 

Idea: describe functioning of an NFA as WS1S formula

For NFA $A = (Q, \{0,1\}^n, q_0, \delta, F)$ with $Q = \{1, \ldots, s\}$, let $\varphi_A(\$, X_1, \ldots, X_n)$ be the formula

$$\exists Y_1 \ldots \exists Y_s (Y_{q_1}(0) \land \\
\forall x(x \leq \$ \to (\forall q \in Q Y_q(x))) \land \\
\land_{1 \leq p < q \leq s} \forall x(x \leq \$ \to \neg(Y_p(x) \wedge Y_q(x))) \land \\
\land_{p \in Q} \forall x(x < \$ \to \neg Y_p(x) \to \Delta_p) \land \\
\land_{p \in Q} (Y_p(\$) \to \Delta'_p))$$

Altogether:

$$\$ > 0 \to \varphi_A(\$, X_1, \ldots, X_n) \land \$ = 0 \to (\neg \exists x x = x)$$

$A$ makes a transition at $x < \$ from state $p$

$$\Delta_p :\to (\neg X_1(x) \land \ldots \land \neg X_n(x) \to \bigvee_{q \in \delta(p)} \begin{pmatrix} 1 \\ \vdots \\ 0 \end{pmatrix} Y_q(x + 1))$$

$$\land \ldots \land (X_1(x) \land \ldots \land X_n(x) \to \bigvee_{q \in \delta(p)} \begin{pmatrix} 1 \\ \vdots \\ 0 \end{pmatrix} Y_q(x + 1))$$

$A$ can make a transition at $\$ from $p$ to some final state

$$\Delta'_p :\to \neg X_1(\$) \land \ldots \land \neg X_n(\$) \land \\
\forall x x = x \text{ if } \delta(p, \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}) \cap F \neq \emptyset, \\
\exists x x \neq x \text{ otherwise}$$

$$\land \ldots \land (X_1(\$) \land \ldots \land X_n(\$) \land \\
\forall x x = x \text{ if } \delta(p, \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}) \cap F \neq \emptyset, \\
\exists x x \neq x \text{ otherwise}$$

**Corollary.** Every WS1S formula is equivalent to a WS1S formula with top-level existential quantification only.

**Summary**

Automata-theoretic decision procedure for WS1S.

Nonelementary worst-case complexity.

Mona uses BDDs for representing DFA transition relations.

$\sim$ Often "good" run times in practice.

Direct automata-theoretic constructions often yield better worst-case complexities for certain subproblems.

Open: triple exponential automata-theoretic procedure for Presburger arithmetic

Logic-automaton connection extends to other classes of automaton (e.g. Büchi automata, Tree automata)

Characterization of complexity classes (e.g. a language is in NP iff definable in existential second-order logic).