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., $(\forall x, y, z)(\text{mult}(x, y, z) \leftrightarrow \text{mult}(y, x, z))$ – commutativity.

- **Paradigm II: Machines** – imperative formalism
  - Specify computations, e.g., Turing machines, finite-state machines, etc.

**Surprising Phenomenon**: Intimate connection between logic and machines – automata-theoretic approach.
Nondeterministic Finite Automata

\[ A = (\Sigma, S, S_0, \rho, F) \]

- **Alphabet:** \( \Sigma \)
- **States:** \( S \)
- **Initial states:** \( S_0 \subseteq S \)
- **Nondeterministic transition function:** \( \rho : S \times \Sigma \rightarrow 2^S \)
- **Accepting states:** \( F \subseteq S \)

**Input word:** \( a_0, a_1, \ldots, a_{n-1} \)

**Run:** \( s_0, s_1, \ldots, s_n \)
- \( s_0 \in S_0 \)
- \( s_{i+1} \in \rho(s_i, a_i) \text{ for } i \geq 0 \)

**Acceptance:** \( s_n \in F \)

**Recognition:** \( L(A) \) – words accepted by \( A \).

**Example:**

![Diagram of a nondeterministic finite automaton](image)

- ends with 1’s

**Fact:** NFAs define the class \( \text{Reg} \) of regular languages.
Logic of Finite Words

View finite word $w = a_0, \ldots, a_{n-1}$ over alphabet $\Sigma$ 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: $(\exists x)((\forall y)(\neg(x < y)) \land P_a(x))$ – 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:** \( E = \{(s, t) : t \in \rho(s, a) \text{ for some } a \in \Sigma\} \)

**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* (space complexity: NLOGSPACE-complete).
**MSO Satisfiability – Finite Words**

**Satisfiability**: models(ψ) ≠ ∅

**Satisfiability Problem**: Decide if given ψ is satisfiable.

**Lemma**: ψ is satisfiable iff Aψ is nonempty.

**Corollary**: MSO satisfiability is decidable.
- Translate ψ to Aψ.
- Check nonemptiness of Aψ.

**Complexity**:
- **Upper Bound**: Nonelementary Growth
  \[2 \cdot 2^n\] (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* $A = (\Sigma, S, S_0, \rho, F)$

- $\Sigma$: 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 $\omega$-Reg of $\omega$-regular languages.
Examples

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

\[\begin{array}{c}
\cdot \\
\downarrow \\
0 \\
\uparrow \\
0
\end{array}
\begin{array}{c}
\cdot \\
\downarrow \\
1 \\
\uparrow \\
1
\end{array}\]

– infinitely many 1’s

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

\[\begin{array}{c}
\cdot \\
\downarrow \\
0, 1 \\
\uparrow \\
1
\end{array}
\begin{array}{c}
\cdot \\
\downarrow \\
1 \\
\uparrow \\
1
\end{array}\]

– finitely many 0’s
Logic of Infinite Words

View infinite word $w = a_0, a_1, \ldots$ over alphabet $\Sigma$ as a mathematical structure:
- Domain: $\mathbb{N}$
- 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 event point.

$$(\exists Q)(\forall x)(\forall y)((((Q(x) \land y = x + 1) \rightarrow (\neg Q(y))) \land
((\neg Q(x)) \land y = x + 1) \rightarrow Q(y)) \land
(x = 0 \rightarrow Q(x)) \land (Q(x) \rightarrow q(x)))$$
NBA vs. MSO

**Theorem** [Büchi, 1962]: MSO $\equiv$ NBA
- Both MSO and NBA define the class $\omega$-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
Problem: subset construction fails!

\[ \rho(\{s\}, 0) = \{s, t\}, \rho(\{s, t\}, 0) = \{s, t\} \]

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**: $E = \{(s, t) : t \in \rho(s, a) \text{ for some } a \in \Sigma\}$

**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 (space complexity: NLOGSPACE-complete).
Satisfiability: $\text{models}(\psi) \neq \emptyset$

Satisfiability Problem: Decide if given $\psi$ is satisfiable.

**Lemma:** $\psi$ is satisfiable iff $A_\psi$ is nonempty.

Corollary: MSO satisfiability is decidable.

- Translate $\psi$ to $A_\psi$.
- Check nonemptiness of $A_\psi$.

