ALTERNATION

MOSHE Y. VARDI

RICE UNIVERSITY

COMPLEXITY THEORY

Key CS question: What can be automated?

Next question: How hard it is to automate it?

Hardness: measure computational resources

- time
- space

Hierarchy:

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

NONDETERMINISM

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

P vs NP:

- PTIME: can be solved in polynomial time
- NPTIME: solution can be checked in poly time

Hierarchy:

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

CO-NONDETERMINISM

Intuition:

- nondeterminism: check solutions
- co-nondeterminism: check counterexamples

Example:

- satisfiability: nondet.
- validity : co-nondet.

Hierarchy:

$$
\begin{array}{ccc}
 & \text{LOGSPACE} & \\
 & \cap\!| & \\
\text{NLOGSPACE} & = & \text{CO-NLOGSPACE} \\
 & \cap\!| & \\
 & \text{PTIME} & \\
 & \cap\!| & \\
\text{NPTIME} & & \text{CO-NPTIME} \\
 & \cap\!| & \\
 & \text{PSPACE} & \\
 & \| & \\
\text{NPSPACE} & = & \text{CO-NPSPACE} \\
 & |\!\cap & \\
 & \text{EXPTIME} &
\end{array}
$$

Alternation

Determinism vs. (co) non determinism

Unique transition vs. multiple transition

- non determinism: existential choice
- co-non determinism: universal choice

Alternation: [CKS, 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: propositional formula over $x_1, \ldots, x_n$

Decision Problems:

SAT: 1. Is φ satisfiable? NPTIME

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

VALID: 2. Is φ valid? co-NPTIME

check that for all truth assignments τ we have $\tau \models \varphi$

QBF: 3. 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 ... φ holds.

Alternation and Games

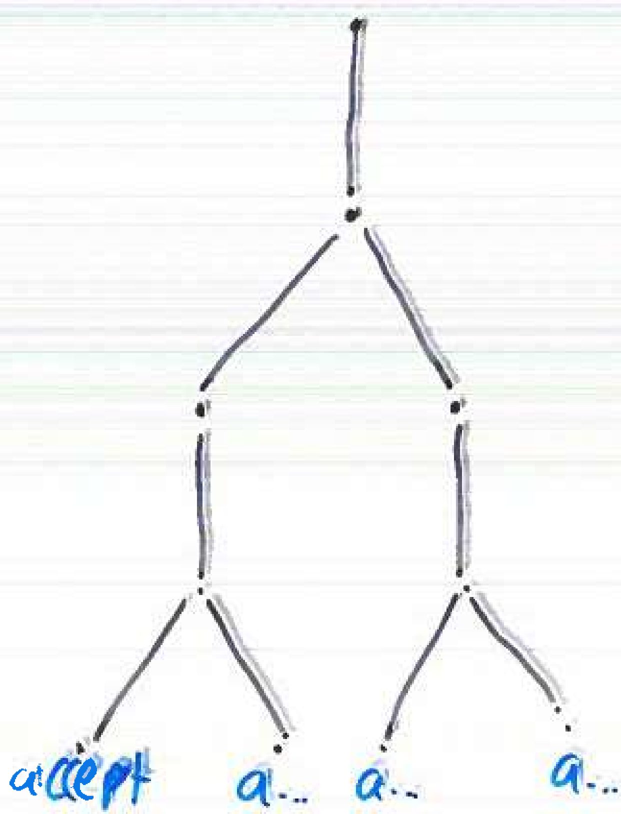
Players: $\exists$-player, $\forall$-player

$\exists$-state: $\exists$-player chooses transition

$\forall$-state: $\forall$-player chooses transition

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

Run: strategy tree for $\exists$-player



Alternation and Complexity

Upper Bounds:

- $\mathrm{ATIME}[f(n)] \subseteq \mathrm{SPACE}[f^2(n)$
  - Intuition: search for strategy tree recursively
- $\mathrm{ASPACE}[f(n)] \subseteq \mathrm{TIME}[2^{f(n)}]$
  - Intuition: compute set of winning configurations bottom-up.

