Logic, Automata, Games, and Algorithms

Moshe Y. Vardi

Rice University

Two Separate Paradigms in Mathematical Logic

- **Paradigm I**: *Logic* – declarative formalism
  - Specify properties of mathematical objects, e.g., $(\forall x, y, x)(mult(x, y, z) \leftrightarrow mult(y, x, z))$ – commutativity.
- **Paradigm II**: *Machines* – imperative formalism
  - Specify computations, e.g., Turing machines, finite-state machines, etc.

Surprising Phenomenon: Intimate connection between logic and machines – *topic of this talk*.

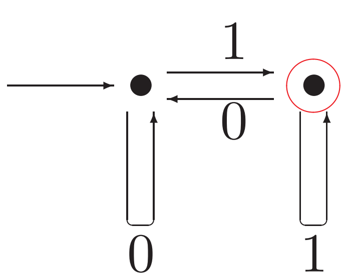
Nondeterministic Finite Automata

$A = (\Sigma, S, S_0, \rho, F)$

- *Alphabet*: Σ
- *States*: S
- *Initial states*: $S_0 \subseteq S$
- *Nondeterministic transition function*:
$\rho : S \times \Sigma \to 2^S$
- *Accepting states*: $F \subseteq S$

Input word: $a_0, a_1, \ldots, a_{n-1}$
Run: $s_0, s_1, \ldots, s_n$
- $s_0 \in S_0$
- $s_{i+1} \in \rho(s_i, a_i)$ for $i \geq 0$

Acceptance: $s_n \in F$
Recognition: $L(A)$ – words accepted by A.

Example: 

– ends with 1's

Fact: NFAs define the class *Reg* of regular languages.

Logic of Finite Words

View finite word $w = a_0, \ldots, a_{n-1}$ over alphabet Σ as a mathematical structure:

- Domain: $0, \ldots, n-1$
- Binary relations: $<, \leq$
- Unary relations: $\{P_a : a \in \Sigma\}$

First-Order Logic (FO):

- Unary atomic formulas: $P_a(x)$ ($a \in \Sigma$)
- Binary atomic formulas: $x < y, x \leq y$

Example: $(\exists x)((\forall y)(\neg(x < y)) \wedge P_a(x))$ – last letter is a.

Monadic Second-Order Logic (MSO):

- Monadic second-order quantifier: $\exists Q$
- New unary atomic formulas: $Q(x)$

NFA vs. MSO

Theorem [Büchi, Elgot, Trakhtenbrot, 1957-8 (independently)]: MSO $\equiv$ NFA

- Both MSO and NFA define the class Reg.

Proof: Effective

- From NFA to MSO ($A \mapsto \varphi_A$)
  - Existence of run – existential monadic quantification
  - Proper transitions and acceptance - first-order formula
- From MSO to NFA ($\varphi \mapsto A_\varphi$): closure of NFAs under
  - *Union* – disjunction
  - *Projection* – existential quantification
  - *Complementation* – negation

NFA Complementation

Run Forest of A on w:

- Roots: elements of S_0.
- Children of s at level i: elements of $\rho(s, a_i)$.
- Rejection: no leaf is accepting.

Key Observation: collapse forest into a DAG – at most one copy of a state at a level; width of DAG is $|S|$.

Subset Construction Rabin-Scott, 1959:

- $A^c = (\Sigma, 2^S, \{S_0\}, \rho^c, F^c)$
- $F^c = \{T : T \cap F = \emptyset\}$
- $\rho^c(T, a) = \bigcup_{t \in T} \rho(t, a)$
- $L(A^c) = \Sigma^* - L(A)$

Complementation Blow-Up

$A = (\Sigma, S, S_0, \rho, F)$, $|S| = n$
$A^c = (\Sigma, 2^S, \{S_0\}, \rho^c, F^c)$

Blow-Up: 2^n upper bound

Can we do better?

Lower Bound: 2^n
Sakoda-Sipser 1978, Birget 1993

$L_n = (0+1)^* 1 (0+1)^{n-1} 0 (0+1)^*$

- L_n is easy for NFA
- $\overline{L_n}$ is hard for NFA

NFA Nonemptiness

Nonemptiness: $L(A) \neq \emptyset$

Nonemptiness Problem: Decide if given A is nonempty.

Directed Graph $G_A = (S, E)$ of NFA $A = (\Sigma, S, S_0, \rho, F)$:
- *Nodes*: S
- Edges: $E = \{(s,t) : t \in \rho(s,a)$ for some $a \in \Sigma\}$

Lemma: A is nonempty iff there is a path in G_A from S_0 to F.

