Model Checking for Database Theoreticians

Moshe Y. Vardi

Rice University

Engines of Progress: Semiconductor Technology

Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

Result: Cost of memory and MIPS dropped roughly five orders of magnitude (10^5) over the last 30 years.

Semiconductor industry 10-year outlook: *there is no physical barrier to the transistor effect in silicon being the principal element in the semiconductor industry to the year 2015*.

But: Will the current business model for the semiconductor industry be viable until 2015?

A Major Challenge: design productivity crisis

- complexity growth rate: 60% per year
- Productivity growth rate: 20% per year

Critical need: better design tools

Design Verification

A watershed event: Pentium FDIV bug, 1995

- Bug would result in occasional inaccuracies when doing floating-point arithmetic.
- Eventually Intel promised to replace all Pentiums with the fixed chip.
- Cost to Intel: $500M.

Verification methodology:

- *Traditional*: simulation on carefully chosen test sequences
- *New*: formal verification of entire state space

Formal Verification

- **Theorem proving**: formally prove that hardware is correct
  - requires a large number of expert users
  - application cycle slower than design cycle
- **Model checking**: uncommonly effective debugging tool
  - a systematic exploration of the design state space
  - good at catching difficult “corner cases”

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
- $\pi : W \rightarrow Powerset(Prop)$: 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: $w_0, w_1, w_2, \ldots$

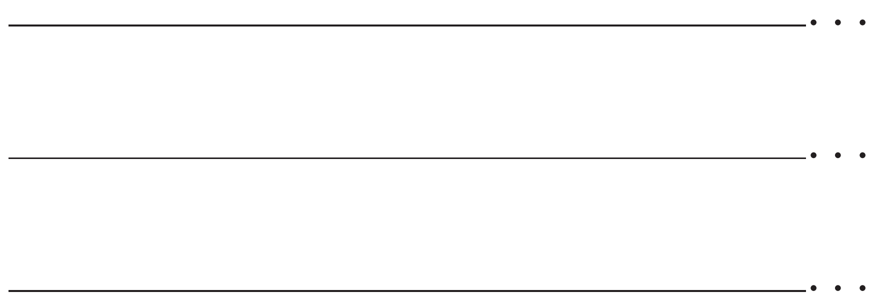
- $w_0 \in I$
- $(w_i, w_{i+1}) \in E$ for $i = 0, 1, \ldots$

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.



Algorithmic Foundations

Basic Graph-Theoretic Problems:

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

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

Note: These paths may correspond to error traces.

- *Deadlock*: A finite path from I to a state in which both $write_1$ and $write_2$ holds.
- *Livelock*: An infinite path from I along which snd holds infinitely often, but 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

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

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.

Semantics

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

Automata on Finite Words

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

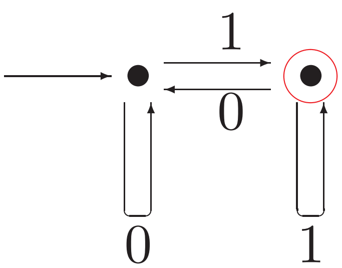
- *Alphabet*: Σ
- *States*: S
- *Initial states*: $S_0 \subseteq S$
- *Transition function*: $\rho : S \times \Sigma \to 2^S$
- *Accepting states*: $F \subseteq S$

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: $s_n \in F$.



– ends with 1's

Automata on Infinite Words

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

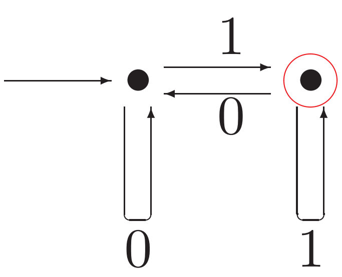
- *Alphabet*: Σ
- *States*: S
- *Initial states*: $S_0 \subseteq S$
- *Transition function*: $\rho : S \times \Sigma \to 2^S$
- *Accepting states*: $F \subseteq S$

Input word: $a_0, a_1, \ldots$

Run: $s_0, s_1, \ldots$

- $s_0 \in S_0$
- $s_{i+1} \in \rho(s_i, a_i)$ for $i \geq 0$

Acceptance: F visited infinitely often



– infinitely many 1's

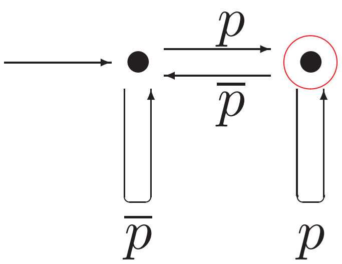
Temporal Logic vs. Automata

Paradigm: Compile high-level logical specifications into low-level finite-state language

