Little Engines of Proof: Lecture 18

N. Shankar, L. de Moura, H. Ruess, A. Tiwari
shankar@csl.sri.com
URL: http://www.csl.sri.com/~shankar/LEP.html

(Based on lectures by Felix Klaedtke, Universität Freiburg)

Computer Science Laboratory
SRI International
Menlo Park, CA

The Logic-Automaton Connection

Correspondence discovered in 50s/60s

Automata	Languages	Logics
DFA, NFA	Regular languages	WS1S
Büchi automata	ω-regular languages	S1S
Tree automata	Regular tree languages	(W)S2S

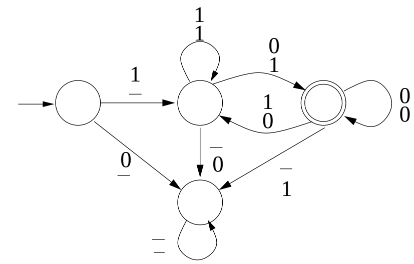
Applications. Circuit Verification, integer and real arithmetic, queues, model checking, pointer verification, XML queries, . . .

The Logic-Automaton Connection (cont.)

$$X(0) \wedge \forall p\big(X(p) \rightarrow \exists q(succ(p,q) \wedge Y(q))\big)$$

$$\Leftrightarrow \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \ldots \right\}$$

$\Leftrightarrow$



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.

$$\varphi ::= succ(p,q) \;\big|\; X(p) \;\big|\; \neg\varphi \;\big|\; \varphi \wedge \varphi \;\big|\; \exists p\varphi \;\big|\; \exists X\varphi$$

WS1S semantics

Structure. Natural numbers N with successor relation

Interpretation.

- $I: \ p \longmapsto n \in N$
- $I: \ X \longmapsto N \in \mathcal{F}(N)$

Truth value. (of a formula wrt interpretation I)

$I \models Y(x)$	iff	$I(x) \in I(Y)$
$I \models succ(x,y)$	iff	$I(x) + 1 = I(y)$
$I \models \neg\varphi$	iff	$I \not\models \varphi$
$I \models \varphi \wedge \psi$	iff	$I \models \varphi$ and $I \models \psi$
$I \models \exists x \varphi$	iff	$I[n/x] \models \varphi,$ for some $n \in N$
$I \models \exists X \varphi$	iff	$I[N/X] \models \varphi,$ for some $N \in \mathcal{F}(N)$

Validity.

$\models \varphi$ iff $I \models \varphi$, for all interpretations I

Syntactic sugar

Standard connectives and quantifiers

$\varphi \vee \psi$	for	$\neg(\neg\varphi \wedge \neg\psi)$
$\forall x \varphi$	for	$\neg\exists x \neg\varphi$
$\vdots$		$\vdots$

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

$x = 0$	$:=$	$\neg\exists z\, succ(z,x)$
$x = y$	$:=$	$\forall Z\big(Z(x) \leftrightarrow Z(y)\big)$
$x = y + n$	$:=$	$\exists z_0 \ldots \exists z_n \big(z_0 = y \wedge x = z_n \wedge \bigwedge_{0 \le i < n} succ(z_i, z_{i+1})\big)$
$X(x+n)$	$:=$	$\exists z\big(z = x + n \wedge X(z)\big)$
$x \le y$	$:=$	$\forall U\big(U(y) \wedge \forall z\big(U(z+1) \to U(z)\big) \to U(x)\big)$
$x < y$	$:=$	$x \le y \wedge \neg x = y$
$\vdots$		$\vdots$

Syntactic Sugar (cont.)

Intersection.

