

Finite Automata and Applications

Automata and Verification of Reactive Systems

Nelma Moreira, Rogério Reis

Artificial Intelligence and Computer Science Laboratory
Language, Complexity and Cryptography Group

Thematic Seminar – MAPi
25/11/2008

- Automata-theoretic approach to temporal reasoning
- Finite automata accept computations that satisfy a given formula
- Decouples the combinatorial and logical parts
- Asymptotically optimal algorithms

Model Checking

Formal Software Verification

Given program P and specification s determine whether or not the behavior of P meets the specification s .

Reactive systems

Consists of several components which interact with each other and the environment. Normally can be described by a finite state system and are described by temporal properties. Ex: processors, operating systems, communication protocols, automotive electronics, etc.

Model Checking

is a automatic technique for verifying correctness properties of (safety-critical) reactive systems.

Model checking is based on **temporal logic**.

A temporal model \mathcal{M} is set of **states** and **transitions** between them.

A formula ϕ can be **true** in some states and **false** in others.

The static notion of Truth is replaced by a dynamic one.

Given a state s ,

$$\mathcal{M}, s \models \phi$$

A *model checker* is an algorithm to decide this problem.

Time

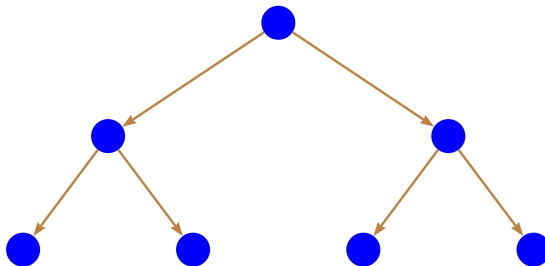
Time points are named **states**.

Linear: Time is a set of paths, and a path is a sequence of states.



One future!

Branching: Time is represented by a tree rooted in the present moment.



Several futures!

Temporal Linear Logic, LTL

Prop, set of propositional variables p, q, r, s, \dots

Syntax

$$\phi ::= \perp \mid \top \mid p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid \\ (\mathbf{X}\phi) \mid (\mathbf{F}\phi) \mid (\mathbf{G}\phi) \mid (\phi\mathbf{U}\phi)$$

Temporal connectives

$\mathbf{X}\phi$ ϕ holds in the *next* state

$\mathbf{F}\phi$ ϕ holds *in some* future state

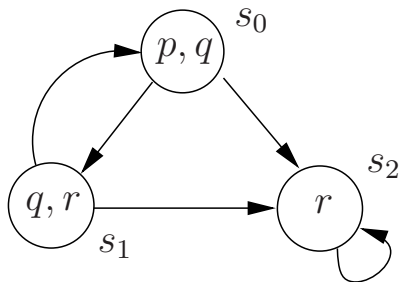
$\mathbf{G}\phi$ ϕ holds *in all* future states (*Global*)

$\phi\mathbf{U}\psi$ ϕ holds *until* ψ (*Until*)

Transition Systems (Kripke Models)

Transition system

$\mathcal{M} = (S, \rightarrow, L)$ where S is a set of states, $\rightarrow \subseteq S \times S$, total binary relation (i.e. $\forall s \in S, \exists s' \in S, s \rightarrow s'$) and $L: S \rightarrow 2^{\text{Prop}}$ labelling function

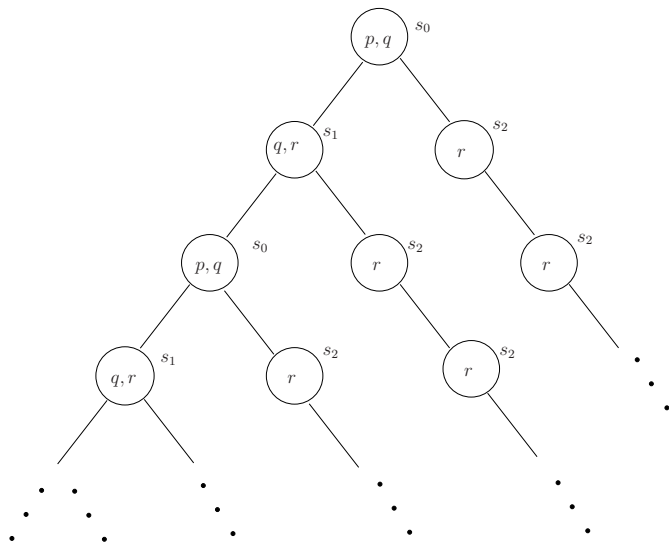


(Computation) path

A **path** π in a model $\mathcal{M} = (S, \rightarrow, L)$ is a infinite sequence of states s_1, s_2, s_3, \dots in S , such that for each $i \geq 1$, $s_i \rightarrow s_{i+1}$. We write $\pi = s_1 \rightarrow s_2 \rightarrow \dots$ and π^i for the suffix starting at s_i .