Lower Bounds:

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

CONSEQUENCES

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

Applications:

- "In APTIME" → "In PSPACE"
- "APTIME-hard" → "PSPACE-hard"

Example: QBF

- Natural algorithm is in APTIME
  → "In PSPACE"
- Prove APTIME-hardness a la Cook
  → "PSPACE-hard"

∴ PSPACE-complete

MODAL LOGIC

Syntax:

- propositional logic
- $\Diamond\varphi$, $\Box\varphi$

Proviso: positive normal form

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

- W : worlds
- $R \subseteq W^2$: possibility relation
- $\pi : W \to 2^{Prop}$: truth assignment

$R(w) = \{u : R(w,u)\}$

Semantics:

- $M, w \models p$ if $p \in \pi(w)$

  $\neg p$ — $p \notin \pi(w)$

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

  $\Box\varphi$ ...... for all ...

MODEL CHECKING

Input:

- φ : modal formula
- $\mathcal{M} = (W, R, \pi)$: kripke structure
- $w \in W$: world

problem: $\mathcal{M}, w \models \varphi$?

Algorithm: $mc(\mathcal{M}, w, \varphi)$

case:

$\varphi = p$: return $p \in \pi(w)$

$\varphi = \neg p$: return $p \notin \pi(w)$

$\varphi = \alpha_1 \vee \alpha_2$: ($\exists$-branch) return $mc(\mathcal{M}, w, \alpha_i)$

$\varphi = \alpha_1 \wedge \alpha_2$: ($\forall$-branch) return $mc(\mathcal{M}, w, \alpha_i)$

$\varphi = \Diamond \alpha$: ($\exists$-branc) return $mc(\mathcal{M}, u, \alpha)$ for $u \in R(w)$

$\varphi = \Box \alpha$: ($\forall$-branch) return $mc(\mathcal{M}, u, \alpha)$ for $u \in R(\alpha)$

esac.

Correctness: immediate

COMPLEXITY ANALYSIS

Algorithm's state: $(\mathcal{M}, u, \varphi)$

- $\mathcal{M}$: fixed
- u: $O(\log |\mathcal{M}|)$ bits
- φ: $O(\log |\varphi|)$ bits

Conclusion: $ASPACE(\log |\mathcal{M}| + \log |\varphi|)$

$\therefore$ $MC \in ALOGSPACE = PTIME$

SATISFIABILITY

Proviso: no $\Diamond$, no PNF, no $\vee$

- $sub(\varphi)$: subformulas of φ and their negation ($\neg\neg\omega = \omega$)
- valuation for φ: $\alpha : sub(\varphi) \to \{0,1\}$
  - $\alpha(\varphi) = 1$
  - $\alpha(\omega) = 1$ iff $\alpha(\neg\omega) = 0$
  - $\alpha(\omega_1 \wedge \omega_2) = 1$ iff $\alpha(\omega_1) = 1$ & $\alpha(\omega_2) = 1$
- $\Box(\varphi) = \{\Box\omega : \Box\omega \in sub(\varphi)\}$

Lemma: φ is satisfiable iff it has a valuation α such that for all $\Box\psi \in \Box(\varphi)$ with $\alpha(\Box\psi) = 0$ we have that $\neg\psi \wedge \bigwedge_{\substack{\Box\omega \in \Box(\varphi) \\ \alpha(\Box\omega) = 1}} \omega$ is satisfiable.