Complexity:
- **Upper Bound:** Nonelementary Growth
  \[ 2^{O(n \log n)} \]
  (tower of height $O(n)$)
- **Lower Bound** [Stockmeyer, 1974]: Satisfiability of FO over infinite words is nonelementary (no bounded-height tower).
Prior, 1914–1969, Philosophical Preoccupations:

- **Religion**: Methodist, Presbyterian, 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”
Temporal and Classical Logics

Key Theorems:

- **Kamp, 1968**: Linear temporal logic with past and binary temporal connectives (“until” and “since”) has precisely the expressive power of FO over the integers.

- **Thomas, 1979**: FO over naturals has the expressive power of star-free $\omega$-regular expressions (MSO=$\omega$-regular).

Precursors:

- **Büchi, 1962**: On infinite words, MSO=RE

- **McNaughton & Papert, 1971**: On finite words, FO=star-free-RE
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
- $\pi : W \rightarrow \text{Powerset}(\text{Prop})$: 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 \( \varphi \) –

- all computations in \( L(M) \) satisfy \( \varphi \).

-------------------------------------------------------------

-------------------------------------------------------------

-------------------------------------------------------------

-------------------------------------------------------------
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** $\varphi$: $\varphi$ holds in the next state.
- **eventually** $\varphi$: $\varphi$ holds eventually
- **always** $\varphi$: $\varphi$ holds from now on
- **$\varphi$ until** $\psi$: $\varphi$ holds until $\psi$ holds.

$$
\pi, w \models next \varphi \text{ if } w \cdots \varphi \cdots
$$

$$
\pi, w \models \varphi \text{ until } \psi \text{ if } w \varphi \cdots \varphi \cdots \psi \cdots
$$
Examples

- always not (CS₁ and CS₂): 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]).

\[ \text{LTL}=\text{FO}=\text{star-free } \omega\text{-RE } < \text{MSO}=\omega\text{-RE} \]

Meyer on LTL, 1980, in “Ten Thousand and One Logics of Programming”:

“The corollary due to Meyer – I have to get in my controversial remark – is that that [GPSS’80] makes it theoretically uninteresting.”
Easy Direction: $\text{LTL} \leftrightarrow \text{FO}$

Example: $\varphi = \theta \text{ until } \psi$

$\text{FO}(\varphi)(x) :$

$$(\exists y)(y > x \land \text{FO}(\psi)(y) \land (\forall z)((x \leq z < y) \rightarrow \text{FO}(\theta)(z)))$$

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 (PSPACE-complete).

**Industrial Impact:**

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

**Question**: What is the key to efficient translation?

**Answer**: *Games*!

**Digression**: Games, complexity, and algorithms.
Complexity Theory

**Key CS Question, 1930s:** What can be mechanized?

**Next Question, 1960s:** How hard it is to mechanize it?

**Hardness:** Usage of computational resources

- *Time*
- *Space*

**Complexity Hierarchy:**

\[
\text{LOGSPACE} \subseteq \text{PTIME} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \subseteq \ldots
\]
Nondeterminism

Intuition: “It is easier to criticize than to do.”

P vs NP:

\textit{PTIME}: Can be \textit{solved} in polynomial time

\textit{NPTIME}: Can be \textit{checked} in polynomial time

Complexity Hierarchy:

\text{LOGSPACE} \subseteq \text{NLOGSPACE} \subseteq \text{PTIME} \subseteq \text{NPTIME} \\
\subseteq \text{PSPACE} = \text{NPSPACE} \subseteq \text{EXPTIME} \subseteq \text{NEXPTIME} \subseteq \ldots
Co-Nondeterminism

Intuition:

- **Nondeterminism**: check solutions – e.g., satisfiability
- **Co-nondeterminism**: check counterexamples – e.g., unsatisfiability

Complexity Hierarchy:

\[
\text{LOGSPACE} \subseteq \text{NLOGSPACE} = \text{co-NLOGSPACE} \subseteq \text{PTIME} \subseteq \text{NPTIME} = \text{co-NPTIME} \subseteq \text{PSPACE} = \text{co-NPSPACE} \subseteq \text{EXPTIME} \ldots
\]
Alternation

(Co)-Nondeterminism–Perspective Change:

- **Old**: Checking (solutions or counterexamples)
- **New**: Guessing moves
  - **Nondeterminism**: existential choice
  - **Co-Nondeterminism**: universal choice

**Alternation**: Chandra-Kozen-Stockmeyer, 1981
Combine $\exists$-choice and $\forall$-choice
  - $\exists$-state: $\exists$-choice
  - $\forall$-state: $\forall$-choice

**Easy Observations:**

- $\text{NPTIME} \subseteq \text{APTIME} \supseteq \text{co-NPTIME}$
- $\text{APTIME} = \text{co-APTIME}$
Example: Boolean Satisfiability

\( \varphi \): Boolean formula over \( x_1, \ldots, x_n \)

Decision Problems:

1. **SAT**: Is \( \varphi \) satisfiable? – NPTIME

   Guess a truth assignment \( \tau \) and check that \( \tau \models \varphi \).

2. **UNSAT**: Is \( \varphi \) unsatisfiable? – co-NPTIME

   Guess a truth assignment \( \tau \) and check that \( \tau \models \varphi \).

3. **QBF**: Is \( \exists x_1 \forall x_2 \exists x_3 \ldots \varphi \) true? – APTIME

   Check that for some \( x_1 \) for all \( x_2 \) for some \( x_3 \) \ldots \( \varphi \) holds.
Alternation $= \text{Games}$

**Players**: $\exists$-player, $\forall$-player

- $\exists$-state: $\exists$-player chooses move
- $\forall$-state: $\forall$-player chooses move

**Acceptance**: $\exists$-player has a winning strategy

**Run**: Strategy tree for $\exists$-player
Alternation and Unbounded Parallelism

“Be fruitful, and multiply”:

- $\exists$-move: fork *disjunctively*

- $\forall$-move: fork *conjunctively*

Note:

- Minimum communication between child processes
- Unbounded number of child processes
Alternation and Complexity

CKS’81:

Upper Bounds:

• $\text{ATIME}[f(n)] \subseteq \text{SPACE}[f^2(n)]$

  \textit{Intuition}: Search for strategy tree recursively

• $\text{ASPACE}[f(n)] \subseteq \text{TIME}[2f(n)]$

  \textit{Intuition}: Compute set of winning configurations bottom up.

Lower Bounds:

• $\text{SPACE}[f(n)] \subseteq \text{ATIME}[f(n)]$

• $\text{TIME}[2f(n)] \subseteq \text{ASPACE}[f(n)]$
Consequences

Upward Collapse:

- $\text{ALOGSPACE} = \text{PTIME}$
- $\text{APTIME} = \text{PSPACE}$
- $\text{APSPACE} = \text{EXPTIME}$

Applications:

- “In $\text{APTIME}$” $\rightarrow$ “in $\text{PSPACE}$”
- “$\text{APTIME}$-hard” $\rightarrow$ “$\text{PSPACE}$-hard”.

QBF:

- Natural algorithm is in $\text{APTIME}$ $\rightarrow$ “in $\text{PSPACE}$”
- Prove $\text{APTIME}$-hardness à la Cook $\rightarrow$ “$\text{PSPACE}$-hard”.

Corollary: QBF is PSPACE-complete.
Modal Logic K

Syntax:

• Propositional logic

• $\Diamond \varphi$ (possibly $\varphi$), $\Box \varphi$ (necessarily $\varphi$)

Proviso: Positive normal form

Kripke structure: $M = (W, R, \pi)$

• $W$: worlds

• $R \subseteq W^2$: Possibility relation
  \[ R(u) = \{v : (u, v) \in R\} \]

• $\pi : W \rightarrow 2^{Prop}$: Truth assignments

Semantics

• $M, w \models p$ if $p \in \pi(w)$

• $M, w \models \Diamond \varphi$ if $M, u \models \varphi$ for some $u \in R(w)$

• $M, w \models \Box \varphi$ if $M, u \models \varphi$ for all $u \in R(w)$
Modal Model Checking

Input:

- $\varphi$: modal formula
- $M = (W, R, \pi)$: Kripke structure
- $w \in W$: world

Problem: $M, w \models \varphi$?