The set of computation paths from a given state s can be seen as an **infinite computation tree**.

Computation Tree



Satisfaction relation

Let $\mathcal{M} = (S, \rightarrow, L)$ and $\pi = s_1 \rightarrow \dots$

- 1 $\pi \models \top$
- 2 $\pi \not\models \perp$
- 3 $\pi \models p$ iff $p \in L(s_1)$
- 4 $\pi \models \neg\phi$ iff $\pi \not\models \phi$
- 5 $\pi \models \phi \wedge \psi$ iff $\pi \models \phi$ and $\pi \models \psi$
- 6 $\pi \models \phi \vee \psi$ iff $\pi \models \phi$ or $\pi \models \psi$
- 7 $\pi \models \phi \rightarrow \psi$ iff whenever $\pi \models \phi$ then $\pi \models \psi$
- 8 $\pi \models X\phi$ iff $\pi^2 \models \phi$
- 9 $\pi \models G\phi$ iff $\forall i \geq 1, \pi^i \models \phi$
- 10 $\pi \models F\phi$ iff $\exists i \geq 1, \pi^i \models \phi$
- 11 $\pi \models \phi U \psi$ iff $\exists i \geq 1, \pi^i \models \psi$ e $\forall 1 \leq j < i, \pi^j \models \phi$

Satisfaction from a state

Let $\mathcal{M} = (S, \rightarrow, L)$, $s \in S$ and ϕ an LTL formula. We write $\mathcal{M}, s \models \phi$, if, for every execution path π starting at s , we have $\pi \models \phi$.

Consider $\mathcal{M} = (S = \{s_0, s_1, s_2\}, \{s_0 \rightarrow s_1, s_0 \rightarrow s_2, s_1 \rightarrow s_2, s_1 \rightarrow s_0, s_2 \rightarrow s_2\}, L(s_0) = \{p, q\}, L(s_1) = \{q, r\}, L(s_2) = \{r\})$. Determine which relation are true:

- 1 $\mathcal{M}, s_0 \models p \wedge q$
- 2 $\mathcal{M}, s_0 \models Xr$
- 3 $\mathcal{M}, s_0 \models X(q \wedge r)$
- 4 $\mathcal{M}, s_0 \models G\neg(p \wedge r)$
- 5 $\mathcal{M}, s_0 \models GFp$
- 6 $\mathcal{M}, s_0 \models GFp \rightarrow GFr$

Practical patterns of specifications

Propositional variables corresponds to states in a reactive system.

- It is impossible to get to a state where `started` holds, but `ready` does not hold: $G\neg(\text{started} \wedge \neg\text{ready})$
- For any state, if a `request` (of some resource) occurs, then it will eventually be acknowledged: $G(\text{request} \rightarrow F\text{ack})$
- A certain process is enabled infinitely often on every computation path: $GF\text{enabled}$
- Whatever happens, a certain process will eventually be permanently deadlocked: $FG\text{deadlock}$
- If the process is enabled infinitely often, then it runs infinitely often: $GF\text{enabled} \rightarrow GF\text{run}$
- From any state it is possible to get to a `restart` state (i.e., there is a path from all states to a state satisfying `restart`): **It can not be expressed in LTL!**

In LTL it is not possible to express the existence of a path!

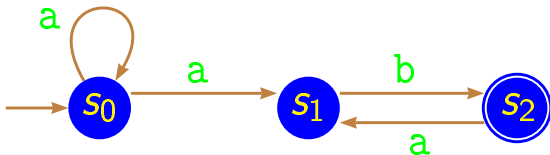
Nondeterministic finite automata

Σ alphabet, $L \subseteq \Sigma^*$ language of finite words

$\mathcal{A} = (\Sigma, S, S^0, \delta, F)$ where $S^0, F \subseteq S$, $\delta : S \times \Sigma \rightarrow 2^S$

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \delta(s_0, w) \cap F \neq \emptyset \wedge s_0 \in S^0\}$$

where $\delta(s, \epsilon) = s$ and $\delta(s, ax) = \delta(\delta(s, a), x)$.



$$L((a^*(ab)^*)$$

Languages accepted by NFAs are closed for union, intersection and complementation; and are equivalent to languages accepted by DFAs.

Infinite words

An infinite word over Σ is a sequence $a_0 a_1 a_2 \dots a_m a_{m+1} \dots$ where $a_i \in \Sigma$.
 Σ^ω set of infinite words over Σ .

