AUTOMATA + LOGIC: A MATCH MADE IN HEAVEN

MOSHE Y. VARDI

RICE UNIVERSITY

MODEL CHECKING -
The Very Idea

What is model checking?

1. A formalism to express properties of transition systems.

2. An algorithm to check properties expressed in that formalism.

Crux: High-level formalism

Currently: Several MC frameworks (linear, branching, reactive, ...)

Needed: high-level framework for MC

Proposed Answer: **Automata**

TRANSITION SYSTEMS

$T = (W, W_0, R, \pi)$

W: state set

$W_0 \subseteq W$: initial states

$R \subseteq W^2$: transition relation

$\pi : W \to 2^{Prop}$: observation function

Finite run: $w_0, w_1 \dots w_n$

$w_0 \in W_0$, $(w_i, w_{i+1}) \in R$

Finite trace: $\pi(w_0), \pi(w_1) \dots \pi(w_n)$

$FTraces(T) = \{t :$ t is a finite trace of $T\}$

Comment: This is the linear-time view

Linear Temporal Logic

LTL

- Atomic propositions
- Boolean connectives
- Temporal connectives: next, always, until

semantics: $\tau \models \varphi$

$$\underbrace{\tau}_{\text{finite trace}} \models \underbrace{\varphi}_{\text{LTL formula}}$$

$$\text{f.models}(\varphi) = \{\tau : \tau \models \varphi\}$$

correctness: $T \models \varphi$

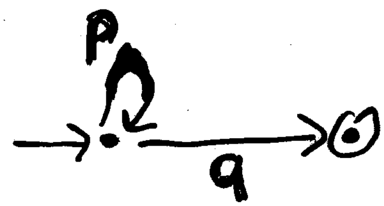
$$\Updownarrow$$

$$\text{FTraces}(T) \subseteq \text{fmodels}(\varphi)$$

LTL vs. Automata

Thm: Given an LTL formula φ, there is an NFA A_φ, of size $2^{O(|\varphi|)}$, such that finmodels$(\varphi) = L(A_\varphi)$.

Example: $p \, \mathbf{U} \, q$



Example: always p

Complexity: V.+Wolper

$|A_\varphi| = 2^{O(|\varphi|)}$

History

Words as relational structures:

$w = a_0 \ldots\ldots a_{n-1} \in \Sigma^*$

$\mathcal{M}_w = (\{0, \ldots, n-1\}, succ, P_{a_1} \ldots P_{a_k})$

Büchi - Elgot - Trakhtenbrot, ~1960

Regular $\equiv$ Monadic second-order Logic on words

$\downarrow$ NFA

$\downarrow$ set quantification

$$\text{Regular} \underset{\text{non-el.}}{\overset{\text{poly}}{\rightleftarrows}} \text{MSO}$$

Emptiness

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

.nite alphabet

tate set

initial states

MODEL CHECKING

T. F. A. E.:

- $T \models \varphi$
- $FTraces(T) \subseteq \cancel{f}models(\varphi)$
- $FTraces(T) \cap \cancel{f}models(\neg\varphi) = \emptyset$
- $FTraces(T) \cap L(A_{\neg\varphi}) = \emptyset$
- $L(T \times A_{\neg\varphi}) = \emptyset$

Algorithm:

1. construct $A_{\neg\varphi}$
2. construct $T \times A_{\neg\varphi}$
3. check $L(T \times A_{\neg\varphi}) = \emptyset$

Complexity: time : $|T| \cdot 2^{O(|\varphi|)}$

Going to The Limit

Nonterminating systems:

$T = (W, W_0, R, \pi)$

Run: $w_0, w_1, \ldots \quad \in W^\omega$

$w_0 \in W_0,\ (w_i, w_{i+1}) \in R$

Trace: $\pi(w_0), \pi(w_1), \ldots$

$Traces(T) = \{\tau : \tau$ is a trace of $T\}$

LTL semantics: $\tau \models \varphi$

trace LTL formula

$models(\varphi) = \{\tau : \tau \models \varphi\}$

correctness: $T \models \varphi$

$\Updownarrow$

$Traces(T) \subseteq models(\varphi)$

Needed theory of automata on infinite words

Büchi Automata

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

Σ: finite alphabet

S: state set

$S_0 \subseteq S$: initial states

$\rho \subseteq \Sigma \times S \times S$: transition relation

$F \subseteq S$: accepting states

Input word: $a_0, a_1, a_2, \ldots \in \Sigma^\omega$

Run: $r = s_0\ s_1\ s_2 \ldots \in S^\omega$

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

Limit: $\lim(r) = \{s : s$ occurs i.o. in $r\}$

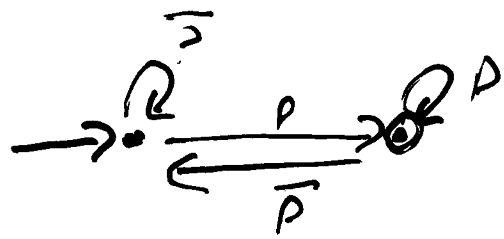
Acceptance: $\lim(r) \cap F \neq \emptyset$