The Compilation Theorem: V.&Wolper, 1983

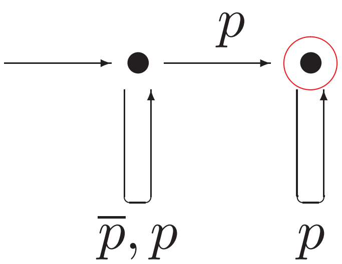
Given an LTL formula ϕ, one can construct an automaton A_ϕ such that a computation σ satisfies ϕ if and only if σ is accepted by A_ϕ. Furthermore, the size of A_ϕ is at most exponential in the length of ϕ.

always eventually p:



– infinitely many p's

eventually always p:



– finitely many $\overline{p}$'s

Model Checking

The following are equivalent:

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

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

Intuition: $A_{\neg\phi}$ 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 $\phi_1, \cdots, \phi_n$. It suffices to check that M satisfies ϕ_i for $1 \leq i \leq n$.

Moral: There is life after exponential explosion.

The real problem: too many design states – symbolic methods needed

Industrial Acceptance

Industrial benefits:

- reduced time to market
- increased design integrity

Acceptance by semiconductor industry: AT&T, Cadence, Fujitsu, HP, IBM, Intel, Lucent, Motorola, NEC, SGI, Siemens, Sun

Academic Tools:

- UC Berkeley: VIS
- CMU: SMV

Industrial tools:

- Cadence: FormalCheck
- IBM: RuleBase
- Intel: Forecast
- Lucent: SPIN

What about Software?

Data vs. Control:

- *Data intensive*: input/output transformation – scientific programs
- *Control intensive*: communication, coordination, synchronization, etc. – reactive programs

Critical Observation: Control-intensive (as opposed to data-intensive) programs often have finite (though often exceedingly large) state spaces – *model checking is applicable*.

Example: TCAS (Traffic Alert and Collision Avoidance System) - Univ. of Wash.

Obstacle: Recursive procedures – control space is infinite.

Modeling Infinite-State Procedures

Rewrite Systems: $\mathcal{R} = (V, R, x_0)$

- V: finite alphabet
- R: finite rewrite-rule set
- $x_0 \in V^*$: initial word

$\mathcal{R}$ induces transition system $M_{\mathcal{R}} = (W_{\mathcal{R}}, I_{\mathcal{R}}, E_{\mathcal{R}})$

- W_R: words in V^* derived from x_0
- $I_{\mathcal{R}} = \{x_0\}$
- $(x, y) \in E_{\mathcal{R}}$ if $x \overset{R}{\leadsto} y$

Intuition: model program step by rewrite step

Modeling Recursive Calls

Context-Free Systems: $\mathcal{R} = (V, R, x_0)$

Context-Free Rewrite Rule: $a \rightarrow v$

- $a \in V$, $x \in V^*$

Rewriting: $az \rightsquigarrow xz$ (only leftmost symbol is rewritten)

Intuition:

- $(p, s) \rightarrow (q, t)(p, s)$: procedure p in state s calling procedure q in state t.
- $(p, s) \rightarrow (p, t)$: procedure p in state s changes its state to t.
- $(p, s) \rightarrow \epsilon$: procedure p in state s returns

Context-Free System: Example

Remove Add

ate To state

$$\langle q_0, A, AB, q_0\rangle, \ \langle q_0, A, \epsilon, q_0\rangle, \ \langle q_0, B, \epsilon, q_0\rangle$$

$$\begin{tikzcd}
q_0A \arrow[r] \arrow[d] & q_0AB \arrow[r] \arrow[d] & q_0ABB \arrow[r] \arrow[d] & q_0ABBB \arrow[r, dashed] \arrow[d] & {} \arrow[d, phantom] \\
q_0 & q_0B \arrow[l] & q_0BB \arrow[l] & q_0BBB \arrow[l] & {} \arrow[l, dashed]
\end{tikzcd}$$

The Other Components

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

- *Fairness*:
  - NFA $\mathcal{A}_F$: fair states – $L(A_F)$
- *Observations*:
  - NFA $\mathcal{A}_p$ for $p \in Prop$ – p holds at states in $L(A_p)$.
  - $\pi(x) = \{p \in Prop : x \in L(A_p)\}$

Crux: Finite representation of infinite-state systems (cf. constraint databases)

size:

$$(\Sigma_{a \to x \in \mathcal{R}} |x| + 1) + (\Sigma_{p \in Prop} |\mathcal{A}_p|) + |\mathcal{A}_F|$$

Decidability vs. Undecidability

Classical Result, Bar Hillel, Perles and Shamir, 1961: The *universality problem* for context-free grammars is undecidable.