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

Subset.

$$Y \subseteq X \ := \ \forall x(Y(x) \to X(x))$$

Set equality.

$$Y = X \ := \ Y \subseteq X \wedge X \subseteq Y$$

Emptiness.

$$X = \emptyset \ := \ \forall Y(Y \subseteq X \to Y = X)$$

Singleton.

$$Sing(X) \ := \ X \neq \emptyset \wedge \forall Y(Y \subseteq X \to (Y = X \vee Y = \emptyset))$$

Minimal syntax

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

$$\psi ::= X \subseteq Y \mid Succ(X,Y) \mid \neg\psi \mid \psi \wedge \psi \mid \exists X \psi$$

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

Introduce for each FO variable x a fresh MSO variable $\widehat{x}$

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

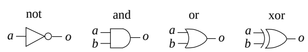
$$\forall x \exists y\big(succ(x,y) \wedge Z(y)\big) \longmapsto$$
$$\forall \widehat{x}\big(Sing(\widehat{x}) \to \exists \widehat{y}\big(Sing(\widehat{y}) \wedge Succ(\widehat{x},\widehat{y}) \wedge \widehat{y} \subseteq Z\big)\big)$$

Circuits in WS1S

Encode quantified Boolean logic in WS1S

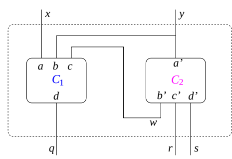
$$\forall x \exists y (x \leftrightarrow y) \quad \rightsquigarrow \quad \forall X \exists Y \big(X(0) \leftrightarrow Y(0) \big)$$

Logical gates as Boolean relations



$$\begin{array}{llllll} not(a,o) & := & \neg a \leftrightarrow o & and(a,b,o) & := & a \wedge b \leftrightarrow o \\ or(a,b,o) & := & a \vee b \leftrightarrow o & xor(a,b,o) & := & a \wedge \neg b \vee \neg a \wedge b \leftrightarrow o \end{array}$$

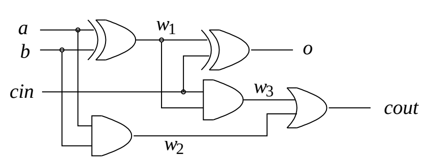
Combine circuits with $\wedge$ and $\exists$



$$C(x,y,q,r,s) := \exists w \big(C_1(x,y,w,q) \wedge C_2(y,w,r,s) \big)$$

Modeling with Boolean logic: a full adder

a	b	cin	o	$cout$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



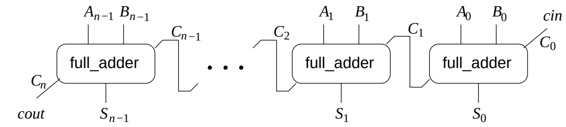
$$\begin{array}{l} spec(a,b,cin,o,cout) := \\ \quad \big(o \leftrightarrow (\neg a \wedge b \wedge cin) \vee \ldots \big) \wedge \\ \quad \big(cout \leftrightarrow (\neg a \wedge \neg b \wedge cin) \vee \ldots \big) \end{array}$$

$$\begin{array}{l} full\_adder(a,b,cin,o,cout) := \\ \quad \exists w_1 \exists w_2 \exists w_3 \big(xor(a,b,w_1) \wedge xor(w_1,cin,o) \wedge \\ \qquad and(a,b,w_2) \wedge and(cin,w_1,w_3) \wedge \\ \qquad or(w_3,w_2,cout) \big) \end{array}$$

Correctness

$$spec(a,b,cin,o,cout) \leftrightarrow full\_adder(a,b,cin,o,cout)$$

Family of adders: structural model



- General case (n-bit ripple-carry adder):
  1. wire together n full adders where ith carry-out is $(i+1)$st carry-in
  2. first carry is cin and last carry is $cout$
- In WS1S:

$$\begin{array}{l} adder(n,A,B,S,cin,cout) := \\ \quad \exists C \Big(\forall x \big(x < n \rightarrow full\_adder(A(x),B(x),C(x),S(x),C(x+1)) \big) \wedge \\ \qquad \big(C(0) \leftrightarrow cin \big) \wedge \big(C(n) \leftrightarrow cout \big) \Big) \end{array}$$