- Decidable in time linear in size of A, using *breadth-first search* or *depth-first search*.

MSO Satisfiability – Finite Words

Satisfiability: $models(\psi) \neq \emptyset$

Satisfiability Problem: Decide if given ψ is satisfiable.

Lemma: ψ is satisfiable iff A_ψ is nonnempty.

Corollary: MSO satisfiability is decidable.

- Translate ψ to A_ψ.
- Check nonemptiness of A_ψ.

Complexity:

- *Upper Bound*: Nonelementary Growth

$$2^{\cdot^{\cdot^{\cdot^{2^n}}}}$$

(tower of height $O(n)$)

- *Lower Bound* [Stockmeyer, 1974]: Satisfiability of FO over finite words is nonelementary (no bounded-height tower).

Automata on Infinite Words

Büchi Automaton, 1962: $A = (\Sigma, S, S_0, \rho, F)$

- Σ: finite alphabet
- S: finite state set
- $S_0 \subseteq S$: initial state set
- $\rho : S \times \Sigma \rightarrow 2^S$: transition function
- $F \subseteq S$: accepting state set

Input: $w = a_0, a_1 \ldots$
Run: $r = s_0, s_1 \ldots$

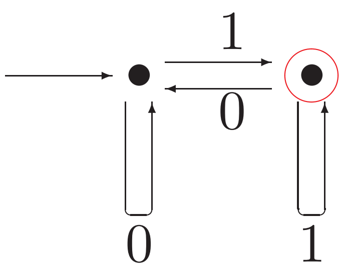
- $s_0 \in S_0$
- $s_{i+1} \in \rho(s_i, a_i)$

Acceptance: run visits F *infinitely often*.

Fact: NBAs define the class ω-*Reg* of ω-regular languages.

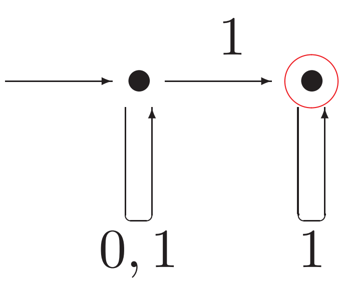
Examples

$((0+1)^*1)^\omega$:



– infinitely many 1’s

$(0+1)^*1^\omega$:



– finitely many 0’s

Logic of Infinite Words

View infinite word $w = a_0, a_1, \ldots$ over alphabet Σ as a mathematical structure:

- Domain: N
- Binary relations: $<, \leq$
- Unary relations: $\{P_a : a \in \Sigma\}$

First-Order Logic (FO):

- Unary atomic formulas: $P_a(x)$ ($a \in \Sigma$)
- Binary atomic formulas: $x < y, x \leq y$

Monadic Second-Order Logic (MSO):

- Monadic second-order quantifier: $\exists Q$
- New unary atomic formulas: $Q(x)$

Example: q holds at every even point.

$$
\begin{aligned}
(\exists Q)(\forall x)(\forall y)((((Q(x) \wedge y = x + 1) \rightarrow (\neg Q(y)))\wedge \\
(((\neg Q(x)) \wedge y = x + 1) \rightarrow Q(y)))\wedge \\
(x = 0 \rightarrow Q(x)) \wedge (Q(x) \rightarrow q(x))),
\end{aligned}
$$

NBA vs. MSO

Theorem [Büchi, 1962]: MSO $\equiv$ NBA

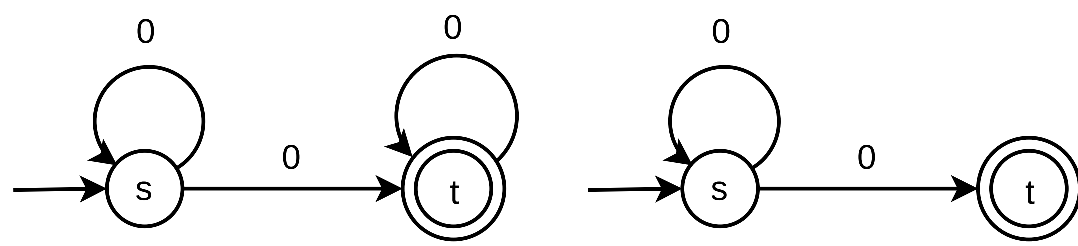
- Both MSO and NBA define the class ω-Reg.

