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 – *automata-theoretic approach*.

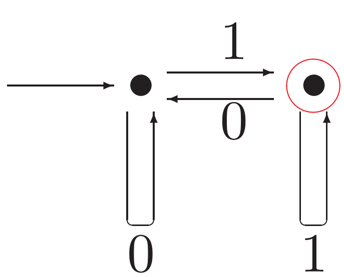
Nondeterministic Finite Automata

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

- *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* (space complexity: NLOGSPACE-complete).

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

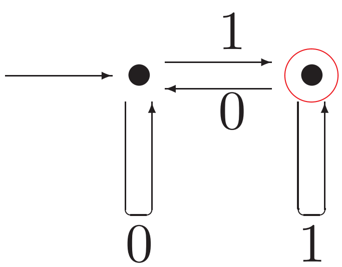
Input: $w = a_0, a_1 \ldots$
Run: $r = s_0, s_1 \ldots$
- $s_0 \in S_0$
- $s_{i+1} \in \rho(s_i, a_i)$

Acceptance: run visits F *infinitely often*.

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

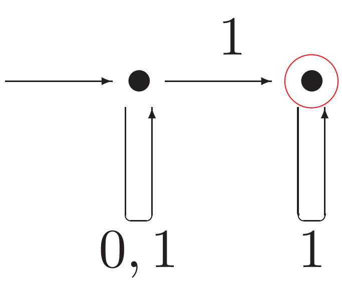
Examples

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



– 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 event 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

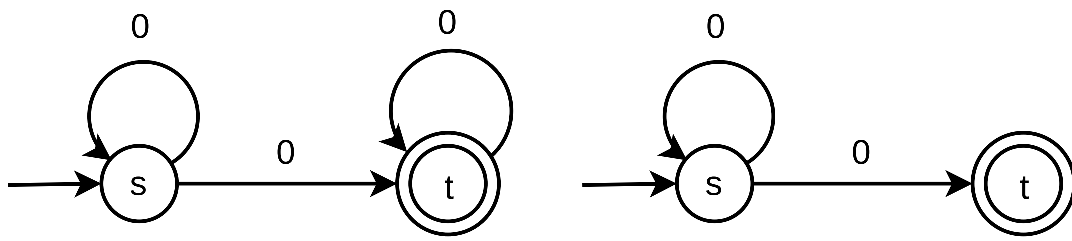
Theorem [Büchi, 1962]: MSO $\equiv$ NBA
- Both MSO and NBA define the class ω-Reg.

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!



$$\rho(\{s\},0) = \{s,t\},\ \rho(\{s,t\},0) = \{s,t\}$$

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: $E = \{(s, t) : t \in \rho(s, a)$ 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).