$aaabbbbaabbb \dots$ can be represented by the ω -regular expression $(aaabbb)^\omega$

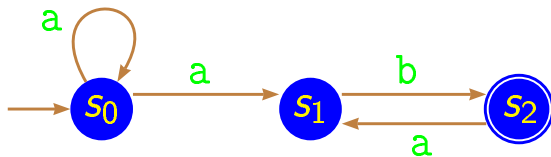
Büchi Automaton (nondeterministic)

$\mathcal{A}_\omega = (\Sigma, S, S^0, \delta, F)$ where $S^0, F \subseteq S$, $\delta : S \times \Sigma \rightarrow 2^S$

Given $w = a_0 a_1 \dots$, a **run** in w (r_w) is a sequence of states s_0, s_1, \dots , with $s_0 \in S^0$ and $s_{i+1} \in \delta(s_i, a_i)$, $i \geq 0$.

$$\text{lim}(r_w) = \{s \mid s = s_i \text{ for infinitely many } i\}$$

$$L(\mathcal{A}_\omega) = \{w \in \Sigma^\omega \mid \exists r_w \text{lim}(r_w) \cap F \neq \emptyset\}$$



$w = aaababab \dots$ is accepted

$$L_\omega((a^*(ab)^\omega))$$

Closure and Characterization

Languages recognizable by Büchi automata are closed under union, intersection and complementation (**VERY HARD TO PROVE!!!**).

But determinism is weaker than nondeterminism. For instance $(0 + 1)^*1^\omega$ can not be recognized by a deterministic Büchi automaton.

Theorem (Characterization)

An ω -language L is recognizable by a Büchi automaton if and only if L is a finite union of sets VW^ω , where $V, W \subset \Sigma^$ are FA recognizables.*

There are other equivalent acceptance conditions: for **Müller automata** we would have determinism, $F \subseteq 2^S$ and $\lim(r_w) \in F$. And complementation is easier!! Other conditions by Rabin, Streett, etc.

$L_\omega(\mathcal{A}_\omega) \neq \emptyset$? and $L_\omega(\mathcal{A}_\omega) \neq \Sigma^\omega$?

Theorem

The nonemptiness problem for a \mathcal{A}_ω Büchi automaton is decidable in linear time (NLOGSPACE-complete).

Proof.

Given $\mathcal{A}_\omega = (\Sigma, S, S^0, \delta, F)$, $L_\omega(\mathcal{A}_\omega)$ is nonempty iff exists a $s_0 \in S_0$ and $t \in F$ such that s_0 is connected to t and t is connected to itself. \square

Theorem

The nonuniversality problem for a \mathcal{A}_ω Büchi automaton is decidable in exponential time (PSPACE-complete).

Proof.

To obtain a complementary automaton $\overline{\mathcal{A}_\omega}$ is exponential and $L_\omega(\mathcal{A}_\omega) \neq \Sigma^\omega \leftrightarrow L_\omega(\overline{\mathcal{A}_\omega}) \neq \emptyset$. \square

Computation paths can be seen as infinite words over 2^{Prop} . To each LTL formula ϕ we can associate a Büchi automaton \mathcal{A}_ϕ , such that the language $L_\omega(\mathcal{A}_\phi)$ is the set of computation paths that satisfy ϕ .

To simplify the exposition and because they are exponentially more succinct than NFAs we are going to introduce **alternating automata**.

In a NFA the (existential) choice between two or more states can be seen as a disjunction.

Alternation generalize this concept to other boolean formulas (and an universal choice).

Alternating Finite Automata (AFA)

Given a set X , let $\mathcal{B}^+(X)$ be the set of Boolean formulas with \wedge and \vee (and \top and \perp)

$Y \subseteq X$ satisfies a formula $\theta \in \mathcal{B}^+(X)$ if the truth assignment that assigns 1 to the members of Y and assigns 0 otherwise, satisfies θ .

$\{s_1, s_3\}$ satisfies $(s_1 \vee s_2) \wedge (s_3 \vee s_4)$

In NFA we can represent δ using $\mathcal{B}^+(S)$. For example

$\delta(s_0, a) = \{s_1, s_0\} = s_1 \vee s_0$.

In AFA, $\delta(a, s)$ can be any formula of $\mathcal{B}^+(S)$.

$\delta(s_0, a) = (s_1 \wedge s_2) \vee (s_3 \wedge s_4)$ means that the automaton accepts aw from s_0 , if it accepts w from s_1 and from s_2 , or accepts w from s_3 and from s_4 .

In this case, we have an existential and a universal choice.

Alternating Finite Automata (AFA)

$A = (\Sigma, S, s^0, \delta, F)$ where $s^0 \in S$, is the initial state, $F \subseteq S$ is the set of final states, and $\delta : S \times \Sigma \rightarrow \mathcal{B}^+(S)$

A **run** is now a tree r (S -labelled) with root, ϵ . The *level* of a node x ($|x|$) is its distance from the root. $r(x)$ is the label of a node x .

A *branch* $\beta = x_0x_1\dots$ is a maximal sequence of nodes such that $x_0 = \epsilon$ and x_i is the parent of x_{i+1} , $i \geq 0$. And $r(\beta) = r(x_0)r(x_1)\dots$ is the sequence of labels along the branch.