Proof: Effective

- From NBA to MSO ($A \mapsto \varphi_A$)
  - Existence of run – existential monadic quantification
  - Proper transitions and acceptance - first-order formula
- From MSO to NBA ($\varphi \mapsto A_\varphi$): closure of NBAs under
  - *Union* – disjunction
  - *Projection* - existential quantification
  - *Complementation* - negation

Büchi Complementation

Problem: subset construction fails!



History

- Büchi'62: doubly exponential construction.
- SVW'85: 16^{n^2} upper bound
- Saf'88: n^{2n} upper bound
- Mic'88: $(n/e)^n$ lower bound
- KV'97: $(6n)^n$ upper bound
- FKV'04: $(0.97n)^n$ upper bound
- Yan'06: $(0.76n)^n$ lower bound
- Schewe'09: $(0.76n)^n$ upper bound

NBA Nonemptiness

Nonemptiness: $L(A) \neq \emptyset$

Nonemptiness Problem: Decide if given A is nonempty.

Directed Graph $G_A = (S, E)$ of NBA $A = (\Sigma, S, S_0, \rho, F)$:
- *Nodes*: S
- Edges: $E = \{(s,t) : t \in \rho(s,a)$ for some $a \in \Sigma\}$

Lemma: A is nonempty iff there is a path in G_A from S_0 to some $t \in F$ and from t to itself – *lasso*.

- Decidable in time linear in size of A, using *depth-first search* – analysis of cycles in graphs.

MSO Satisfiability – Infinite Words

Satisfiability: $models(\psi) \neq \emptyset$

Satisfiability Problem: Decide if given ψ is satisfiable.

> **Lemma**: ψ is satisfiable iff A_ψ is nonnempty.

Corollary: MSO satisfiability is decidable.

- Translate ψ to A_ψ.
- Check nonemptiness of A_ψ.

Complexity:

- *Upper Bound*: Nonelementary Growth

$$2^{\cdot^{\cdot^{\cdot^{2^{O(n \log n)}}}}}$$

(tower of height $O(n)$)

- *Lower Bound* [Stockmeyer, 1974]: Satisfiability of FO over infinite words is nonelementary (no bounded-height tower).

Logic and Automata for Infinite Trees

Labeled Infinite k-ary Tree: $\tau : \{0, \ldots, k-1\}^* \rightarrow \Sigma$

Tree Automata:

- Transition Function– $\rho : S \times \Sigma \rightarrow 2^{S^k}$

MSO for Trees:

- Atomic predicates: $E_1(x, y), \ldots, E_k(x, y)$

Theorem [Rabin, 1969]:
Tree MSO $\equiv$ Tree Automata

- Major difficulty: complementation.

Corollary: Decidability of satisfiability of MSO on trees – one of the most powerful decidability results in logic.

Standard technique during 1970s: Prove decidability via reduction to MSO on trees.

- *Nonelementary complexity*.

Temporal Logic

Prior, 1914–1969, Philosophical Preoccupations:

- *Religion*: Methodist, Presbytarian, atheist, agnostic
- *Ethics*: “Logic and The Basis of Ethics”, 1949
- *Free Will, Predestination, and Foreknowledge*:

– “The future is to some extent, even if it is only a very small extent, something we can make for ourselves”.

– “Of what will be, it has now been the case that it will be.”

– “There is a deity who infallibly knows the entire future.”

> Mary Prior: “I remember his waking me one night [in 1953], coming and sitting on my bed, . . ., and saying he thought one could make a formalised tense logic.”

- 1957: “Time and Modality”

The Temporal Logic of Programs

Precursors:

- Prior: “There are practical gains to be had from this study too, for example in the representation of time-delay in computer circuits”
- Rescher & Urquhart, 1971: applications to processes (“a programmed sequence of states, deterministic or stochastic”)

[Pnueli, 1977]:

- Future linear temporal logic (LTL) as a logic for the specification of non-terminating programs
- Temporal logic with “next” and “until”.

Programs as Labeled Graphs

Key Idea: Programs can be represented as transition systems (state machines)

Transition System: $M = (W, I, E, F, \pi)$

- W: states
- $I \subseteq W$: initial states
- $E \subseteq W \times W$: transition relation
- $F \subseteq W$: fair states
- $\pi : W \rightarrow Powerset(Prop)$: Observation function

Fairness: An assumption of "reasonableness" – restrict attention to computations that visit F infinitely often, e.g., "the channel will be up infinitely often".

