Alternation
as
An Algorithmic Construct

Moshe Y. Vardi

Rice University

Special Programme on Logic and Algorithms

Newton Institute for Mathematical Sciences

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:

LOGSPACE $\subseteq$ PTIME $\subseteq$ PSPACE $\subseteq$ EXPTIME $\subseteq$ . . .

Nondeterminism

Intuition: "It is easier to critic than to do."

P vs NP:

PTIME: Can be solved in polynomial time

NPTIME: Can be checked in polynomial time

Complexity Hierarchy:

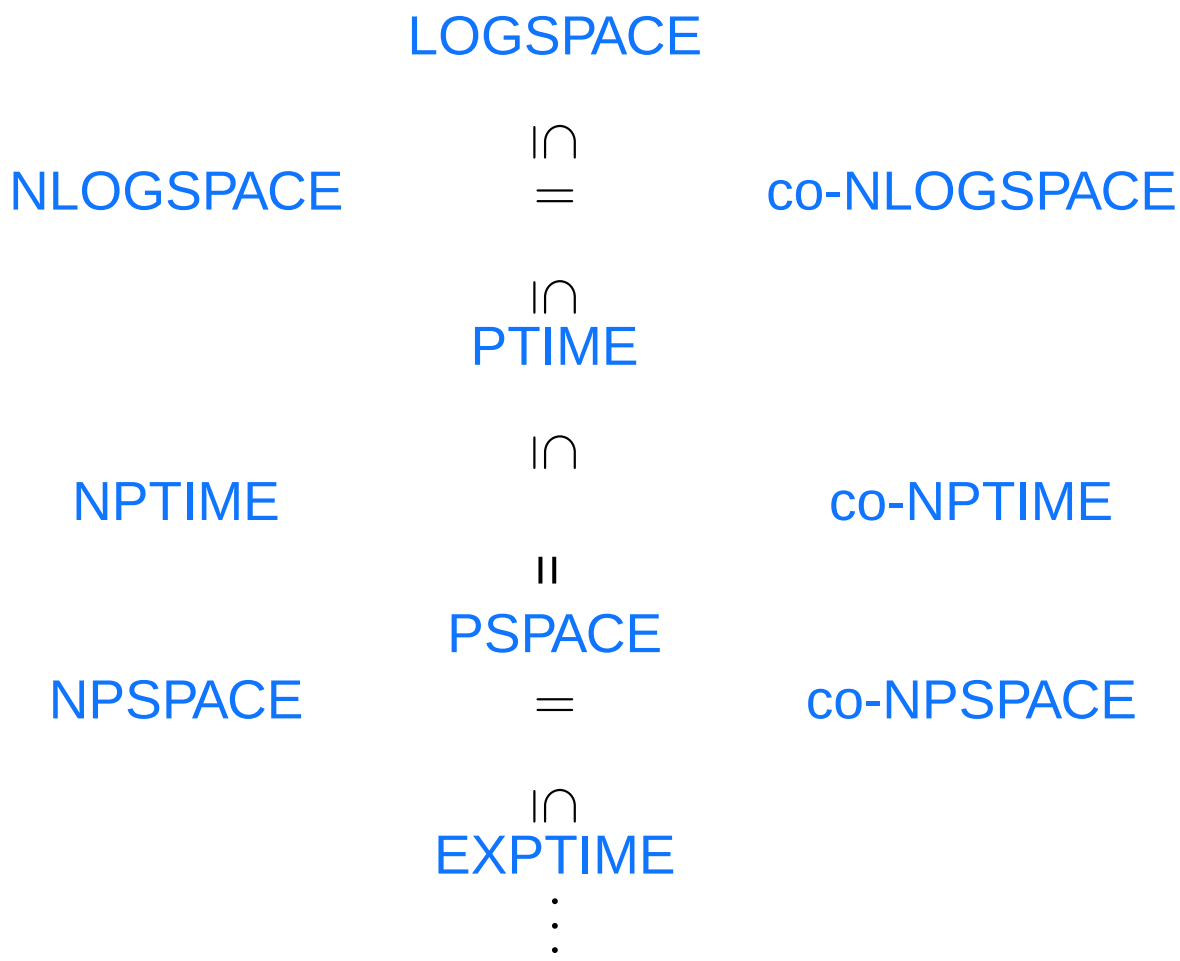
$$\text{LOGSPACE} \subseteq \text{NLOGSPACE} \subseteq \text{PTIME} \subseteq \text{NPTIME} \subseteq \text{PSPACE} = \text{NPSPACE} \subseteq \text{EXPTIME} \subseteq \text{NEXPTIME} \subseteq \ldots$$

Co-Nondeterminism

Intuition:

- *Nondeterminism*: check solutions
- *Co-nondeterminism*: check counterexamples

Complexity Hierarchy:



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:

- NPTIME $\subseteq$ APTIME $\supseteq$ co-NPTIME
- APTIME $=$ co-APTIME

Example: Boolean Satisfiability

φ: Boolean formula over $x_1, \ldots, x_n$

Decision Problems:

1. **SAT**: *Is φ satisfiable?* – NPTIME

   Guess a truth assignment τ and check that $\tau \models \varphi$.