Language: $L_\omega(A) = \{w : A$ accepts $w\}$

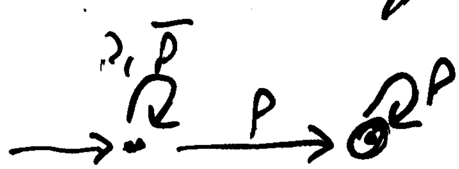
LTL vs. AUTOMATA

Thm: Given an LTL formula φ there is a Büchi automaton A_φ such that models$(\varphi) = L_\omega(A_\varphi)$.

Example: always eventually p



Example eventually always p



omplexity: V. + Wolper

$|A_\varphi| = 2^{O(|\varphi|)}$

History

Inf. words as relational structures

$w = a_0 a_1 a_2 \ldots \quad \in \Sigma^\omega$

$M_w = (\{0,1,2,\ldots\}, succ, P_{a_1} \ldots P_{a_k})$

Büchi, 1962:

ω-regular = MSO on inf. words

||

Büchi automata

Büchi Automata $\underset{\text{non-el.}}{\overset{\text{poly}}{\rightleftarrows}}$ MSO

Hard: complementation

Emptiness

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

graph of A:

$$G_A = (S, E_A), \quad E_A = \{(s,t) : \underbrace{(a,s,t) \in \rho}_{\text{for some } a \in \Sigma}\}$$

Lemma: $L_\omega(A) \neq \emptyset$ iff there is a path in G_A from S_0 to a cycle that visits F.

Cor: Büchi emptiness can be checked in linear time.

Proof: Use DFS.

MODEL CHECKING

T. F. A. E.:

- $T \models \varphi$
- $Traces(T) \subseteq models(\varphi)$
- $Traces(T) \cap models(\neg\varphi) = \emptyset$
- $Traces(T) \cap L_\omega(A_{\neg\varphi}) = \emptyset$
- $L_\omega(T \times A_{\neg\varphi}) = \emptyset$

Algorithm

1. construct $A_{\neg\varphi}$
2. construct $T \times A_{\neg\varphi}$
3. check $L_\omega(T \times A_{\neg\varphi}) = \emptyset$

- complexity: $|T| \cdot 2^{O(|\varphi|)}$ time

nching Off

?

?

$$\cdot) = \;\to \rho \to \rho \to \rho \to \rho \to \cdots$$
$$\to \rho \to \bar{\rho} \to \bar{\rho} \to \bar{\rho} \to \cdots$$
$$\to \rho \to \rho \to \bar{\rho} \to \bar{\rho} \to \cdots$$
$$\vdots$$

$$\tau) = \quad \begin{array}{l} \rho \to \bar{\rho} \to \bar{\rho} \cdots \\ \downarrow \\ \rho \to \bar{\rho} \to \bar{\rho} \cdots \\ \downarrow \\ \rho \to \bar{\rho} \to \bar{\rho} \cdots \\ \vdots \end{array}$$

Branching Temporal Logic

LTL	CTL
next	$\forall$ next $\exists$ next
always	$\forall$ always $\exists$ always
eventualy	$\forall$ eventually $\exists$ eventually

Example $\forall$ always (p $\rightarrow$ $\exists$ eventually q)

$$\text{models}(\varphi) = \{ T : T \models \varphi \}$$

correctness: $T \models \varphi$

$$\Updownarrow$$

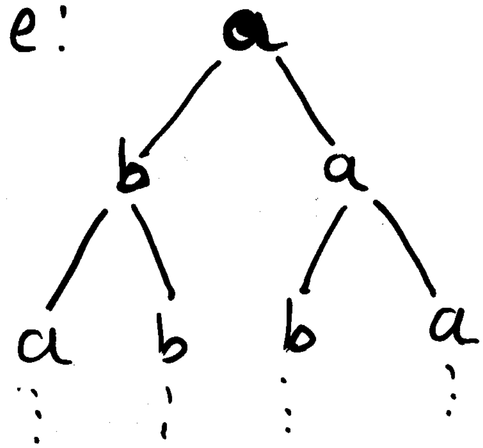
$$\text{tree}(T) \in \text{models}(\varphi)$$