Algorithm: $K\text{-MC}(\varphi, M, w)$

case
- $\varphi$ propositional: return $\pi(w) \models \varphi$
- $\varphi = \theta_1 \lor \theta_2$: (∃-branch) return $K\text{-MC}(\theta_i, M, w)$
- $\varphi = \theta_1 \land \theta_2$: (∀-branch) return $K\text{-MC}(\theta_i, M, w)$
- $\varphi = \lozenge \psi$: (∃-branch) return $K\text{-MC}(\psi, M, u)$ for $u \in R(w)$
- $\varphi = \Box \psi$: (∀-branch) return $K\text{-MC}(\psi, M, u)$ for $u \in R(w)$
esac.

Correctness: Immediate!
Complexity Analysis

Algorithm’s state: $(\theta, M, u)$

- $\theta$: $O(\log |\varphi|)$ bits
- $M$: fixed
- $u$: $O(\log |M|)$ bits

Conclusion: $\text{ASPACE}[\log |M| + \log |\varphi|]$

Therefore: $\text{K-MC} \in \text{ALOGSPACE}=\text{PTIME}$
(originally by Clarke&Emerson, 1981).
Modal Satisfiability

- \( \text{sub}(\varphi) \): all subformulas of \( \varphi \)

- **Valuation for** \( \varphi - \alpha \): \( \text{sub}(\varphi) \rightarrow \{0, 1\} \)

*Propositional consistency:*

- \( \alpha(\varphi) = 1 \)
- Not: \( \alpha(p) = 1 \) and \( \alpha(\neg p) = 1 \)
- Not: \( \alpha(p) = 0 \) and \( \alpha(\neg p) = 0 \)
- \( \alpha(\theta_1 \land \theta_2) = 1 \) implies \( \alpha(\theta_1) = 1 \) and \( \alpha(\theta_2) = 1 \)
- \( \alpha(\theta_1 \land \theta_2) = 0 \) implies \( \alpha(\theta_1) = 0 \) or \( \alpha(\theta_2) = 0 \)
- \( \alpha(\theta_1 \lor \theta_2) = 1 \) implies \( \alpha(\theta_1) = 1 \) or \( \alpha(\theta_2) = 1 \)
- \( \alpha(\theta_1 \lor \theta_2) = 0 \) implies \( \alpha(\theta_1) = 0 \) and \( \alpha(\theta_2) = 0 \)

*Definition:* \( \square(\alpha) = \{ \theta : \alpha(\square \theta) = 1 \} \).

*Lemma:* \( \varphi \) is satisfiable iff there is a valuation \( \alpha \) for \( \varphi \) such that if \( \alpha(\Diamond \psi) = 1 \), then \( \psi \land \bigwedge \square(\alpha) \) is satisfiable.
Intuition

Lemma: \( \varphi \) is satisfiable iff there is a valuation \( \alpha \) for \( \varphi \) such that if \( \alpha(\Diamond \psi) = 1 \), then \( \psi \land \bigwedge \Box(\alpha) \) is satisfiable.

Only if: \( M, w \models \varphi \)
Take: \( \alpha(\theta) = 1 \leftrightarrow M, w \models \theta \)

If: Satisfy each \( \Diamond \) separately

\[
\begin{array}{c}
\Diamond \beta, \Box \gamma, \Diamond \delta, \Diamond \eta \\
\beta, \gamma, \delta & \beta, \gamma, \eta
\end{array}
\]
**Algorithm**

**Algorithm**: $K$-SAT($\varphi$)

($\exists$-branch): Select valuation $\alpha$ for $\varphi$
($\forall$-branch): Select $\psi$ such that $\alpha(\Box\psi) = 1$, and return $K$-SAT($\psi \land \bigwedge \Box(\alpha)$)

**Correctness**: Immediate!

**Complexity Analysis**:

- Each step is in PTIME.
- Number of steps is polynomial.

*Therefore*: $K$-SAT $\in$ APTIME=PSPACE (originally by Ladner, 1977).

*In practice*: Basis for practical algorithm – valuations selected using a SAT solver.
Lower Bound

Easy reduction from APTIME:

- Each TM configuration is expressed by a propositional formula.
- $\exists$-moves are expressed using $\Diamond$-formulas (à la Cook).
- $\forall$-moves are expressed using $\Box$-formulas (à la Cook).
- Polynomially many moves $\rightarrow$ formulas of polynomial size.