Addition in WS1S

Behavioral Specification

$$val(n,S) + 2^n * vl(cout) = vl(cin) + val(n,A) + val(n,B)$$

Encode functions as relations

$$\begin{array}{rcl} mod2(a,b,c,d) & := & a \leftrightarrow b \leftrightarrow c \leftrightarrow d \\ atLeast2(a,b,c,d) & := & d \leftrightarrow (a \wedge b) \vee (b \wedge c) \vee (a \wedge c) \\ add(A,B,S) & := & \exists C \big(\neg C(0) \wedge \forall p \big(mod2(A(p),B(p),C(p),S(p)) \wedge \\ & & \qquad atLeast2(A(p),B(p),C(p),C(p+1)) \big) \big) \\ val(n,X,Y) & := & \forall p \big(Y(p) \leftrightarrow p < n \wedge X(p) \big) \\ powof2(n,b,X) & := & \forall p \big(X(p) \leftrightarrow p = n \wedge b \big) \end{array}$$

Encode behavioral specification

$$\begin{array}{l} adder\_beh(n,A,B,S,cin,cout) := \exists S' \exists CO \exists CI \exists A' \exists B' \exists X \exists Y \exists Z \\ \big(val(n,S,S') \wedge powof2(n,cout,CO) \wedge add(S',CO,X) \wedge \\ \quad powof2(0,cin,CI) \wedge val(n,A,A') \wedge val(n,B,B') \wedge \\ \quad add(CI,A',Y) \wedge add(Y,B',Z) \wedge X = Z \big) \end{array}$$

Verification

Equivalence of structural model and behavioral model

$$\forall n \forall A \forall B \forall S \forall ci \forall co \Big(adder(n, A, B, S, ci, co) \leftrightarrow adder\_beh(n, A, B, S, ci, co) \Big)$$

Functional behavior

$$\forall n \forall A \forall B \forall ci \Bigg(\forall x \Big(A(x) \rightarrow x < n \Big) \wedge \forall x \Big(B(x) \rightarrow x < n \Big) \rightarrow \exists S \exists co \Big(\forall x \Big(S(x) \rightarrow x < n \Big) \wedge adder(n, A, B, S, ci, co) \wedge \forall S' \forall co' \Big(\forall x \Big(S'(x) \rightarrow x < n \Big) \wedge adder(n, A, B, S', ci, co') \rightarrow S = S' \wedge (co \leftrightarrow co') \Big) \Big) \Bigg)$$

Algebraic properties, e.g., commutativity

$$\forall n \forall A \forall B \forall S \forall ci \forall co \Big(adder(n, A, B, S, ci, co) \leftrightarrow adder(n, B, A, S, ci, co) \Big)$$

Notice. Induction built in!

Presburger Arithmetic

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

Example. $\forall x.\ \exists y.\ y + y = x \vee y + y + 1 = x$

PA decidable (at least nondeterministic $2^{2^{cn}}$).

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

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 $add(X, Y, Z)$.

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

Although the known upper bound is worse...

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 $\exists x.\ 2 * x = y$. $\rightsquigarrow$ 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 $[.]$. Its main step is to eliminate bound variables in the scope of $[.]$; e.g.

$$\exists x.\ (\frac{2}{3}[\frac{1}{5} + (\frac{1}{2}x + \frac{1}{4})] > 15)$$

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

Exercise. Is $times(X, Y, Z)$ expressible in WS1S?

Words as Interpretations

Word $w \in \{0,1\}^*$ induces interpretation for a variable

$$01 \quad \rightsquigarrow \quad I(X) = \{1\} \qquad\qquad 0101 \quad \rightsquigarrow \quad I(X) = \{1, 3\}$$

Word $w \in (\{0,1\}^n)^*$ induces interpretation I_w for n variables

$$w = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \cdots \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \rightsquigarrow \begin{array}{lcl} I_w(X_1) & = & \{1\} \\ I_w(X_2) & = & \{1, 3\} \\ I_w(X_3) & = & \emptyset \end{array}$$

Language of formula $\psi(X_1, \ldots, X_n)$

$$L(\psi) := \{\, w \in (\{0,1\}^n)^* \mid I_w \models \psi \,\}$$

Notice: $w \in L(\psi)$ iff $w0^n \ldots 0^n \in L(\psi)$.