Run

A **run** on a word $w = a_0 \dots a_n$ is a S -labeled finite tree, r_w such that $r(\epsilon) = s^0$ and:

if $|x| = i < n$, $r(x) = s$ and $\delta(s, a_i) = \theta$ then x has k children x_1, \dots, x_k for some $k \leq |S|$, and $\{r(x_1), \dots, r(x_k)\}$ satisfies θ .

If $\delta(r(x), a_i) = \top$ then x has no children (and \perp never occurs).

Alternating Finite Automata (AFA)

Accepting run

A run is **accepting** if all nodes at depth n are labeled by states in F .
Either a branch hits \top or hits a final state.

Language recognized by an AFA

$$L(A) = \{w \in \Sigma^* \mid \exists \text{ an accepting } r_w\}$$

Theorem (Equivalence AFA and NFA)

A language is recognizable by a NFA iff it is recognizable by a AFA (but the AFA are exponentially more succinct).

Complementation

If $\theta \in \mathcal{B}^+(S)$ its dual $\bar{\theta}$ is obtained from θ by switching \top and \perp , and \vee and \wedge . Given AFA $A = (\Sigma, S, s^0, \delta, F)$, $\bar{A} = (\Sigma, S, s^0, \bar{\delta}, S - F)$ and $\bar{\delta}(s, a) = \overline{\delta(s, a)}$. Then $L(\bar{A}) = \Sigma^* \setminus L(A)$.

Alternating Büchi Automata

Let $A = (\Sigma, S, s^0, \delta, F)$ be alternating automaton on infinite words. We call it an alternating Büchi automaton.

Run

A **run** of A on an infinite word $w = a_0 a_1 \dots$ is a S -labeled tree r such that $r(\epsilon) = s^0$ and:

if $|x| = i$, $r(x) = s$ and $\delta(s, a_i) = \theta$, then x has k children x_1, \dots, x_k with $k \leq |S|$ and $\{r(x_1), \dots, r(x_k)\}$ satisfies θ .

A run is **accepting** if every infinite branch in r includes infinitely many labels in F . If $\delta(s, a_i) = \top$, x has no children. A finite run is accepting.

$$L_\omega(A) = \{w \in \Sigma^\omega \mid \text{there exists an accepting run } r \text{ on } w\}$$

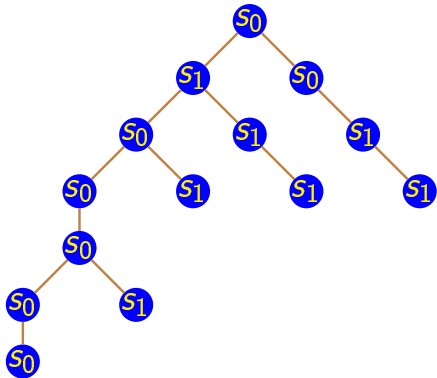
Example

$(\{a, b, c, d\}, \{s_0, s_1\}, \delta, s_0, \{s_1\})$

δ	a	b	c	d
s_0	\top	\top	$s_0 \wedge s_1$	s_0
s_1	\perp	\top	s_1	\top

Consider a run on *ccdcdad* ...

c
c
c
d
c
d
a



Alternating Büchi Automata

Theorem

A language is recognizable by an alternating Büchi automaton if and only if it is recognizable by a nondeterministic Büchi automaton (but alternating automata are exponentially more succinct).

Complementation

Complementation of alternating Büchi automata is not easy: going infinitely often through accepting states is not the same as going infinitely often through non-accepting states.

Theorem

The nonemptiness problem for alternating Büchi automata is decidable in exponential time (PSPACE-complete).

Let $\mathcal{M} = (S, \rightarrow, L)$ be a model and $\pi = s_1 \rightarrow \dots$ a computation path. Given L , a path can be seen as an infinite word over 2^{Prop} .

The set of paths satisfying a given formula are exactly those accepted by some automaton on infinite words (Sherman, Pnueli and Harel, 1984).

Theorem

Given an LTL formula ϕ one can build an alternating Büchi automaton $A_\phi = (2^{\text{Prop}}, S, \phi, \delta, F)$ where $|S| = \mathcal{O}(|\phi|)$ such that $L_\omega(A_\phi)$ is exactly the set of paths satisfying the formula ϕ .

Proof.

Let S be the set of all subformulas of ϕ and their negation. Let F be the set of all formulas in S of the form $\neg(\psi U \phi)$. Let the dual of a formula to be extended to negation $\overline{\neg\phi} = \phi$ and $\overline{\phi} = \neg\phi$. The transition function δ is:

- $\delta(p, a) = \top$ if $p \in a$
- $\delta(p, a) = \perp$ if $p \notin a$
- $\delta(\phi \wedge \psi, a) = \delta(\phi, a) \wedge \delta(\psi, a)$
- $\delta(\neg\phi, a) = \overline{\delta(\phi, a)}$
- $\delta(X\phi, a) = \phi$
- $\delta(\phi U \psi, a) = \delta(\psi, a) \vee (\delta(\phi, a) \wedge \phi U \psi)$