Runs and Computations

Run: $w_0, w_1, w_2, \ldots$

- $w_0 \in I$
- $(w_i, w_{i+1}) \in E$ for $i = 0, 1, \ldots$

Computation: $\pi(w_0), \pi(w_1), \pi(w_2), \ldots$

- $L(M)$: set of computations of M

Verification: System M satisfies specification φ –

- all computations in $L(M)$ satisfy φ.

$\ldots$

$\ldots$

$\ldots$

Specifications

Specification: properties of computations.

Examples:

- “No two processes can be in the critical section at the same time.” – *safety*
- “Every request is eventually granted.” – *liveness*
- “Every continuous request is eventually granted.” – *liveness*
- “Every repeated request is eventually granted.” – *liveness*

Temporal Logic

Linear Temporal logic (LTL): logic of temporal sequences (Pnueli, 1977)

Main feature: time is implicit

- *next* φ: φ holds in the next state.
- *eventually* φ: φ holds eventually
- *always* φ: φ holds from now on
- φ *until* ψ: φ holds until ψ holds.

- $\pi, w \models \mathit{next}\ \varphi$ if w $\bullet \longrightarrow \underset{\varphi}{\bullet} \longrightarrow \quad \bullet \longrightarrow \bullet \longrightarrow \bullet \ldots$
- $\pi, w \models \varphi\ \mathit{until}\ \psi$ if w $\underset{\varphi}{\bullet} \longrightarrow \underset{\varphi}{\bullet} \longrightarrow \quad \underset{\varphi}{\bullet} \longrightarrow \underset{\psi}{\bullet} \longrightarrow \bullet \ldots$

Examples

- always not (CS_1 and CS_2): mutual exclusion (safety)
- always (Request implies eventually Grant): liveness
- always (Request implies (Request until Grant)): liveness
- always (always eventually Request) implies eventually Grant: liveness

Expressive Power

Gabbay, Pnueli, Shelah & Stavi, 1980: Propositional LTL has precisely the expressive power of FO over the naturals (builds on [Kamp, 1968]).

Easy Direction: LTL$\mapsto$FO

Example: φ is θ $until$ ψ
$FO(\varphi)(x)$:

$$(\exists y)(y > x \wedge FO(\psi)(y) \wedge (\forall z)((x \leq z < y) \rightarrow FO(\theta)(z))$$

Corollary: There is a translation of LTL to NBA via FO.

- But: Translation is nonelementary.

Elementary Translation

Theorem [V.&Wolper, 1983]: There is an exponential translation of LTL to NBA.
Corollary: There is an exponential algorithm for satisfiability in LTL.

Industrial Impact:

- Practical verification tools based on LTL.
- Widespread usage in industry.

Question: What is the key to efficient translation?
Answer: *Games*!

Alternating Automata

Alternating automata: 2-player games

Nondeterministic transition: $\rho(s, a) = t_1 \vee t_2 \vee t_3$

Alternating transition: $\rho(s, a) = (t_1 \wedge t_2) \vee t_3$
"either both t_1 and t_2 accept or t_3 accepts".

- $(s, a) \mapsto \{t_1, t_2\}$ or $(s, a) \mapsto \{t_3\}$
- $\{t_1, t_2\} \models \rho(s, a)$ and $\{t_3\} \models \rho(s, a)$

Alternating transition relation: $\rho : S \times \Sigma \rightarrow \mathcal{B}^+(S)$ (positive Boolean formulas over S)

Alternative Approach: *existential* and *universal* states [Chandra, Kozen &Stckmeyer, 1980]

Alternating Automata

Brzozowski&Leiss, 1980: Boolean automata

$A = (\Sigma, S, s_0, \rho, F)$
- Σ, S, $F \subseteq S$: as before
- $s_0 \in S$: initial state
- $\rho : S \times \Sigma \rightarrow \mathcal{B}^+(S)$: alternating transition function

Game:

- Board: $a_0, a_1 \ldots$
- Positions: $S \times N$
- Initial position: $(s_0, 0)$
- Automaton move at (s, i):
choose $T \subseteq S$ such that $T \models \rho(s, a_i)$
- Opponent's response:
move to $(t, i + 1)$ for some $t \in T$
- Automaton wins if play goes through infinitely many positions (s', i) with $s' \in F$

Acceptance: Automaton has a winning strategy.