Catching Bugs with A Lasso

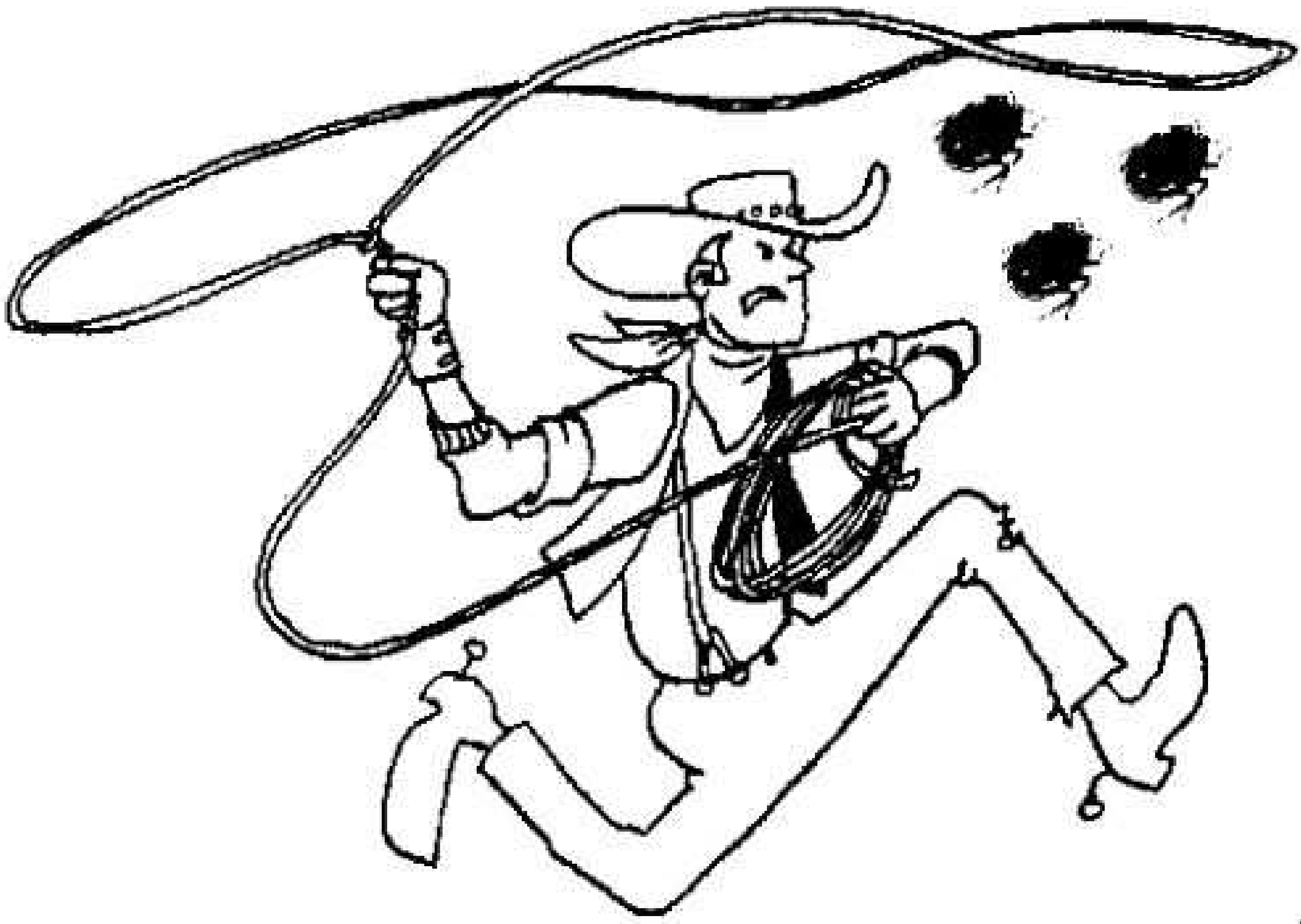


Figure 1: Ashutosh's blog, November 23, 2005

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”

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 ω-regular expressions (MSO=ω-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
- $\pi : W \rightarrow Powerset(Prop)$: 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.

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.

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

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

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]).

LTL=FO=star-free ω-RE $<$ MSO=ω-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."

Computational Complexity

Easy Direction: LTL$\mapsto$FO

Example: $\varphi = \theta \ until \ \psi$
$FO(\varphi)(x)$:

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

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 (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:

LOGSPACE $\subseteq$ PTIME $\subseteq$ PSPACE $\subseteq$ EXPTIME $\subseteq$ $\ldots$

Nondeterminism

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

P vs NP:

PTIME: Can be *solved* in polynomial time
NPTIME: Can be *checked* in polynomial time

Complexity Hierarchy:

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

Co-Nondeterminism

Intuition:

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

Complexity Hierarchy:

$$
\begin{array}{ccc}
 & \text{LOGSPACE} & \\
 & \rotatebox{-90}{$\subseteq$} & \\
\text{NLOGSPACE} & = & \text{co-NLOGSPACE} \\
 & \rotatebox{-90}{$\subseteq$} & \\
 & \text{PTIME} & \\
 & \rotatebox{-90}{$\subseteq$} & \\
\text{NPTIME} & & \text{co-NPTIME} \\
 & \rotatebox{90}{$=$} & \\
 & \text{PSPACE} & \\
\text{NPSPACE} & = & \text{co-NPSPACE} \\
 & \rotatebox{-90}{$\subseteq$} & \\
 & \text{EXPTIME} & \\
 & \vdots &
\end{array}
$$

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.

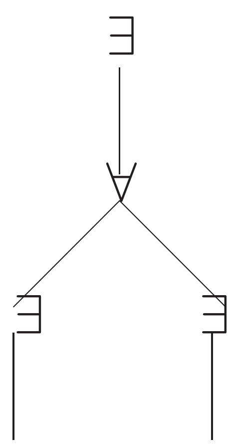
Alternation = Games

Players: ∃-player, ∀-player

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

Acceptance: ∃-player has a winning strategy

Run: Strategy tree for ∃-player



Alternation and 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:

- 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

Upward 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

  $R(u) = \{v : (u, v) \in R\}$

- $\pi : W \to 2^{Prop}$: 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

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.