LTL Satisfiability and Alternating Büchi Automata

Consider r a run of A_ϕ . Each infinite branch is labeled from some point on by $\phi U \psi$ or $\neg(\phi U \psi)$. Since

$$\delta(\neg(\phi U \psi), a) = \overline{\delta(\psi, a)} \wedge (\overline{\delta(\phi, a)} \vee \neg(\phi U \psi))$$

an infinite branch labeled from some point by $\neg(\phi U \psi)$ ensures that $\phi U \psi$ indeed fails at that point, since ψ fails. That is why those states are final. On the other hand, an infinite branch labeled by $\phi U \psi$ does not ensure that $\phi U \psi$ holds. The proof follows by induction on the structure of the formula ϕ .

Corollary

Given an LTL formula ϕ , one can build a Büchi automaton A_ϕ such that $L(A_\phi)$ is exactly the set of paths satisfying the formula ϕ .

Example

Let $\phi = (X\neg p)Uq$.

$$A_\phi = (2^{\{p,q\}}, \{\phi, \neg\phi, X\neg p, \neg X\neg p, \neg p, p, \neg q, q\}, \phi, \delta, \{\neg\phi\})$$

s	$\{p, q\}$	$\{p\}$	$\{q\}$	\emptyset
ϕ	\top	$\neg p \wedge \phi$	\top	$\neg p \wedge \phi$
$\neg\phi$	\perp	$p \vee \neg\phi$	\perp	$p \vee \neg\phi$
$X\neg p$	$\neg p$	$\neg p$	$\neg p$	$\neg p$
$\neg X\neg p$	p	p	p	p
$\neg p$	\perp	\perp	\top	\top
p	\top	\top	\perp	\perp
q	\top	\perp	\top	\perp
$\neg q$	\perp	\top	\perp	\top

In the state ϕ if q does not hold then $\neg p$ and ϕ must hold in the next state. As $\phi \notin F$, eventually q must be satisfiable.

Given $M = (S, \rightarrow, L)$ and Prop

Any $\pi = s_0 s_1 \dots$ can be seen as sequence of subsets of Prop

We associate with M a Büchi automaton

$$A_M = (2^{\text{Prop}}, S, \{s_0\}, \delta, S),$$

such that $s' \in \delta(s, a)$ iff $s \rightarrow s'$ and $a = L(s)$.

Since the set of final states is S , every π is an accepting run and $L_\omega(A_M)$ is the set of paths of M .

$M, s_0 \models \phi$ reduces to know if

$$L_\omega(A_M) \subseteq L_\omega(A_\phi)$$

Or equivalently,

$$L_\omega(A_M) \cap L_\omega(\overline{A_\phi}) = \emptyset$$

As $L_\omega(\overline{A_\phi}) = L_\omega(A_{\neg\phi})$, complementation is not a problem and $A_{\neg\phi}$ has $2^{\mathcal{O}(|\phi|)}$ states. The intersection automaton has $|S|.2^{\mathcal{O}(|\phi|)}$ states.

Theorem

Checking whether a model M satisfies an LTL formula ϕ can be done in time $\mathcal{O}(|S|.2^{\mathcal{O}(|\phi|)})$

Usually the specification is rather short...

Branching-time Logic - CTL

The evolution of the transition system corresponds to an infinite computation tree.

Computation Tree Logic (CTL) allows explicit and existential quantification over paths.

Prop, set of propositional variables p, q, r, s, \dots

Syntax

$$\phi ::= \perp \mid \top \mid p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \Rightarrow \phi) \mid (AX\phi) \mid \\ (EX\phi) \mid (AF\phi) \mid (EF\phi) \mid (AG\phi) \mid (EG\phi) \mid A[\phi U \phi] \mid E[\phi U \phi]$$

Temporal connectives

A means *along All paths*

E means *along at least one path*

F, G, X and U as in LTL

A and E can only appear with one of the other temporal connectives.

Examples

$AG(p \Rightarrow EGr)$

$EFE[rUq]$

$E[A[rUp]Uq]$

$A[AX\neg pUE[EX(p \wedge q)U\neg p]]$

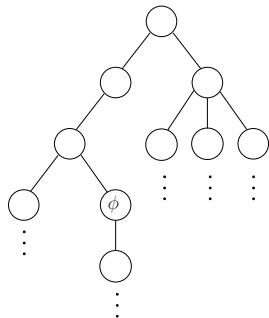
Satisfiability

Given $\mathcal{M} = (S, \rightarrow, L)$, a state $s \in S$ and an CTL formula ϕ we define $\mathcal{M}, s \models \phi$:

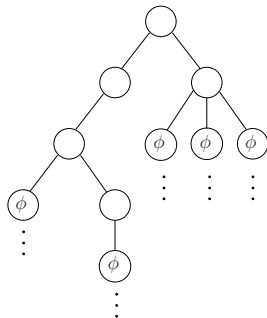
- 1 $\mathcal{M}, s \models \top$ and $\mathcal{M}, s \not\models \perp$
- 2 $\mathcal{M}, s \models p$ iff $p \in L(s)$
- 3 $\mathcal{M}, s \models \neg\phi$ iff $\mathcal{M}, s \not\models \phi$
- 4 $\mathcal{M}, s \models \phi \wedge \psi$ iff $\mathcal{M}, s \models \phi$ and $\mathcal{M}, s \models \psi$
- 5 $\mathcal{M}, s \models \phi \vee \psi$ iff $\mathcal{M}, s \models \phi$ or $\mathcal{M}, s \models \psi$
- 6 $\mathcal{M}, s \models \phi \Rightarrow \psi$ iff if $\mathcal{M}, s \models \phi$ then $\mathcal{M}, s \models \psi$
- 7 $\mathcal{M}, s \models AX\phi$ iff for all s_1 such that $s \rightarrow s_1$, $\mathcal{M}, s_1 \models \phi$
- 8 $\mathcal{M}, s \models EX\phi$ iff exists s_1 such that $s \rightarrow s_1$ e $\mathcal{M}, s_1 \models \phi$

Satisfiability

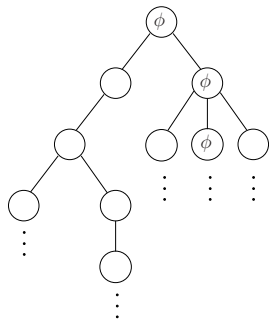
- 9 $\mathcal{M}, s \models \text{AG}\phi$ iff for all paths $\pi = s_1 \rightarrow s_2 \cdots$ with $s_1 = s$, and all $s_j \in \pi$ $\mathcal{M}, s_j \models \phi$
- 10 $\mathcal{M}, s \models \text{EG}\phi$ iff if exists a path $\pi = s_1 \rightarrow s_2 \cdots$ with $s_1 = s$ such that for all $s_j \in \pi$ $\mathcal{M}, s_j \models \phi$
- 11 $\mathcal{M}, s \models \text{AF}\phi$ iff for all paths $\pi = s_1 \rightarrow s_2 \cdots$ with $s_1 = s$, and for some $s_j \in \pi$, $\mathcal{M}, s_j \models \phi$
- 12 $\mathcal{M}, s \models \text{EF}\phi$ iff exists a path $\pi = s_1 \rightarrow s_2 \cdots$ with $s_1 = s$ and for some $s_j \in \pi$, $\mathcal{M}, s_j \models \phi$
- 13 $\mathcal{M}, s \models \text{A}[\phi_1 \text{U} \phi_2]$ iff for all paths $\pi = s_1 \rightarrow s_2 \cdots$ with $s_1 = s$, we have that $\phi_1 \text{U} \phi_2$ is satisfied, i.e. there is $s_j \in \pi$ $\mathcal{M}, s_j \models \phi_2$, and for each $j < i$ $\mathcal{M}, s_j \models \phi_1$
- 14 $\mathcal{M}, s \models \text{E}[\phi_1 \text{U} \phi_2]$ iff there is a path $\pi = s_1 \rightarrow s_2 \cdots$ with $s_1 = s$, and that path satisfies $\phi_1 \text{U} \phi_2$, i.e. there is $s_j \in \pi$ $\mathcal{M}, s_j \models \phi_2$, and for each $j < i$ $\mathcal{M}, s_j \models \phi_1$.



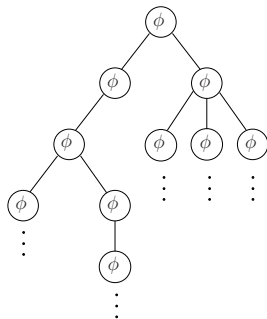
$EF\phi$



$AF\phi$



$EG\phi$



$AG\phi$

CTL is not strictly more expressive than LTL.

$$Fp \Rightarrow Fq$$

cannot be expressed in CTL as it is not the same as:

$$AFp \Rightarrow AFq \text{ or } AG(p \Rightarrow AFq)$$

CTL*

CTL where it is not mandatory that a LTL connective $\{X, G, F, U\}$ is preceded by a A or E.

Examples:

$$A[(pUr) \vee (qUr)]$$

$$E(GF\phi)$$

Syntax

① *state formulas*

$$\phi ::= \top \mid p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid A\alpha \mid E\alpha$$

② *path formulas*

$$\alpha ::= \phi \mid (\neg\alpha) \mid (\alpha \wedge \alpha) \mid \alpha U \alpha \mid (G\alpha) \mid (F\alpha) \mid (X\alpha)$$

CTL* is strictly more expressive than both LTL and CTL, but computationally much more inefficient.

A tree τ over \mathbb{N} is a subset of \mathbb{N}^* that is prefix-closed. For each node x , $\text{arity}(x)$ is the number of children of x .