Büchi Tree Automata

Rabin 1969 $A = (\Sigma, S, S_0, \rho, F)$

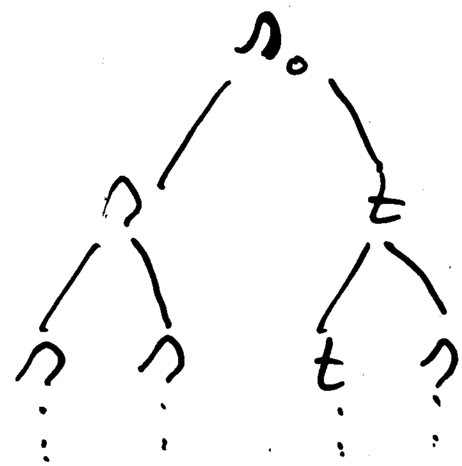
$\rho \subseteq \Sigma \times S^3$

Input (binary) tree:



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

Run:



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

$s_0 \in S_0$ $(a, s_0, s, t) \in \rho$

Formally: $r(\varepsilon) \in S_0$, $(\tau(x), r(x), r(x0), r(x1)) \in \rho$

Acceptance: F visited i.o. along every branch

CTL vs AUTOMATA

<u>Thm</u>: Given a CTL formula φ, there is a Büchi tree automaton A_φ s.t. binary-model$(\varphi) = L_\omega(A_\varphi)$

Complexity: V. + Wolper

$$|A_\varphi| = 2^{O(|\varphi|)}$$

Application: φ is satisfiable iff $L_\omega(A_\varphi) \neq \emptyset$.

Emptiness: V. + Wolper: quadratic fixpoint algorithm

<u>Corollary</u>: CTL-SAT $\in$ EXPTIME

shown first by Emerson + Halpern, 1985

matching l.b.: Fischer + Ladner, 1978

History

Finite trees:

Donner, Thatcher-Wright, ~1968:

Automata on finite tress ≡ MSO on finite trees

Rabin, 1969:

Automata on infinite trees ≡ MSO on infinite tress

<u>Hard</u>: complementation

CTL MODEL CHECKING

T. F. A. E.:

- $T \models \varphi$
- $tree(T) \in models(\varphi)$
- $tree(T) \in L_w(A_\varphi)$
- $L_w(T \times A_\varphi) \neq \emptyset$

Algorithm:

1. construct A_φ
2. construct $T \times A_\varphi$
3. check $L_w(T \times A_\varphi) \neq \emptyset$

Complexity: $|T|^2 \cdot 2^{O(|\varphi|)}$ time

Baaaad!!! CTL MC is in $O(|T| \cdot |\varphi|)$ time

Another Look at Tree Automata

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

$\rho \subseteq \Sigma \times S^3$ - transition relation

Equivalently: $\rho : \Sigma \times S \rightarrow 2^{S^2}$

Example: ρ: (a, s_0, s_1, s_2)
(a, s_0, s_2, s_3)

$\downarrow$

$\rho(a, s_0) = \{(s_1, s_2), (s_2, s_3)\}$

Equivalently:

$\rho(a, s_0) = [\langle 0, s_1 \rangle \wedge \langle 1, s_2 \rangle] \vee$
$[\langle 0, s_2 \rangle \wedge \langle 1, s_3 \rangle]$

Directions: 0 - left

1 - right

Alternating Automata

Idea: mix $\wedge$ and $\vee$ freely

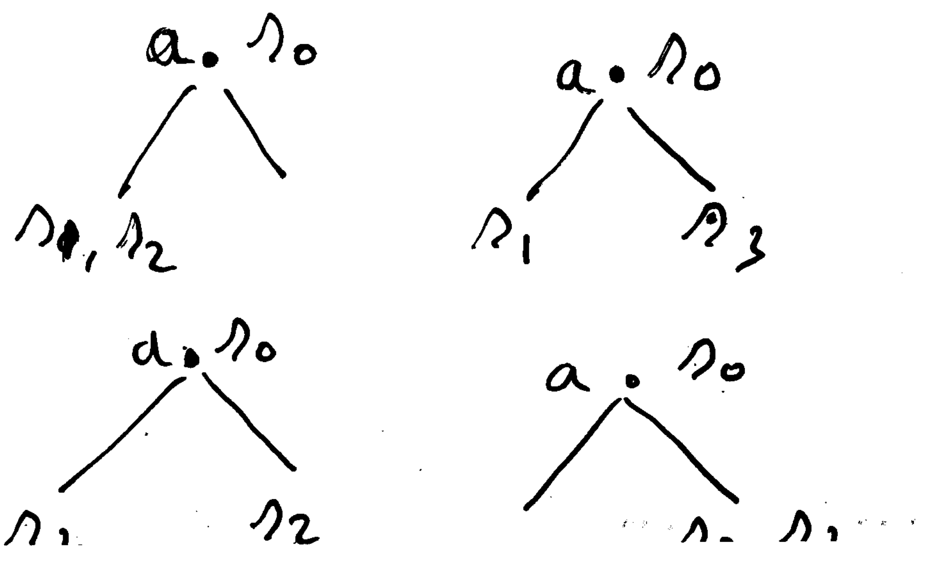
Example: $\rho(a, s_0) = [\langle 0, s_1\rangle \vee \langle 1, s_2\rangle]$
$\wedge\, [\langle 0, s_2\rangle \vee \langle 1, s_3\rangle]$