Therefore: K-SAT is PSPACE-complete (originally by Ladner, 1977).
LTL Refresher

**Syntax:**
- Propositional logic
- $next\, \varphi, \varphi \text{ until } \psi$

**Temporal structure:** $M = (W, R, \pi)$
- $W$: worlds
- $R : W \rightarrow W$: successor function
- $\pi : W \rightarrow 2^{Prop}$: truth assignments

**Semantics**
- $M, w \models p$ if $p \in \pi(w)$
- $M, w \models next \, \varphi$ if $M, R(w) \models \varphi$

- $M, w \models \varphi \text{ until } \psi$ if $w \bullet \varphi \bullet \varphi \bullet \varphi \bullet \psi \bullet \ldots$

**Fact:** $(\varphi \text{ until } \psi) \equiv (\psi \lor (\varphi \land next(\varphi \text{ until } \psi)))$. 
Temporal Model Checking

Input:

- $\varphi$: temporal formula
- $M = (W, R, \pi)$: temporal structure
- $w \in W$: world

Problem: $M, w \models \varphi$?

Algorithm: \text{LTL-MC}(\varphi, M, w) – game semantics

\begin{cases} 
\varphi \text{ propositional: return } \pi(w) \models \varphi \\
\varphi = \theta_1 \lor \theta_2: (\exists\text{-branch}) \text{ return } \text{LTL-MC}(\theta_i, M, w) \\
\varphi = \theta_1 \land \theta_2: (\forall\text{-branch}) \text{ return } \text{LTL-MC}(\theta_i, M, w) \\
\varphi = \text{next } \psi: \text{ return } \text{LTL-MC}(\psi, M, R(w)) \\
\varphi = \theta \text{ until } \psi: \text{ return } \text{LTL-MC}(\psi, M, w) \text{ or return } \\
\left( \text{LTL-MC}(\theta, M, w) \text{ and } \text{LTL-MC}(\theta \text{ until } \psi, M, R(w)) \right) \\
esac.

But: When does the game end?
Problem: Algorithm may not terminate!!!

Solution: Redefine games

- Standard alternation is a finite game between $\exists$ and $\forall$.

- Here we need an infinite game.

- In an infinite play $\exists$ needs to visit non-$\text{until}$ formulas infinitely often – “not get stuck in one $\text{until}$ formula”.

Büchi Alternation Muller&Schupp, 1985:

- Infinite computations allowed

- On infinite computations $\exists$ needs to visit accepting states $\infty$ often.

**Lemma:** Büchi-ASPACE$[f(n)] \subseteq \text{TIME}[2^{f(n)}]

**Corollary:** LTL-MC $\in$ Büchi-ALOGSPACE$=$PTIME
**LTL Satisfiability**

**Hope:** Use Büchi alternation to adapt K-SAT to LTL-SAT.

**Problems:**

- What is time bounded Büchi alternation Büchi-ATIME[$f(n)$]?

- Successors cannot be split!
Alternating Automata

**Alternating automata**: 2-player games

**Nondeterministic transition**: \( \rho(s, a) = t_1 \lor t_2 \lor t_3 \)

**Alternating transition**: \( \rho(s, a) = (t_1 \land t_2) \lor t_3 \)
“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 function**: \( \rho : S \times \Sigma \to \mathcal{B}^+(S) \)
(positive Boolean formulas over \( S \))

- \( P \models \rho(s, a) \) – \( P \) satisfies \( \rho(s, a) \)
  - \( P \models \text{true} \)
  - \( P \not\models \text{false} \)
  - \( P \models (\theta \lor \psi) \) if \( P \models \theta \) or \( P \models \psi \)
  - \( P \models (\theta \land \psi) \) if \( P \models \theta \) and \( P \models \psi \)
Alternating Automata on Finite Words

Brzozowski&Leiss, 1980: Boolean automata

\[ A = (\Sigma, S, s_0, \rho, F) \]

- \( \Sigma, S, F \subseteq S \): as before
- \( s_0 \in S \): initial state
- \( \rho : S \times \Sigma \rightarrow B^+(S) \): alternating transition function

Game:
- Board: \( a_0, \ldots, a_{n-1} \)
- Positions: \( S \times \{0, \ldots, n - 1\} \)
- 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 at \( (s', n) \) if \( s' \in F \)

Acceptance: Automaton has a winning strategy.
Expressiveness

Expressiveness: ability to recognize sets of “boards”, i.e., languages.