Automata-Based Decision Procedure for WS1S

Theorem. (Büchi, Elgot, Trakhtenbrot)

For every formula $\psi(X_1, \ldots, X_n)$, we can construct a DFA $\mathcal{A}_\psi$ with $L(\mathcal{A}_\psi) = L(\psi)$

Decision Procecure. For $\varphi(x_1, \ldots, x_m, X_1, \ldots, X_n)$

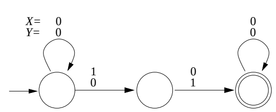
1. Eliminate FO variables in $\varphi \;\rightsquigarrow\; \psi(\widehat{x}_1, \ldots, \widehat{x}_m, X_1, \ldots, X_n)$
2. Construct DFA $\mathcal{A}_\psi$ accepting w iff $I_w \models \psi$
3. Output
   - "valid" if $L(\mathcal{A}_{Sing(\widehat{x}_1) \wedge \ldots \wedge Sing(\widehat{x}_m) \rightarrow \psi}) = (\{0,1\}^{m+n})^*$
   - "unsatisfiable" if $L(\mathcal{A}_{Sing(\widehat{x}_1) \wedge \ldots \wedge Sing(\widehat{x}_m) \wedge \psi}) = \emptyset$
   - otherwise: words $w, w' \in (\{0,1\}^{m+n})^*$ with

     $w \in \underbrace{L(\mathcal{A}_{Sing(\widehat{x}_1) \wedge \ldots \wedge Sing(\widehat{x}_m) \rightarrow \psi})}_{\text{satisfying models}}$ and $w' \notin \underbrace{L(\mathcal{A}_{Sing(\widehat{x}_1) \wedge \ldots \wedge Sing(\widehat{x}_m) \wedge \psi})}_{\text{counter models}}$

Proof of the theorem

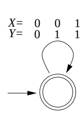
Construct $\mathcal{A}_\psi$ recursively with $L(\mathcal{A}_\psi) = L(\psi)$

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



To prove: $L(\mathcal{A}_{Succ(X,Y)}) = L(Succ(X, Y))$

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



Negation

$\neg\psi$: complementing $\mathcal{A}_\psi$

Correctness:

$$L(\neg\psi) = \overline{L(\psi)} = \overline{L(\mathcal{A}_\psi)} = L(\mathcal{A}_{\neg\psi})$$

Complexity: linear

- By induction hypothesis, $\mathcal{A}_\psi$ is *deterministic*
- Complementing $\mathcal{A}_\psi$ can by done by flipping final and non-final states (assuming that $\mathcal{A}_\psi$ is complete)

Conjunction

Step case. $\varphi \wedge \psi$: product construction of $\mathcal{A}_\varphi$ and $\mathcal{A}_\psi$

Correctness:

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

$$L(\varphi \wedge \psi) = L(\varphi) \cap L(\psi) = L(\mathcal{A}_\varphi) \cap L(\mathcal{A}_\psi) = L(\mathcal{A}_{\varphi \wedge \psi})$$

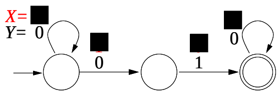
Complexity: $\mathcal{O}(m \cdot n)$

where m is the size of $\mathcal{A}_\varphi$ and n is the size of $\mathcal{A}_\psi$

Existential quantification $\exists X \psi$

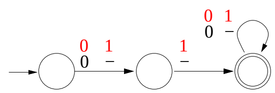
Intuition: automaton guesses the interpretation for X

Try projection of the X-track in $\mathcal{A}_\psi$

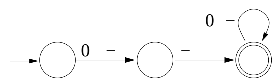


$\exists X$ $\quad$ $Succ(X, Y)$

Projection of the X-track in $\mathcal{A}_\psi$ does *not* do the job!



$\psi = X(1) \wedge Y \subseteq X$



does not accept λ, 0, and 1

Projection & "making states accepting if reachable by 0-paddings"

Existential quantification (cont.)