- *Input*: CFG G
- *Problem*: Does $L(G) = \Sigma^*$?

Modern Result, Muller and Schupp, 1985: The monadic 2nd-order theory of context-free systems is decidable.

- *Input*: Context-free system $\mathcal{R}$, monadic 2nd-order sentence φ
- *Problem*: Does $M_{\mathcal{R}} \models \varphi$?

Decidability vs. Undecidability

Crux:

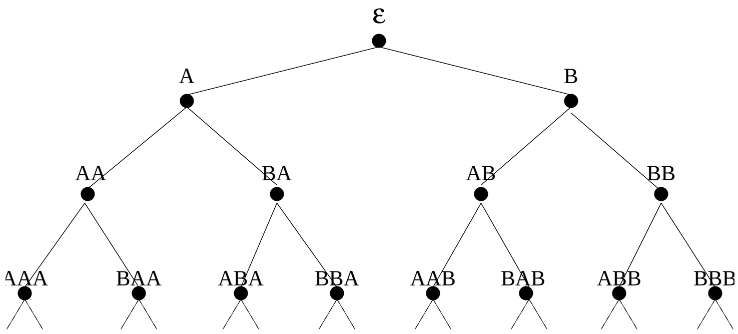
- Derivations trees are regular

  $\mapsto$ *decidability* at tree level.

- Frontiers of derivations trees are not regular

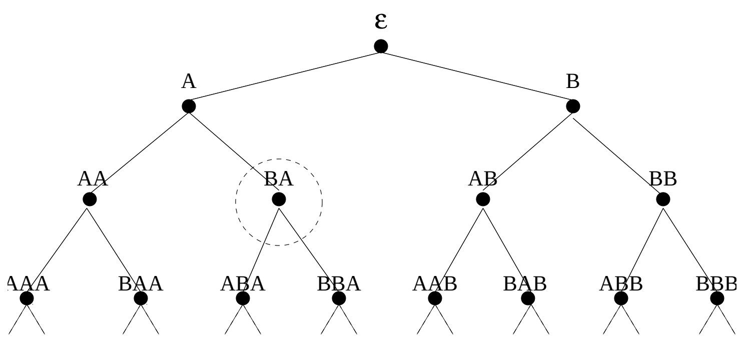
  $\mapsto$ *undecidability* at word level.

Context-Free Systems: Configuration Tree

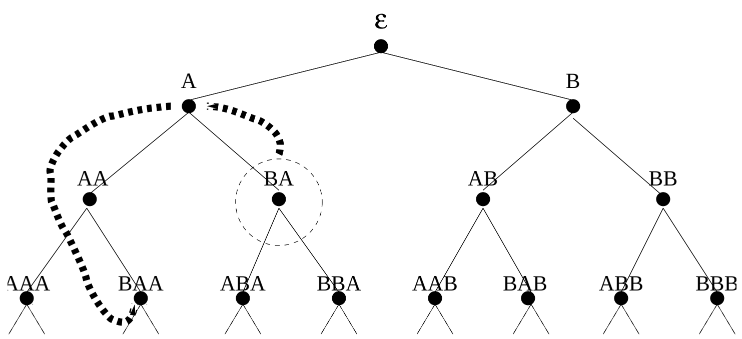


Navigating on the Configuration Tree

Source Configuration



Target Configuration



Automata on Infinite Binary Trees

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

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

Input tree: $\tau : \{0,1\}^* \to \Sigma$

Run: $r : \{0,1\}^* \to S$

- $r(\epsilon) \in S_0$
- $(r(0x), r(1x)) \in \rho(r(x), \tau(x))$ for $x \in \{0,1\}^*$

Acceptance: F visited infinitely often along every branch of r.