Let $D \subset \mathbb{N}$. A tree τ is a D -tree if τ is a tree over \mathbb{N} and $\text{arity}(x) \in D, \forall x \in \tau$.

If $D = \{1, 2\}$ a D -tree is a binary tree.

A tree is called **leafless** if every node has at least one child.

A Σ -labeled tree is a pair (τ, \mathcal{T}) , where $\mathcal{T} : \text{nodes}(\tau) \rightarrow \Sigma$

We are going to consider labeled leafless D -trees.

A nondeterministic Büchi tree automata

$$A = (\Sigma, D, S, S^0, \delta, F)$$

- Σ finite alphabet
- $D \subset \mathbb{N}$ finite set of arities
- S finite set of states
- $S^0 \subseteq S$ set of initial states
- $F \subseteq S$, set of final states
- $\delta : S \times \Sigma \times D \rightarrow 2^{S^*}$ transition function, where $\delta(s, a, k) \subseteq S^k$

When the automaton is in state s and it is reading a k -ary node x of (τ, \mathcal{T}) , it nondeterministically chooses a k -tuple (s_1, \dots, s_k) in $\delta(s, \mathcal{T}(x))$, makes k copies of itself, and then moves to the node $x \cdot i$ in the state s_i for $i = 1, \dots, k$.

Run

A **run** $r : \tau \rightarrow S$ of A on a Σ -labeled D -tree τ is an S -labeled D -tree such that $r(\epsilon) \in S^0$ and for each node x such that $\text{arity}(x) = k$, we have $(r(x \cdot 1), \dots, r(x \cdot k)) \in \delta(r(x), \mathcal{T}(x), k)$.

Accepting Run

The run is **accepting** if $\lim(r(\beta)) \cap F \neq \emptyset$ for every branch $\beta = x_0, x_1, \dots$ of τ .

$$T_\omega(A) = \{\tau \mid \tau \text{ is accepted by } A\}$$

A Büchi automata on infinite words is essentially a Büchi tree automata on $\{1\}$ -trees.

Theorem

The nonemptiness problem for nondeterministic Büchi tree automata is decidable in quadratic time.

An alternating Büchi tree automaton

$$A = (\Sigma, D, S, s^0, \delta, F)$$

- Σ finite alphabet
- $D \subset \mathbb{N}$ finite set of arities
- S finite set of states
- $s^0 \in S$ set of initial states
- $F \subseteq S$, set of final states
- $\delta : S \times \Sigma \times D \rightarrow \mathcal{B}^+(\mathbb{N} \times S)$ partial transition function, where $\delta(s, a, k) \in \mathcal{B}^+(\{1, \dots, k\} \times S)$ for each $s \in S$, $a \in \Sigma$, and $k \in D$ if defined.

Alternating Büchi Tree Automata

A run r of A on a Σ -labeled leafless D -tree (τ, \mathcal{T}) is a $\tau \times S$ -labeled tree. Each node of r corresponds to a node of τ . A node in r , labeled by (x, s) , describes a copy of the automaton that reads the node x of τ in the state s . The labels of a node and its children have to satisfy the transition function.

Run

A **run** r is a $\tau \times S$ -labeled tree (τ_r, \mathcal{T}_r) where

- 1 $\mathcal{T}_r(\epsilon) = (\epsilon, s^0)$
- 2 Let $y \in \tau_r$, $\mathcal{T}_r(y) = (x, s)$, $\text{arity}(x) = k$, and $\delta(s, \mathcal{T}(x), k) = \theta$. Then there is a set

$$Q = \{(c_1, s_1), (c_2, s_2), \dots, (c_n, s_n)\} \subseteq \{1, \dots, k\} \times S$$

such that

- Q satisfies θ , and
- for all $1 \leq i \leq n$, we have $y \cdot i \in \tau_r$ and $\mathcal{T}_r(y \cdot i) = (x \cdot c_i, s_i)$.

Alternating Büchi Tree Automata

Theorem

Let A be an alternating Büchi tree automaton with n states. Then there is a nondeterministic Büchi tree automaton A_n with 3^n states such that $T_\omega(A_n) = T_\omega(A)$.

Theorem

The nonemptiness problem for alternating Büchi tree automata is decidable in exponential time.

This result is not very good for CTL- model checking... but

Theorem

The 1-letter nonemptiness problem for uniform alternating Büchi tree automata is decidable in quadratic time (uniform means $|D| = 1$).

Weak Alternating Büchi Tree Automata

Weak alternating Büchi tree automata (WAA) has a special accepting condition that allows the partition of S in sets that are included in F or disjoint from F . Then:

Theorem

The 1-letter nonemptiness problem for uniform weak alternating tree automata is decidable in linear time.

Or even better

Theorem

The 1-letter nonemptiness problem for uniform limited-alternation WAA of size n and depth m can be solved in space $\mathcal{O}(m \log^2 n)$.