INTUITION

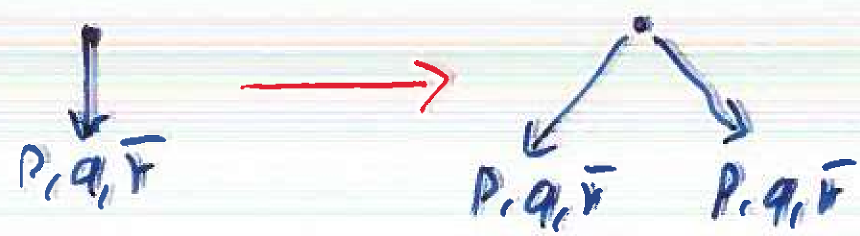
ONLY IF: $M, w \models \varphi$

$$\alpha(\varphi) = 1 \iff M, w \models \varphi$$

IF:

Fact: Modal logic is preserved under bisimulation

$\therefore$ children can be duplicated



Fact: $\neg \Box \varphi \equiv \Diamond \neg \varphi$

Crux: satisfy each diamond separately

ALGORITHM

K-SAT(φ)

$\exists$: select valuation α for φ

$\forall$: select $\Box\psi \in \Box(\varphi)$ with $\alpha(\Box\psi)=0$,

return K-SAT$\Big(\neg\psi \wedge \bigwedge_{\substack{\Box\omega\in\Box(\varphi)\\ \alpha(\Box\omega)=1}} \omega\Big)$

Complexity Analysis:

- Each step is in poly time
- Number of steps is linear

$\therefore$ K-SAT $\in$ APTIME = PSPACE

Remark:

- Above applies to modal logic K.
- Foundation for SAT-K.
- Can be adapted to other logics.

LOWER BOUND

Easy reduction from APTIME

- each 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 $\longrightarrow$ formulas of polynomial size

$\therefore$ SAT(K) is PSPACE-complete

Linear Temporal Logic

Syntax:

- propositional logic
- $\bigcirc\varphi$, $\varphi \, U \, \psi$, $\varphi \, V \, \psi$

proviso: positive normal form

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

- W: worlds
- $R: W \to W$: successor relation
- $\pi: W \to 2^{Prop}$: truth assignment

Semantics:

- $M, w \models \bigcirc\varphi$ if $M, R(w) \models \varphi$
- $M, w \models \varphi \, U \, \psi$ if

MODEL CHECKING

Input: • φ : LTL formula

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

• $w \in W$: world

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

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

case:

$\varphi = p$: return $p \in \pi(w)$

$\varphi = \neg p$: return $p \notin \pi(w)$

$\varphi = \varphi_1 \vee \varphi_2$: ($\exists$) return $MC(M, w, \varphi_i)$

$\varphi = \varphi_1 \wedge \varphi_2$: ($\forall$) return $MC(M, w, \varphi_i)$

$\varphi = \bigcirc\varphi$: return $MC(M, R(w), \varphi)$

$\varphi = \varphi_1 \mathbf{U} \varphi_2$: ($\exists$) return $MC(M, w, \varphi_2)$ or

($\forall$) return $MC(M, w, \varphi_1)$ and

return $MC(M, R(w), \varphi)$

$\varphi = \varphi_1 V \varphi_2$: . . .

esac

FROM FINITE TO INFINITE GAMES

Problem: algorithm does not terminate !!!

Solution: Back to games

- alternation is a finite game between $\exists$ and $\forall$
- here we need an infinite game
- $\exists$ can visit infinitely often only $\forall$-formulas

Büchi-alternation:

- infinite computations allowed
- on infinite computations $\exists$ needs to "force" i.m. good states.

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

$\therefore$ MC(LTL) $\in$ PTIME

SATISFIABILITY

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

Problems:

- How to combine infinite games with time-bounded games (to get PSPACE bound)
- Successors cannot be split

✓

$$\begin{tikzcd} \bullet\ \bigcirc\varphi_1, \bigcirc\varphi_2 \arrow[d] \\ \bullet\ \varphi_1, \varphi_2 \end{tikzcd}$$

✗

$$\begin{tikzcd} & \bullet\ \bigcirc\varphi_1, \bigcirc\varphi_2 \arrow[dl] \arrow[dr] & \\ \varphi_1 & & \varphi_2 \end{tikzcd}$$

AUTOMATA

Intuition: Automata = Games over a board

Nondeterministic automata:

1-player games

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

Σ: alphabet

S: states

$S_0 \subseteq S$: 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")