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

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

Correctness:

- Assume that the formula is $\exists X_1 \psi(X_1, \dots, X_n)$
- π means "delete X_1-track" in a word

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

$$\begin{aligned} L(\exists X_1 \psi) &= \pi(L(\psi)) \big/ (\{0\}^{n-1})^* = \pi(L(\mathcal{A}_\psi)) \big/ (\{0\}^{n-1})^* \\ &= L(\mathcal{A}_{\exists X_1 \psi}) \end{aligned}$$

Complexity: exponential (result has to be *deterministic*)

Complexity of the decision procedure

Quantifier alternation yields exponential blow-ups!

$$\forall X \exists Y \varphi \quad \rightsquigarrow \quad \neg \exists X \neg \exists Y \varphi$$

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

Is the worst case really that bad? Yes

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

$$\|\mathcal{A}_{\varphi_n}\| \geq 2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}} \Big\} \text{ tower of height } n-1$$

states.

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

since WS1S is only non-elementary decidable.

Regular languages and WS1S

Any WS1S formula describes a regular language

Does the converse also hold?

$L \subseteq \{0,1\}^n$ regular $\overset{???}{\Rightarrow}$ there is a formula $\varphi(X_1, \dots, X_n)$ with $L(\varphi) = L$

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

$$w \in L(\varphi) \quad \Rightarrow \quad w0\dots0 \in L(\varphi)$$

- Reason: w and $w0\dots0$ 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 $\varphi(\$, X_1, \dots, X_n)$ with $w \in L$ iff $I_w[|w|/\$] \models \varphi$.

Regular languages and WS1S (Cont.)

Idea: describe functioning of an NFA as WS1S formula

For NFA $\mathcal{A} = (Q, \{0,1\}^n, q_{\mathrm{I}}, \delta, F)$ with $Q = \{1, \ldots, s\}$, let $\varphi_{\mathcal{A}}(\$, X_1, \ldots, X_n)$ be the formula

$$\begin{aligned}
\exists Y_1 \ldots \exists Y_s \Big(& Y_{q_{\mathrm{I}}}(0) \wedge \\
& \forall x (x \leq \$ \rightarrow \big(\textstyle\bigvee_{q \in Q} Y_q(x)\big)) \wedge \\
& \textstyle\bigwedge_{1 \leq p < q \leq s} \forall x (x \leq \$ \rightarrow \neg\big(Y_p(x) \wedge Y_q(x)\big)) \wedge \\
& \textstyle\bigwedge_{p \in Q} \forall x \big(x < \$ \wedge Y_p(x) \rightarrow \varDelta_p\big) \wedge \\
& \textstyle\bigwedge_{p \in Q} \big(Y_p(\$) \rightarrow \varDelta'_p\big)\Big)
\end{aligned}$$

Altogether:

$$\$ > 0 \rightarrow \varphi_{\mathcal{A}}(\$, X_1, \ldots, X_n) \wedge \$ = 0 \rightarrow (\neg \exists x x = x)$$

Regular languages and WS1S (Cont.)

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

$$\begin{aligned}
\varDelta_p \quad :\leftrightarrow \quad & \Big(\neg X_1(x) \wedge \ldots \wedge \neg X_n(x) \rightarrow \bigvee_{q \in \delta(p, \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix})} Y_q(x+1)\Big) \\
& \wedge \ldots \wedge \\
& \Big(X_1(x) \wedge \ldots \wedge X_n(x) \rightarrow \bigvee_{q \in \delta(p, \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix})} Y_q(x+1)\Big)
\end{aligned}$$

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

$$\begin{aligned}
\varDelta'_p \quad :\leftrightarrow \quad & \neg X_1(\$) \wedge \ldots \wedge \neg X_n(\$) \wedge \begin{cases} \forall x x = x & \text{if } \delta(p, \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}) \cap F \neq \emptyset \\ \exists x x \neq x & \text{otherwise} \end{cases} \\
& \vee \ldots \vee \\
& X_1(\$) \wedge \ldots \wedge X_n(\$) \wedge \begin{cases} \forall x x = x & \text{if } \delta(p, \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}) \cap F \neq \emptyset \\ \exists x x \neq x & \text{otherwise} \end{cases}
\end{aligned}$$

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.

$\rightsquigarrow$ 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).