A model $\mathcal{M} = (S, \rightarrow, L)$ with a state $s \in S$ is a *tree model* if (S, \rightarrow) is a tree and s is its root. Then \mathcal{M} can be seen as a leafless 2^{Prop} -labeled tree. \mathcal{M} is a D -tree model, for $D \subseteq \mathbb{N}$, if (S, \rightarrow) is a D -tree.

Theorem

Given a CTL formula ϕ and a finite set $D \subseteq \mathbb{N}$, one can build a limited-alternation WAA $A_\phi = (\Sigma, D, S, s^0, \delta, F)$, where $\Sigma = 2^{\text{Prop}}$ and $|S|$ is in $\mathcal{O}(|\phi|)$, such that $T_\omega(A_\phi)$ is exactly the set of D -tree models satisfying ϕ .

For branching temporal logic, each model corresponds to a single *computation tree*.

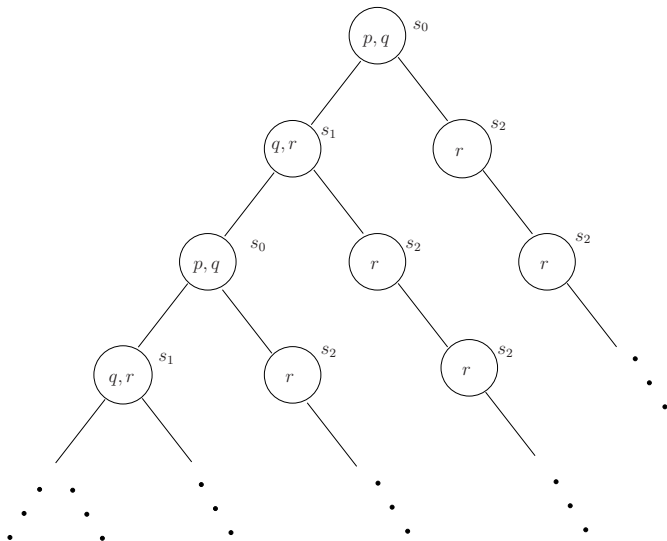
Model checking is reduced to checking acceptance of this computation tree by the automaton describing the formula.

A model $\mathcal{M} = (S, \rightarrow, L)$ with a state $s_0 \in S$, can be seen as a S -tree (τ_M, \mathcal{T}_M) that corresponds to the unwinding of \mathcal{M} from s_0 . For $s \in S$, the *arity*(s) is the number of \rightarrow -successors of s and let $\text{succ}(s) = (s_1, \dots, s_{\text{arity}(s)})$. τ_M and \mathcal{T}_M are defined by:

- 1 $\epsilon \in \tau_M$ and $\mathcal{T}_M(\epsilon) = s_0$
- 2 For $y \in \tau_M$ with $\text{succ}(\mathcal{T}_M(y)) = (s_1, \dots, s_k)$ and for all $1 \leq i \leq k$, we have $y \cdot i \in \tau_M$ and $\mathcal{T}_M(y \cdot i) = s_i$.

Let D be the set of arities of states of \mathcal{M} . τ_M is a D -tree.

A Computation Tree (again)



Let $(\tau_M, L \cdot \mathcal{T}_M)$ be the 2^{Prop} -labeled D -tree defined by

$L \cdot \mathcal{T}_M(y) = L(\mathcal{T}_M(y))$ for $y \in \tau_M$.

Let ϕ be a CTL formula.

Suppose that $A_{D,\phi} = (2^{\text{Prop}}, D, S^\phi, \phi, \delta, F)$ is an alternating automaton that accepts exactly all D -tree models that satisfy ϕ

$(\tau_M, L \cdot \mathcal{T}_M)$ is accepted by $A_{D,\phi}$ iff $M, s_0 \models \phi$

But It is possible to get an alternating Büchi tree automaton $A_{M,\phi}$ on a 1-letter alphabet that is nonempty iff $(\tau_M, L \cdot \mathcal{T}_M)$ is accepted by $A_{D,\phi}$

Theorem

- $A_{M,\phi} = (\{a\}, D, S \times S^\phi, \delta', (s_0, \phi), S \times F)$ is a limited-alternation WAA, which is a product of \mathcal{M} and $(A_{D,\phi})$, i.e.,
 $|A_{M,\phi}| = \mathcal{O}(|M| \cdot |A_{D,\phi}|)$
- $T_\omega(A_{M,\phi})$ is nonempty if and only if $M, s_0 \models \phi$.

Theorem

Checking whether a model M satisfies a CTL formula ϕ can be done in time $\mathcal{O}(|M| \cdot |\phi|)$ or in space $\mathcal{O}(|\phi| \log^2 |M|)$.

	Model Checking	Satisfiability
LTL	PSPACE-complete	PSPACE-complete
CTL	P-complete	DEXPTIME-complete
CTL*	P-complete	D2-EXPTIME-complete
μ -calculus	$NP \cap co-NP$	DEXPTIME-complete