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

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

Acceptance: $s_n \in F$

ALTERNATING AUTOMATA

Alternating automata: 2-player games

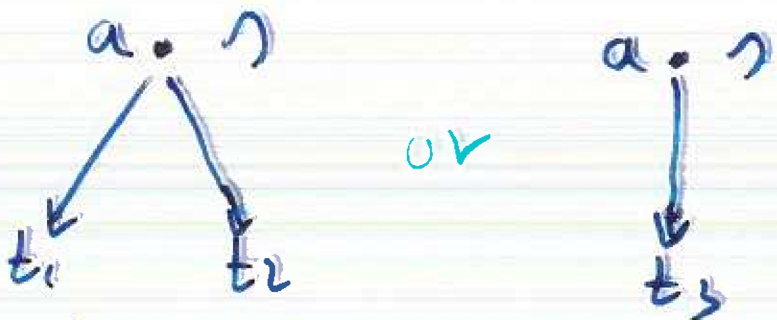
Nondeterministic transition:

$$\delta(q,a) = t_1 \vee t_2 \vee t_3$$

Alternating transition:

$$\delta(q,a) = (t_1 \wedge t_2) \vee t_3$$

"either both t_1 and t_2 accept or t_3 accept"



$\{t_1, t_2\} \models \delta(q,a)$ $\{t_3\} \models \delta(q,a)$

Alternating transition function:

$$\delta : S \times \Sigma \longrightarrow B^+(S)$$

positive Boolean formula

ALTERNATING AUTOMATA

BL'80: ("Boolean Automata")

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

Σ: alphabet

S: states

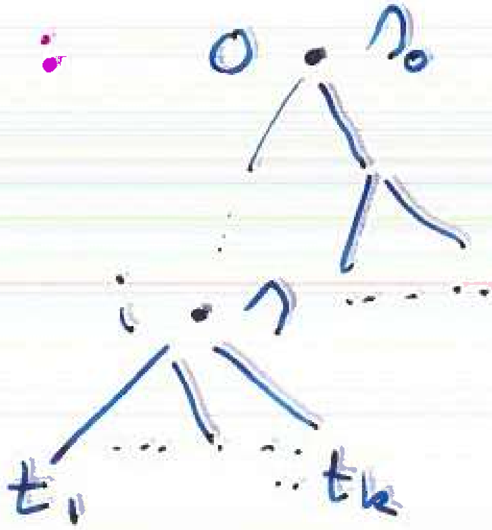
$s_0 \in S$: initial state

$\rho : S \times \Sigma \to B^+(S)$: alt. trans. function

$F \subseteq S$: accepting states

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

Run:



("winning strategy")

$\{t_1 \dots t_k\} \models \rho(s, a_i)$

EXPRESSIVE POWER

BL'80, CKS'81:

- Non det. automata = regular lang.
- Alt. automata = regular lang.

What is then the point?

Succinctness: exponential gap

- Translation from alt. automata to non det. automata is exponential.
- In the worst case this is the best one can do.

Büchi Automata

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

Σ: alphabet

S: states

$S_0 \subseteq S$: initial state

$\rho : S \times \Sigma \to 2^S$: trans. function

$F \subseteq S$: accepting states

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

Run : $s_0, s_1, s_2 \ldots$ ("infinite game")

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

Acceptance: F is visited i.o.

Alternating Büchi Automata

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

Σ: alphabet

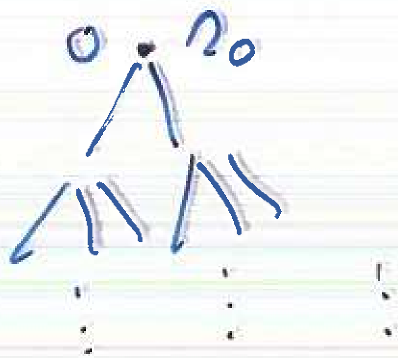
S: states

$s_0 \in S$: initial state

$\rho: S \times \Sigma \to B^+(S)$: alt. trans function

$F \subseteq S$: accepting states

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

Run:  ("winning strategy for infinite game")

Acceptance: infinite branches visit F i.o.