BL’80,CKS’81:

- Nondeterministic automata: regular languages
- Alternating automata: regular languages

What is the point?: Succinctness

Exponential gap:

- Exponential translation from alternating automata to nondeterministic automata
- In the worst case this is the best possible

Crux: 2-player games $\leftrightarrow$ 1-player games
Eliminating Alternation

Alternating automaton: $A = (\Sigma, S, s_0, \rho, F)$

Subset Construction [BL’80, CKS’81]

- $A^n = (\Sigma, 2^S, \{s_0\}, \rho^n, F^n)$
- $\rho^n(P, a) = \{T : T \models \bigwedge_{t \in P} \rho(t, a)\}$
- $F^n = \{P : P \subseteq F\}$

Lemma: $L(A) = L(A^n)$
Alternating Büchi Automata

\[ A = (\Sigma, S, s_0, \rho, F') \]

**Game:**

- **Infinite board:** \( a_0, a_1 \ldots \)
- **Positions:** \( S \times \{0, 1, \ldots\} \)
- **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

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

- $$\rho(m, 1) = m$$
- $$\rho(m, 0) = m \land s$$
- $$\rho(s, 1) = \text{true}$$
- $$\rho(s, 0) = s$$

**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.

$$L(A) = \text{infinitely many 1’s.}$$
Expressiveness

Miyano & Hayashi, 1984:

• Nondeterministic Büchi automata: \( \omega \)-regular languages

• Alternating automata: \( \omega \)-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
Eliminating Büchi Alternation

Alternating automaton: \( A = (\Sigma, S, s_0, \rho, F) \)

**Subset Construction with Breakpoints [MH’84]:**

- \( A^n = (\Sigma, 2^S \times 2^S, \{s_0\}, \emptyset, 2^S \times \{\emptyset\}) \)
- \( \rho^n((P, \emptyset), a) = \{(T, T - F) : T \models \bigwedge_{t \in P} \rho(s, a)\} \)
- \( \rho^n((P, Q), a) = \{(T, T' - F) : T \models \bigwedge_{t \in P} \rho(t, a) \}
\text{ and } T' \models \bigwedge_{t \in Q} \rho(t, a)\} \)
- \( F^n = 2^S \times \{\emptyset\} \)

**Lemma:** \( L(A) = L(A^n) \)

**Intuition:** Double subset construction

- First component: standard subset construction
- Second component: keeps track of obligations to visit \( F \)
Back to LTL

Old temporal structure: $M = (W, R, \pi)$

- $W$: worlds
- $R : W \rightarrow W$: successor function
- $\pi : W \rightarrow 2^{Prop}$: truth assignments

New temporal structure: $\sigma \in (2^{Prop})^\omega$ (unwind the function $R$)

Temporal Semantics: $models(\varphi) \subseteq (2^{Prop})^\omega$

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

**Intuition**: Consider LTL-MC as an alternating Büchi automaton.
From LTL-MC to Alternating Büchi Automata

**Algorithm:** LTL-MC(ϕ, M, w)

```plaintext
case
φ propositional: return π(w) |= φ
φ = θ₁ ∨ θ₂: (∃-branch) return LTL-MC(θᵢ, M, w)
φ = θ₁ ∧ θ₂: (∀-branch) return LTL-MC(θᵢ, M, w)
φ = next ψ: return LTL-MC(ψ, M, R(w))
φ = θ until ψ: return LTL-MC(ψ, M, w) or return
(LTL-MC(θ, M, w) and LTL-MC(θ until ψ, M, R(w)))
esac.
```

A_φ = \{2^{Prop}, sub(φ), φ, ρ, nonU(φ)\}:

- \( ρ(p, a) = \text{true} \) if \( p ∈ a \),
- \( ρ(p, a) = \text{false} \) if \( p ∉ a \),
- \( ρ(ξ ∨ ψ, a) = ρ(ξ, a) ∨ ρ(ψ, a) \),
- \( ρ(ξ ∧ ψ, a) = ρ(ξ, a) ∧ ρ(ψ, a) \),
- \( ρ(\text{next } ψ, a) = ψ \),
- \( ρ(ξ \text{ until } ψ, a) = ρ(ψ, a) ∨ (ρ(ξ, a) ∧ ξ \text{ until } ψ) \).
Alternating Automata Nonemptiness