2. **UNSAT**: *Is φ unsatisfiable?* – co-NPTIME

   Guess a truth assignment τ 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 $\ldots$ φ holds.

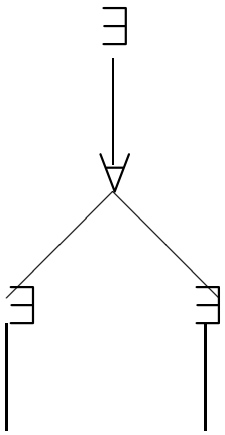
Alternation and Games

Players: ∃-player, ∀-player

- ∃-state: ∃-player chooses move
- ∀-state: ∀-player chooses move

Acceptance: ∃-player has a winning strategy

Run: Strategy tree for ∃-player



Alternation and Complexity

CKS'81:

Upper Bounds:

- ATIME$[f(n)] \subseteq$ SPACE$[f^2(n)]$

  Intuition: Search for strategy tree recursively

- ASPACE$[f(n)] \subseteq$ TIME$[2^{f(n)}]$

  Intuition: Compute set of winning configurations bottom up.

Lower Bounds:

- SPACE$[f(n)] \subseteq$ ATIME$[f(n)]$
- TIME$[2^{f(n)}] \subseteq$ ASPACE$[f(n)]$

Consequences

Collapse:

- ALOGSPACE=PTIME
- APTIME=PSPACE
- APSPACE=EXPTIME

Applications:

- “In APTIME” $\longrightarrow$ “in PSPACE”
- “APTIME-hard” $\longrightarrow$ “PSPACE-hard”.

QBF:

- Natural algorithm is in APTIME $\longrightarrow$ “in PSPACE”
- Prove APTIME-hardness à la Cook $\longrightarrow$ “PSPACE-hard”.

Corollary: QBF is PSPACE-complete.

Modal Logic K

Syntax:

- Propositional logic
- $\Diamond\varphi$ (possibly φ), $\Box\varphi$ (necessarily φ)

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:

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

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

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

case
φ propositional: return $\pi(w) \models \varphi$
$\varphi = \theta_1 \vee \theta_2$: ($\exists$-branch) return K-$MC(\theta_i, M, w)$
$\varphi = \theta_1 \wedge \theta_2$: ($\forall$-branch) return K-$MC(\theta_i, M, w)$
$\varphi = \Diamond\psi$: ($\exists$-branch) return K-$MC(\psi, M, u)$ for $u \in R(w)$
$\varphi = \Box\psi$: ($\forall$-branch) return K-$MC(\psi, M, u)$ for $u \in R(w)$
esac.

Correctness: Immediate!

Complexity Analysis

Algorithm's state: (θ, M, u)

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

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

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

Modal Satisfiability

- $sub(\varphi)$: all subformulas of φ
- *Valuation* for φ – α: $sub(\varphi) \rightarrow \{0, 1\}$

  Propositional consistency:

  - $\alpha(\varphi) = 1$
  - Not: $\alpha(p) = 1$ and $\alpha(\neg p) = 1$
  - Not: $\alpha(p) = 0$ and $\alpha(\neg p) = 0$
  - $\alpha(\theta_1 \wedge \theta_2) = 1$ implies $\alpha(\theta_1) = 1$ and $\alpha(\theta_2) = 1$
  - $\alpha(\theta_1 \wedge \theta_2) = 0$ implies $\alpha(\theta_1) = 0$ or $\alpha(\theta_2) = 0$
  - $\alpha(\theta_1 \vee \theta_2) = 1$ implies $\alpha(\theta_1) = 1$ or $\alpha(\theta_2) = 1$
  - $\alpha(\theta_1 \vee \theta_2) = 0$ implies $\alpha(\theta_1) = 0$ and $\alpha(\theta_2) = 0$

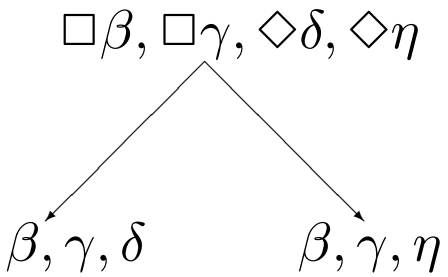
Definition: $\Box(\alpha) = \{\theta : \alpha(\Box\theta) = 1\}$.

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

Intuition

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

If: Satisfy each $\Diamond$ separately



Algorithm

Algorithm: $K\text{-}SAT(\varphi)$

(∃-branch): Select valuation α for φ
(∀-branch): Select ψ such that $\alpha(\Diamond\psi) = 1$, and return $K\text{-}SAT(\psi \wedge \bigwedge \Box(\alpha))$

Correctness: Immediate!