generally: $B^+(X)$ = positive Boolean formulas over X

Transition function:
$$\rho : \Sigma \times S \to B^+(\{0,1\} \times S)$$

Semantics: $\vee$ – existential choice
$\wedge$ – universal choice

Example:



ABT: Alternating Büchi automata on Trees

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

$\rho : \Sigma \times S \to B^+(\{0,1\} \times S)$

Emptiness: 2-player game –

$\vee$-player vs. $\wedge$-player

complexity: EXPTIME complete

(u.b.: Müller + Schupp, l.b.: Seidel)

Thm: [Müller + Schupp], 1988

Given a CTL formula φ, there is an ABT A^a_φ, of size $O(|\varphi|)$, such that $models(\varphi) = L_\omega(A^a_\varphi)$.

Alternation:

- Higher succinctness
- Higher complexity

odel checking

$) \in models(\varphi)$

$T) \in L_\omega(A^a_\varphi)$

$T \times A^a_\varphi) \neq \emptyset$

n:

uct A^a_φ

ct $T \times A^a_\varphi$

$L_\omega(T \times A^a_\varphi) \neq \emptyset$

$T \times A^a_\varphi$ is a ~~letter~~ auto

μ-Calculus

"Mother of all program logic"

- Propositial logic
- Basic modalities: $\exists$ next, $\forall$ next
- Least and greatest fixpoints: μ, ν

Expressiveness:

- More expressive than CTL*
- More expressive than ABT

Paradigm: Logic $\longmapsto$ Automata

Moral: More expressive automata needed!

APT: Alternating Parity automata on Trees

$A = (\Sigma, S, s_0, \rho, \bar{P})$

$\bar{P} = \langle F_1, F_2, \dots, F_k \rangle$, $F_i \subseteq S$

$F_i \subseteq F_{i+1}$

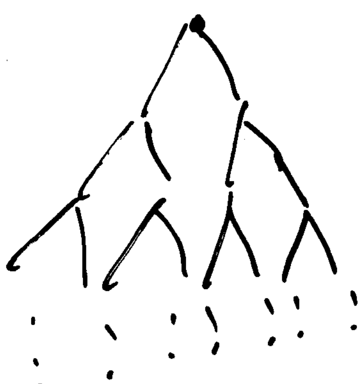
$r = s_0, s_1, \dots \quad \in S^\omega$

$\text{index}(r) = \min_i \; \lim(r) \cap F_i \neq \emptyset$

"minimal index of set visited by r infinitely often"

r satisfies $\bar{P}$: index(r) is even

Run tree satisfies $\bar{P}$: all branches satisfy $\bar{P}$



APT Emptiness

Recall:

- Nondeterministic automata:

  1-letter emptiness = 2-letter emptiness

- Alternating automata:

  1-letter emptiness $\leq$ 2-letter emptiness

APT:

2-letter emptiness: EXPTIME-complete
[Muller+Schupp, 1995]

1-letter emptiness: NP $\cap$ co-NP
[Emerson+Jutla, 1993]

open question: precise complexity

Automata Theory Since 1959

NFA – 1959

Büchi Automata – 1962

Büchi Tree Automata - 1969

Alternating Büchi Tree Automata – 1988

Alternating Parity Tree Automata – 1990

Two-way Alternating Parity Tree Automata - 1998

Trend:

- Increased succinctness
- Increased expressiveness
- Harder emptiness problems

AUTOMATA AS TOOLS

what is computer science?

- Fight complexity with abstraction
- Do not forget efficiency issues

Algorithmic Research:

Quest for powerful abstractions

- BFS
- DFS
- set constraints
- CFL reachability

⋮

Our proposal: automata emptiness