EXAMPLE

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

$$\Sigma = \{0, 1\}$$

$$S = \{s, t\}$$

$$s_0 = s$$

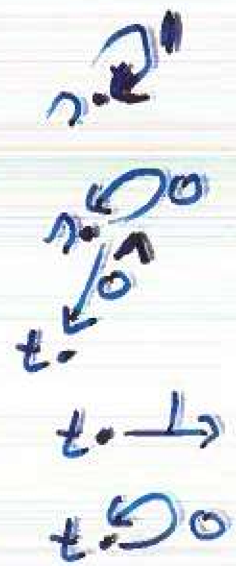
$$F = \{s\}$$

$$\rho(s, 1) = s$$

$$\rho(s, 0) = t \wedge s$$

$$\rho(t, 1) = \text{true}$$

$$\rho(t, 0) = t$$



- t is a subprocess that waits for 1.
- s is a master process that launches t.

$\therefore$ $L(A) =$ i.m. 1's.

EXPRESSIVE POWER

MH'84:

- Non det. Büchi automata = ω-regular lang.
- Alt. Büchi automata = ω-regular lang.

Succinctness: <u>exponential gap</u>

- Translation from alt. Büchi automata to nondet. Büchi automata is exponential
- In the worst case this is the best one can do

Back to Linear Temporal Logic

Temporal Model: $\sigma \in (2^{Prop})^{\omega}$

(Unwind temporal structures)

Temporal semantics:

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

From LTL to ABA:

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

$$V(\varphi) = \{\theta_1 V \theta_2 : \theta_1 V \theta_2 \in sub(\varphi)\}$$

- $\rho(a, p) =$ true if $p \in a$
- $\rho(a, p) =$ false if $p \notin a$
- $\rho(a, \neg p) =$ true if $p \notin a$
- $\rho(a, \neg p) =$ false if $p \in a$
- $\rho(a, \theta_1 \underset{\vee}{\wedge} \theta_2) = \rho(a, \theta_1) \underset{\vee}{\wedge} \rho(a, \theta_2)$

TEMPORAL CONNECTIVES

- $\rho(a, \bigcirc\varphi) = \varphi$
- $\rho(a, \varphi_1 U \varphi_2) = \rho(a, \varphi_2) \vee [\rho(a, \varphi_1) \wedge \varphi_1 U \varphi_2]$
- $\rho(a, \varphi_1 V \varphi_2) = \ldots\ldots$

Important: translation is linear in states.

Example: $\varphi =$ eventually always p

$\Sigma = \{\{p\}, \emptyset\}$

$S = \{s, t\}$

$s_0 = s$

$F = \{t\}$

$\rho(s, a) = s \vee t$

$\rho(t, \{p\}) = t$

$\rho(t, \emptyset) = false$

NONDET. AUTOMATA EMPTINESS

(Non)emptiness problem:

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

Nondet. Büchi automata:

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

$G(A) = (S, E_\rho)$: graph of A

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

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

Corollary: Nondet. Büchi autom. nonemptiness is in NLOGSPACE.

ALT. AUTOMATA EMPTINESS

Given: alt. Büchi auto. A

Two-step algorithm:

1. Construct non det. Büchi auto. A^b s.t. $L(A^b) = L(A)$
2. Test $L(A^b) \neq \emptyset$

Problem: A^b is exponentially large!

Solution: construct A^b on-the-fly!!!

Corollary: Alt. Büchi auto. nonemptiness is in PSPACE.