Automata on Infinite Binary Trees

Parity Tree Automaton: $A = (\Sigma, S, S_0, \rho, G)$

- *Accepting states*: $G = (F_1, F_2, \ldots, F_k)$
  - $F_0 \subseteq F_1 \subseteq \ldots \subseteq F_k = S$
  - k – index of A

Input tree: $\tau : \{0,1\}^* \to \Sigma$

Run: $r : \{0,1\}^* \to S$

Acceptance: along every branch of r, the minimal i such that F_i is visited infinitely often is even.

Parity Tree Automata: Example

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

- $\Sigma = \{0, 1\}$
- $S = \{0, 1\}$
- $S_0 = \{0, 1\}$
- $\rho(s, a) = \{(a, a)\}$
- $G = (\{1\}, \{0, 1\})$

Language: $L(A)$ - trees with finitely many 1's.

Nonemptiness and Satisfiability

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

Nonemptiness Tests:

- Rabin, 1969: Nonemptiness of parity tree automata is decidable.
- V.&Wolper, 1986: Nonemptiness of Büchi tree automata can be checked in quadratic time.
- Pnueli&Rosner, Emerson&Jutla, 1988-9: Nonemptiness of parity tree automata with size n and index k can be checked in time $n^{O(k)}$.

Logic and Automata: Rabin, 1969

An MSO sentence φ can be translated into a parity tree automaton A_φ such that $models(\varphi) = L(A_\varphi)$

Corollary: Satisfiability of MSO over trees is decidable.

Logic and Automata

Basic Observations:

- All program logics, e.g., LTL, CTL*, μ-calculus, are over computation trees.
- All program logics can be expressed in MSO.
- Therefore, Rabin's Theorem yields decidability results for verification problem.
- *But*: the translation from *MSO* to parity tree automata is nonelementary.

Conclusion: Seek direct, elementary translations from program logics to automata, e.g., earlier translation of LTL to Büchi automata on infinite words.

Observation: The more expressive the automata, the easier the translation.

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)

Alternating Büchi Automata on Words

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

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

Game:

- Infinite board: $a_0, a_1 \ldots$
- Positions: $S \times \{0, 1, \ldots\}$
- Initial position: $(s_0, 0)$, $s_0 \in S_0$ - chosen by automaton
- 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 plays go through infinitely many positions (s', i) with $s' \in F$

Acceptance: Automaton has a winning strategy.

Example

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

- States: master and slave

- $\rho(m,1) = m$

- $\rho(m,0) = m \wedge s$

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

- $\rho(s,0) = s$

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.

Two-Way Alternating Parity Automata on Trees

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

- *Alternating transition relation*:

$\rho : S \times \Sigma \rightarrow \mathcal{B}^+(D \times S)$

- $D = \{-1, \epsilon, 0, 1\}$

Example: $\rho(s, a) = (-1, t) \vee ((\epsilon, t) \wedge (0, s))$

Intuition: D – directions

- -1 – upwards
- ϵ – in place
- 0 – left
- 1 – right

Expressiveness, V., 1998: There is an exponential translation from 2-way alternating parity tree automata to 1-way nondeterministic parity automata.

Corollary: Exponential time nonemptiness test – no implementation so far.

This Is A Database Conference!

Motivation: Web services

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

- $\pi : W \rightarrow RelDB$: associates a relational database with each state

Issues:

- Finite or infinite state space
- Bounded or unbounded database cardinality
- Definition of transition relation
- Specification language

Decidability vs. Undecidability:

- Abiteboul, Vianu, Fordham, Yesha, 2000
- Spielmann, 2003
- Deutch, Sui, Vianu, 2004

Decidability

Universal LTL: $\forall$LTL

- Relational atomic formulas
- Boolean and temporal connectives
- All variables are implicitly universally quantified

Example:
$always(\textit{request}(x) \rightarrow eventually\ \textit{grant}(x))$

Spielmann, 2003: Model checking of relational transition system wrt $\forall$LTL is decidable

- by reduction to the existential fragment of transitive-closure logic.

A more direct approach: treat relational atomic formulas as atomic propositions and apply automata-theoretic approach.

- Chomicki&Niwinski, 1995: temporal constraints
- Bacchus&Kabanza, 1996: temporal goals

Conclusions

Key points:

- *Model checking* = *Graphs* + *Logic* + *Automata*
- Technology has great potential for the analysis of discrete systems.

Status Report:

- *Finite-state systems*: mature theory and practice
- *Recursive systems*: mature theory, immature practice
- *Relational systems*: immature theory