Complexity Analysis:

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

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 (á la Cook).
- $\forall$-moves are expressed using $\Box$-formulas (á la Cook).
- Polynomially many moves $\rightarrow$ formulas of polynomial size.

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

Linear Temporal Logic

Syntax:

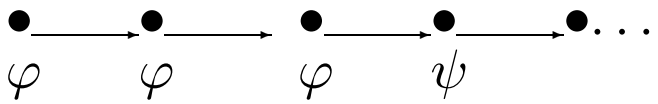
- Propositional logic
- $X\varphi$ (next φ), $\varphi U \psi$ (φ until ψ)

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

- W: worlds
- $R : W \to W$: Successor relation
- $\pi : W \to 2^{Prop}$: Truth assignments

Semantics

- $M, w \models X\varphi$ if $M, R(w) \models \varphi$

- $M, w \models \varphi U \psi$ if w 

Fact: $(\varphi U \psi) \equiv (\psi \vee X(\varphi U \psi))$.

Temporal Model Checking

Input:

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

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

Algorithm: LTL-$MC(\varphi, M, w)$

case
φ propositional: return $\pi(w) \models \varphi$
$\varphi = \theta_1 \vee \theta_2$: ($\exists$-branch) return LTL-$MC(\theta_i, M, w)$
$\varphi = \theta_1 \wedge \theta_2$: ($\forall$-branch) return LTL-$MC(\theta_i, M, w)$
$\varphi = X\psi$: return LTL-$MC(\psi, M, R(w))$
$\varphi = \theta U \psi$: return LTL-$MC(\psi, M, w)$ or return
$\big($ LTL-$MC(\theta, M, w)$ and LTL-$MC(\theta U \psi, M, R(w))$ $\big)$
esac.

From Finite to Infinite Games

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-U formulas infinitely often.

Büchi Alternation Muller&Schupp, 1985:

- Infinite computations allowed
- On infinite computations $\exists$ needs to visit ∞ accepting states.

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

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

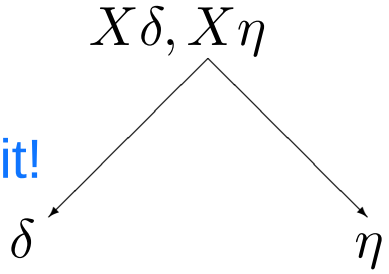
LTL Satisfiability

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!



Automata

Intuition: automata $=$ games over a board

Nondeterministic automata: 1-player games

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

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

Input: $a_0, a_1, \ldots, a_{n-1}$ ("board")

Accepting Run: $s_0, s_1, \ldots, s_n$

- $s_0 \in S_0$
- $s_{i+1} \in \rho(s_i, a_i)$, $i = 0, \ldots, n-1$
- $s_n \in F$

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: $\rho : S \times \Sigma \to \mathcal{B}^+(S)$
(positive Boolean formulas over S)

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
- $\rho : S \times \Sigma \to \mathcal{B}^+(S)$: 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

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

Büchi Automata

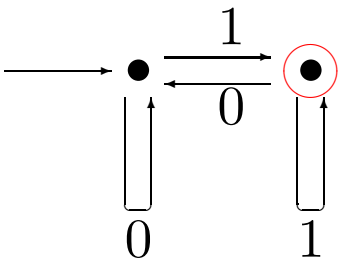
$A = (\Sigma, S, S_0, \rho, F)$

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

Input: $a_0, a_1, a_2, \ldots$ ("infinite board")

Accepting Run: $s_0, s_1, s_2, \ldots$

- $s_0 \in S_0$
- $s_{i+1} \in \rho(s_i, a_i)$, $i = 0, 1 \ldots$
- $s_i \in F$ for infinitely many i's.



– infinitely many 1's

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

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

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

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

Back to LTL

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

- W: worlds
- $R : W \rightarrow W$: Successor relation
- $\pi : W \rightarrow 2^{Prop}$: Truth assignments

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

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

Theorem: For each LTL formula φ there is an alternating Büchi automaton $A_{\varphi} = (2^{Prop}, sub(\varphi), \varphi, \rho, F)$ such that $models(\varphi) = L(A_{\varphi})$.

Intuition: Consider LTL-MC as an alternating Büchi automaton.

Nondeterministic Automata Nonemptiness

Nonemptiness problem: Given A, is $L(A) \neq \emptyset$?

Nondeterministic Büchi Automata: $A = (\Sigma, S, S_0, \rho, F)$

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

Lemma: $L(A) \neq \emptyset$ iff there a path in G_A from S_0 to some $t \in F$ and a cycle from t to itself.

Corollary: Nondeterministic Büchi automata nonemptiness is in NLOGSPACE.

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, 1986).

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 language

libraries

. . .

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: *Alternation* as a high-level algorithmic construct.

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.