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

If: Satisfy each $\Diamond$ separately

$$\begin{tikzcd} & \Box\beta, \Box\gamma, \Diamond\delta, \Diamond\eta \arrow[dl] \arrow[dr] & \\ \beta, \gamma, \delta & & \beta, \gamma, \eta \end{tikzcd}$$

Algorithm

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

($\exists$-branch): Select valuation α for φ
($\forall$-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 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 (á la Cook).
- $\forall$-moves are expressed using $\Box$-formulas (á la Cook).
- Polynomially many moves $\longrightarrow$ formulas of polynomial size.

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

LTL Refresher

Syntax:

- Propositional logic
- $next\ \varphi$, $\varphi\ until\ \psi$

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 $M, R(w) \models \varphi$
- $M, w \models \varphi\ until\ \psi$ if $w\ \underset{\varphi}{\bullet} \longrightarrow \underset{\varphi}{\bullet} \longrightarrow \quad \underset{\varphi}{\bullet} \longrightarrow \underset{\psi}{\bullet} \longrightarrow \bullet \ldots$

Fact: $(\varphi\ until\ \psi) \equiv (\psi \vee (\varphi \wedge next(\varphi\ until\ \psi)))$.

Temporal Model Checking

Input:

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

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

Algorithm: LTL-$MC(\varphi, M, w)$ – *game semantics*

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-$MC(\theta_i, M, w)$
$\varphi = next\ \psi$: return LTL-$MC(\psi, M, R(w))$
$\varphi = \theta\ until\ \psi$: return LTL-$MC(\psi, M, w)$ or return
(LTL-$MC(\theta, M, w)$ and LTL-$MC(\theta\ until\ \psi, M, R(w))$
)
esac.

But: When does the game end?

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-$until$ formulas infinitely often – "not get stuck in one $until$ formula".

Büchi Alternation Muller&Schupp, 1985:

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

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!

$$\begin{tikzcd} & next\ \delta, next\ \eta \arrow[dl] \arrow[dr] & \\ \delta & & \eta \end{tikzcd}$$

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

- $P \models \rho(s,a)$ – P *satisfies* $\rho(s,a)$
  - $P \models$ **true**
  - $P \not\models$ **false**
  - $P \models (\theta \vee \psi)$ if $P \models \theta$ or $P \models \psi$
  - $P \models (\theta \wedge \psi)$ if $P \models \theta$ and $P \models \psi$

Alternating Automata on Finite Words

Brzozowski&Leiss, 1980: Boolean automata

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

- 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

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 $\mapsto$ 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

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

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

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

Eliminating Büchi Alternation

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

Subset Construction [MH'84]:

- $A^n = (\Sigma, 2^S \times 2^S, (\{s_0\}, \emptyset), \rho^n, F^n)$
- $\rho^n((P, \emptyset), a) = \{(T, T - F) : T \models \bigwedge_{t \in P} \rho(s, a)\}$
- $\rho^n((P, Q), a) = \{(T, T' - F) : T \models \bigwedge_{t \in P} \rho(t, a)$ 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
- $R : W \to W$: successor function
- $\pi : W \to 2^{Prop}$: 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 φ there is an alternating Büchi automaton A_φ 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-$MC(\varphi, M, w)$

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-$MC(\theta_i, M, w)$
$\varphi = next\ \psi$: return LTL-$MC(\psi, M, R(w))$
$\varphi = \theta\ until\ \psi$: return LTL-$MC(\psi, M, w)$ or return
(LTL-$MC(\theta, M, w)$ and LTL-$MC(\theta\ until\ \psi, M, R(w))$
)
esac.

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

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

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).

The Role of the Board

Question: I was taught that Büchi games can be solved in quadratic time? Why is nonemptiness of alternating Büchi automata PSPACE-complete?

Answer: It's a bit subtle.

- Checking whether A_φ accepts *the* word given by a Kripke structure M is in PTIME.

- Checking whether A_φ accepts *some* word is PSPACE-complete.

Technically: Nonemptiness over a 1-*letter* alphabet is easy, but nonemptiness over a 2-*letter* alphabet is hard.

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 [V.&Wolper, 1986]: There is an exponential translation of CTL to nondeterministic Büchi tree automata.

Corollary: There is an exponential algorithm for satisfiability in CTL.