**Given:** Alternating Büchi automaton $A$

**Two-step algorithm:**

- Construct *nondeterministic Büchi automaton* $A^n$ such that $L(A^n) = L(A)$ (exponential blow-up)
- Test $L(A^n) \neq \emptyset$ (NLOGSPACE)

**Problem:** $A^n$ is exponentially large.

**Solution:** Construct $A^n$ *on-the-fly*.

**Corollary 1:** Alternating Büchi automata nonemptiness is in PSPACE.

**Corollary 2:** LTL satisfiability is in PSPACE (originally by Sistla & Clarke, 1985).
Alternation

Two perspectives:
- Two-player games
- Control mechanism for parallel processing

Two Applications:
- Model checking
- Satisfiability checking

Bottom line: Alternation is a key algorithmic construct in automated reasoning — used in industrial tools.
- Gastin-Oddoux – LTL2BA (2001)
- Intel IDC – ForSpec Compiler (2001)
Designs are Labeled Graphs

**Key Idea:** Designs can be represented as transition systems (finite-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
- \( \pi : W \to \text{Powerset} (\text{Prop}) \): 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 \( \varphi \) –

- all computations in \( L(M) \) satisfy \( \varphi \).
Algorithmic Foundations

Basic Graph-Theoretic Problems:

• **Reachability:** Is there a *finite* path from $I$ to $F$?

  $I \cdot \longrightarrow \cdot F$

• **Fair Reachability:** Is there an *infinite* path from $I$ that goes through $F$ infinitely often.

  $I \cdot \longrightarrow \cdot F$ to $F$

**Note:** These paths may correspond to error traces.

• **Deadlock:** A finite path from $I$ to a state in which both $\text{write}_1$ and $\text{write}_2$ holds.

• **Livelock:** An infinite path from $I$ along which $\text{snd}$ holds infinitely often, but $\text{rcv}$ never holds.
Computational Complexity

**Complexity**: Linear time

- **Reachability**: breadth-first search or depth-first search
- **Fair Reachability**: depth-first search

The fundamental problem of model checking: the state-explosion problem – from $10^{20}$ states and beyond.

The critical breakthrough: symbolic model checking
Model Checking

The following are equivalent (V.-Wolper, 1985):

- $M$ satisfies $\varphi$
- all computations in $L(M)$ satisfy $\varphi$
- $L(M) \subseteq L(A_{\varphi})$
- $L(M) \cap \overline{L(A_{\varphi})} = \emptyset$
- $L(M) \cap L(A_{\neg \varphi}) = \emptyset$
- $L(M \times A_{\neg \varphi}) = \emptyset$

**In practice:** To check that $M$ satisfies $\varphi$, compose $M$ with $A_{\neg \varphi}$ and check whether the composite system has a reachable (fair) path.

**Intuition:** $A_{\neg \varphi}$ is a “watchdog” for “bad” behaviors. A reachable (fair) path means a bad behavior.
Computational Complexity

**Worst case:** linear in the size of the design space and exponential in the size of the specification.

**Real life:** Specification is given in the form of a list of properties $\varphi_1, \ldots, \varphi_n$. It suffices to check that $M$ satisfies $\varphi_i$ for $1 \leq i \leq n$.

**Moral:** There is life after exponential explosion.

**The real problem:** too many design states – symbolic methods needed
Verification: Good News and Bad News

Model Checking:
- **Given**: System $P$, specification $\varphi$.
- **Task**: Check that $P \models \varphi$

Success:
- **Algorithmic methods**: temporal specifications and finite-state programs.
- **Also**: Certain classes of infinite-state programs
- **Tools**: SMV, SPIN, SLAM, etc.
- **Impact** on industrial design practices is increasing.

Problems:
- Designing $P$ is hard and expensive.
- Redesigning $P$ when $P \not\models \varphi$ is hard and expensive.
Automated Design

Basic Idea:

- Start from spec \( \varphi \), design \( P \) such that \( P \models \varphi \).

\textit{Advantage:}
- No verification
- No re-design

- Derive \( P \) from \( \varphi \) algorithmically.