Example

$A = (\{0,1\}, \{m,s\}, m, \rho, \{m\})$

- $\rho(m,1) = m$
- $\rho(m,0) = m \wedge s$
- $\rho(s,1) = \mathbf{true}$
- $\rho(s,0) = s$

Intuition:

- m is a master process. It launches s when it sees 0.
- s is a slave process. It wait for 1, and then terminates successfully.

$L(A) =$ infinitely many 1's.

Expressiveness

[Miyano&Hayashi, 1984]:

- Nondeterministic Büchi automata: ω-regular languages

- Alternating automata: ω-regular languages

What is the point?: Succinctness

Exponential gap:

- Exponential translation from alternating Büchi automata to nondeterministic Büchi automata

- In the worst case this is the best possible

- PSPACE nonemptiness test: go via nondeterministic automata.

Theorem[V., 1994] : For each LTL formula φ there is an alternating Büchi automaton A_φ with $||\varphi||$ states such that $models(\varphi) = L(A_\varphi)$.

Game Semantics for LTL

Background: game-semantics for FO, à la [Lorenzen, 1958] and [Hintikka, 1973] .

Game for LTL: Protagonist vs Antagonist

- Formula φ
- Infinite word $w = a_0, a_1, \ldots$
- Position (ψ, i) in $subformulas(\varphi) \times N$
- Initial position $(\varphi, 0)$

case
- ψ propositional: Protagonist wins iff ψ holds at a_i
- $\psi = \psi_1 \vee \psi_2$: Protagonist choses ψ_j and moves to (ψ_j, i)
- $\psi = \psi_1 \wedge \psi_2$: Antagonist choses ψ_j and moves to (ψ_j, i)
- $\psi = next\ \theta$: Protagonist moves to $(\theta, i+1)$
- $\psi = \theta\ until\ \chi$: Protagonist moves to (χ, i) or $(\theta \wedge (next\ \psi), i)$

esac.

Crucial Idea: Alternating automata capture game semantics

LTL to to Alternating Büchi Automata

Input formula: φ

- $subf(\varphi)$: subformulas of φ
- $nonU(\varphi)$: non-Until subformulas of φ

Alternating Büchi Automaton:

$A_\varphi = \{2^{Prop}, subf(\varphi), \varphi, \rho, nonU(\varphi)\}$:

- $\rho(p, a) =$ **true** if $p \in a$,
- $\rho(p, a) =$ **false** if $p \notin a$,
- $\rho(\xi \wedge \psi, a) = \rho(\xi, a) \wedge \rho(\psi, a)$,
- $\rho(\xi \vee \psi, a) = \rho(\xi, a) \vee \rho(\psi, a)$,
- $\rho(X\psi, a) = \psi$,
- $\rho(\xi U \psi, a) = \rho(\psi, a) \vee (\rho(\xi, a) \wedge \xi U \psi)$.

Back to Trees

Games, vis alternating automata, provide the key to obtaining elementary decision procedures to numerous modal, temporal, and dynamic logics.

Theorem [Kupferman&V.&Wolper, 1994]: For each CTL formula φ there is an alternating Büchi tree automaton A_φ with $||\varphi||$ states such that $models(\varphi) = L(A_\varphi)$.
Theorem [KVW, 1986]: There is an exponential translation of alternating Büchi tree automata to nondeterministic Büchi tree automata.

Known: Nonemptiness of nondeterministic Büchi tree automata can be checked in quadratic time [V.&Wolper, 1984]

Corollary: There is an exponential algorithm for satisfiability of CTL [Emerson&Halpern, 1985]

Discussion

Major Points:

- The *logic-automata connection* is one of the most fundamental paradigms of logic.
- One of the major benefits of this paradigm is its algorithmic consequences.
- A newer component of this approach is that of *games*, and *alternating automata* as their automata-theoretic counterpart.
- The interaction between logic, automata, games. and algorithms yields a fertile research area.

Tower of Abstractions

Key idea in science: *abstraction tower*

strings

quarks

hadrons

atoms

molecules

amino acids

genes

genomes

organisms

populations

Abstraction Tower in CS

CS Abstraction Tower:

analog devices
digital devices
microprocessors
assembly languages
high-level languages
libraries
software frameworks

Crux: Abstraction tower is the only way to deal with complexity!

Similarly: We need high-level algorithmic building blocks, e.g., *BFS*, *DFS*.

This talk: *Games/alternation* as a high-level algorithmic construct.

Bottom line: Alternation is a key algorithmic construct in automated reasoning — used in industrial tools.