Corollary: LTL-SAT $\in$ PSPACE

Real LTL Model checking

Transition system: $\mathcal{M} = (W, W_0, R, \pi)$

- (W, R, π): serial kripke structure $(\forall s \exists t\ R(s,t))$
- $W_0 \subseteq W$: initial worlds

Computation: $w_0, w_1, w_2, \ldots$

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

Trace: $\pi(w_0), \pi(w_1), \pi(w_2) \ldots$

$L(\mathcal{M})$: traces of $\mathcal{M}$

$\mathcal{M} \models \varphi$: $L(\mathcal{M}) \subseteq models(\varphi)$

Language Containment

T.F.A.E

- $M \models \varphi$
- $L(M) \subseteq models(\varphi)$
- $L(M) \cap models(\neg\varphi) = \phi$
- $L(M) \cap L(A_{\neg\varphi}) = \phi$
- $L(M) \cap L(A^b_{\neg\varphi}) = \phi$
- $L(M \times A^b_{\neg\varphi}) = \phi$

<u>corollary:</u> LTL model checking is logspace in the system and linear space in the spec

In practice:

- linear in system
- exponential in spec

Computation Tree Logic

Linear time: traces of M

Branching time: computation tree of M (unfolding of M)

CTL syntax:

- propositional logic
- $\forall \bigcirc \varphi$, $\exists \bigcirc \varphi$, $\forall \varphi \mathcal{U} \psi$, $\exists \varphi \mathcal{U} \psi$, ...

CTL syntax: quantified futures

- $\exists$: for some future
- $\forall$: for all futures

Example:

$\forall \bigcirc \forall$ eventually p: p inevitably holds in the strict future.

CTL MODEL CHECKING

Input:

- φ: CTL formula
- $\mathcal{M} = (W, R, \pi)$: serial kripke struct.
- $w \in W$: world

Problem: $\mathcal{M}, w \models \varphi$?

Algorithm $MC(\mathcal{M}, w, \varphi)$

Case:

$\varphi = p$: return $p \in \pi(w)$

$\varphi = \alpha_1 \wedge \alpha_2$: $(\forall)$ return $MC(\mathcal{M}, w, \alpha_i)$

$\varphi = \forall \bigcirc \psi$: $(\forall)$ return $MC(\mathcal{M}, u, \alpha)$ for $u \in R(w)$

$\varphi = \forall \alpha_1 \, U \, \alpha_2$: $(\exists)$ return $MC(\mathcal{M}, w, \alpha_2)$

or

$(\forall)$ return $MC(\mathcal{M}, w, \alpha_1)$

and return $MC(\mathcal{M}, u, \varphi)$ for $u \in R(w)$

⋮

LF

Complexity Analysis

CTL-MC is Büchi-ASPACE(log m)

- state is (M, u, φ)
- game is infinite
- only ν-formulas can be visited i.o.

Corollary: CTL-MC $\in$ PTIME

In practice: CTL model checking is linear in both system and spec.

CTL satisfiability

Reminder: LTL satisfiability

1. φ (LTL) $\longrightarrow$ A_φ (ABA)
2. A_φ (ABA) $\longrightarrow$ A_φ^b (NBA)
3. $L(A_\varphi^b) \neq \emptyset$?

CTL satisfiability: same, but with trees

1. φ (LTL) $\longrightarrow$ A_φ (tree ABA)
2. A_φ (tree ABA) $\longrightarrow$ A_φ^b (tree NBA)
3. $L(A_\varphi^b) \neq \emptyset$?

Tree ABA: infinite Büchi games on trees

Bottom line: CTL-SAT $\in$ EXPTIME

TOWER OF ABSTRACTIONS

Key idea in Science: T.O.A.

strings

quarks

hadrons

atoms

molecules

proteins

genes

genomes

organisms

populations

T.O.A IN COMPUTER SCIENCE

Analog devices

Digital devices

micro processors

Assembly Languages

High-level languages

Libraries

⋮

Crux: T.O.A is the only way to deal with complexity

Similarly: we need high-level algorithmic building blocks, e.g., DFS, BFS.

This talk: Alternation as a high-level algorithmic idea

ALTERNATION

Two perspectives:

- Game between two player
- control mechanism for Turing machines

Two applications:

- Model checking (implementation of game-theoretic semantics)
- satisfiability checking

Unification: alternating auto.

- 1-letter emptiness
- 2-letter emptiness