\textit{Advantage:}
- No design

\textbf{In essence:} Declarative programming taken to the limit.
Program Synthesis

**The Basic Idea:** Mechanical translation of human-understandable task specifications to a program that is known to meet the specifications.

**Deductive Approach** (Green, 1969, Waldinger and Lee, 1969, Manna and Waldinger, 1980)

- Prove realizability of function, e.g., \( (\forall x)(\exists y)(Pre(x) \rightarrow Post(x, y)) \)
- Extract *program* from realizability proof.

**Classical vs. Temporal Synthesis:**

- **Classical:** Synthesize transformational programs
- **Temporal:** Synthesize programs for ongoing computations (protocols, operating systems, controllers, etc.)
Synthesis of Ongoing Programs

Specs: Temporal logic formulas

Early 1980s: Satisfiability approach (Wolper, Clarke+Emerson, 1981)
- Given: $\varphi$
- Satisfiability: Construct $M \models \varphi$
- Synthesis: Extract $P$ from $M$.

Example: always $(\text{odd} \rightarrow \text{next } \neg \text{odd}) \wedge$
always $(\neg \text{odd} \rightarrow \text{next } \text{odd})$

\[ \text{odd} \xrightarrow{} \text{odd} \]
Reactive Systems

**Reactivity**: Ongoing interaction with environment (Harel+Pnueli, 1985), e.g., hardware, operating systems, communication protocols, etc. (also, *open systems*).

**Example**: Printer specification –

$J_i$ - job $i$ submitted, $P_i$ - job $i$ printed.

- **Safety**: two jobs are not printed together
  
  $\text{always } \neg (P_1 \land P_2)$

- **Liveness**: every job is eventually printed
  
  $\text{always } \land_{j=1}^{2} (J_i \rightarrow \text{eventually } P_i)$
Satisfiability and Synthesis

**Specification Satisfiable?** Yes!

*Model* $M$: A single state where $J_1$, $J_2$, $P_1$, and $P_2$ are all false.

**Extract program from* $M$? No!**

**Why?** Because $M$ handles only one input sequence.

- $J_1, J_2$: input variables, controlled by environment
- $P_1, P_2$: output variables, controlled by system

**Desired:** a system that handles *all* input sequences.

**Conclusion:** Satisfiability is inadequate for synthesis.
Realizability

$I$: input variables
$O$: output variables

**Game:**
- **System:** choose from $2^O$
- **Env:** choose from $2^I$

**Infinite Play:**
$i_0, i_1, i_2, \ldots$
$0_0, 0_1, 0_2, \ldots$

**Infinite Behavior:** $i_0 \cup o_0, i_1 \cup o_1, i_2 \cup o_2, \ldots$

**Win:** behavior $\models$ spec

**Specifications:** LTL formula on $I \cup O$

**Strategy:** Function $f : (2^I)^* \to 2^O$

**Realizability:** Pnueli+Rosner, 1989
Existence of winning strategy for specification.
Church’s Problem

Church, 1963: Realizability problem wrt specification expressed in MSO (monadic second-order theory of one successor function)

Büchi+Landweber, 1969:
- Realizability is decidable - nonelementary!
- If a winning strategy exists, then a finite-state winning strategy exists.
- Realizability algorithm produces finite-state strategy.


Question: LTL is subsumed by MSO, so what did Pnueli and Rosner do?

Answer: better algorithms - 2EXPTIME-complete.
Standard Critique

Impractical! 2EXPTIME is a horrible complexity.

Response:

- 2EXPTIME is just worst-case complexity.

- 2EXPTIME lower bound implies a doubly exponential bound on the size of the smallest strategy; thus, hand design cannot do better in the worst case.
Real Critique

- Algorithmics not ready for practical implementation.
- Complete specification is difficult.

Response: More research needed!

- Better algorithms
- Incremental algorithms – write spec incrementally
Discussion

**Question:** Can we hope to reduce a 2EXPTIME-complete approach to practice?

**Answer:**

- Worst-case analysis is pessimistic.
  - Mona solves nonelementary problems.
  - SAT-solvers solve huge NP-complete problems.
  - Model checkers solve PSPACE-complete problems.
  - Doubly exponential lower bound for program size.

- We need algorithms that blow-up only on hard instances

- Algorithmic engineering is needed.

- New promising approaches.