From Linear to Branching Time

Question: As I recall, CTL model checking is linear in the size of the formula. How can we do that with tree automata when there is an exponential blow-up in the construction?

Answer: It's all about 1-letter vs 2-letter alphabets.

- Extending the linear construction of alternating automata from LTL formulas to CTL formulas is easy, but we need to use tree automata, rather than word automata.

- Model checking amounts to checking nonemptiness of alternating tree automata over a 1-letter alphabet; it is in PTIME.

- Satisfiability checking amounts to checking nonemptiness of alternating tree automata over a 2-letter alphabet; it is EXPTIME-complete.

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 language

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.

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)

Verification

Model Checking:

- *Given*: System P, specification φ.
- *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 φ, design P such that $P \models \varphi$.

  Advantage:

  - No verification
  - No re-design

- Derive P from φ algorithmically.

  Advantage:

  - No design

In essenece: 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*: φ
- *Satisfiability*: Construct $M \models \varphi$
- *Synthesis*: Extract P from M.

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

$$\begin{tikzcd} odd \arrow[r, shift left] & \overline{odd} \arrow[l, shift left] \end{tikzcd}$$

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
  $always \ \neg(P_1 \wedge P_2)$
- *Liveness*: every jobs is eventually printed
  $always \ \bigwedge_{j=1}^{2}(J_i \rightarrow eventually \ P_i)$

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 $f : (2^I)^* \rightarrow 2^O$

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.
- If a winning strategy exists, then a finite-state winning strategy exists.
- Realizability algorithm produces finite-state strategy.

Rabin, 1972: Simpler solution via Rabin tree automata.

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

Strategy Trees

Infinite Tree: D^* (D - directions)

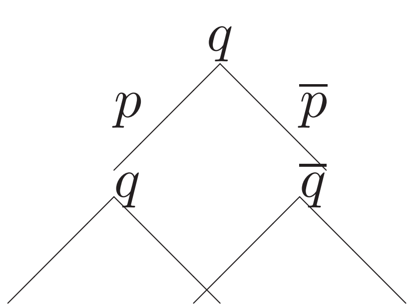
- *Root*: ε
- *Children*: xd, $x \in D^*$, $d \in D$

Labeled Infinite Tree: $\tau : D^* \rightarrow \Sigma$

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

> *Rabin's insight*: A strategy is a labeled tree with directions $D = 2^I$ and alphabet $\Sigma = 2^O$.

Example: $I = \{p\}$, $O = \{q\}$



Winning: Every branch satisfies spec.

Rabin Automata on Infinite k-ary Trees

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

- Σ: finite alphabet
- S: finite state set
- $S_0 \subseteq S$: initial state set
- ρ: transition function
  - $\rho \ : \ S \times \Sigma \rightarrow 2^{S^k}$
- α: acceptance condition
  - $\alpha = \{(G_1, B_1), \ldots, (G_l, B_l)\}$, $G_i, B_i \subseteq S$
  - *Acceptance*: along every branch, for some $(G_i, B_i) \in \alpha$, G_i is visited infinitely often, and B_i is visited finitely often.

Emptiness of Tree Automata

Emptiness: $L(A) = \emptyset$

Emptiness of Automata on Finite Trees: PTIME test (Doner, 1965)

Emptiness of Rabin Automata on Infinite Trees: Difficult

- Rabin, 1969: non-elementary
- Hossley+Rackoff, 1972: 2EXPTIME
- Rabin, 1972: EXPTIME
- Emerson, V.+Stockmeyer, 1985: In NP
- Emerson+Jutla, 1991: NP-complete

Rabin's Realizability Algorithm

REAL(φ):

- Construct Rabin tree automaton A_φ that accepts all winning strategy trees for spec φ.
- Check non-emptiness of A_φ.
- If nonempty, then we have realizability; extract strategy from non-emptiness witness.

Complexity: non-elementary

Reason: A_φ is of non-elementary size for spec φ in MSO.

Post-1972 Developments

- Pnueli, 1977: Use LTL rather than MSO as spec language.
- V.+Wolper, 1983: Elementary (exponential) translation from LTL to automata.
- Safra, 1988: Doubly exponential construction of tree automata for strategy trees wrt LTL spec (using V.+Wolper).
- Rosner+Pnueli, 1989: 2EXPTIME realizability algorithm wrt LTL spec (using Safra).
- Rosner, 1990: Realizability is 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.