Translating LTL to Probabilistic Automata

Dileep Kini   Mahesh Viswanathan

University of Illinois, Urbana-Champaign

Mysore Park Workshop, February 2016

Linear Temporal Logic (LTL)

[Pnueli 1977]

Syntax

$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid$	Boolean operators
$X\varphi \mid \varphi \, U \, \varphi \mid \varphi \, R \, \varphi$	Temporal operators

Linear Temporal Logic (LTL)

[Pnueli 1977]

Syntax

$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid$	Boolean operators
$X\varphi \mid \varphi\, U\, \varphi \mid \varphi\, R\, \varphi$	Temporal operators

Semantics

For $\alpha \in (2^P)^\omega$, $\alpha[i:\infty]$ is the suffix starting at position i

Linear Temporal Logic (LTL)

[Pnueli 1977]

Syntax

$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid$	Boolean operators
$X\varphi \mid \varphi\, U\, \varphi \mid \varphi\, R\, \varphi$	Temporal operators

Semantics

For $\alpha \in (2^P)^\omega$, $\alpha[i:\infty]$ is the suffix starting at position i and $\alpha[0]$ is the first element of sequence.

Linear Temporal Logic (LTL)

[Pnueli 1977]

Syntax

$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid$	Boolean operators
$X\varphi \mid \varphi\, U\, \varphi \mid \varphi\, R\, \varphi$	Temporal operators

Semantics

For $\alpha \in (2^P)^\omega$, $\alpha[i:\infty]$ is the suffix starting at position i and $\alpha[0]$ is the first element of sequence.

- $\alpha \models p$ iff $p \in \alpha[0]$

Linear Temporal Logic (LTL)
[Pnueli 1977]

Syntax

$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid$	Boolean operators
$X\varphi \mid \varphi\, U\, \varphi \mid \varphi\, R\, \varphi$	Temporal operators

Semantics

For $\alpha \in (2^P)^\omega$, $\alpha[i : \infty]$ is the suffix starting at position i and $\alpha[0]$ is the first element of sequence.

- $\alpha \models p$ iff $p \in \alpha[0]$
- $\alpha \models X\varphi$ iff $\alpha[1 : \infty] \models \varphi$

Linear Temporal Logic (LTL)

[Pnueli 1977]

Syntax

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \quad \text{Boolean operators}$$
$$X\varphi \mid \varphi \, U \, \varphi \mid \varphi \, R \, \varphi \quad \text{Temporal operators}$$

Semantics

For $\alpha \in (2^P)^\omega$, $\alpha[i:\infty]$ is the suffix starting at position i and $\alpha[0]$ is the first element of sequence.

- $\alpha \models p$ iff $p \in \alpha[0]$
- $\alpha \models X\varphi$ iff $\alpha[1:\infty] \models \varphi$
- $\alpha \models \varphi \, U \, \psi$ iff there is j such that $\alpha[j:\infty] \models \psi$ and for all $i < j$ $\alpha[i:\infty] \models \varphi$

Linear Temporal Logic (LTL)

[Pnueli 1977]

Syntax

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \qquad \text{Boolean operators}$$
$$X\varphi \mid \varphi\, U\, \varphi \mid \varphi\, R\, \varphi \qquad \text{Temporal operators}$$

Semantics

For $\alpha \in (2^P)^\omega$, $\alpha[i:\infty]$ is the suffix starting at position i and $\alpha[0]$ is the first element of sequence.

- $\alpha \models p$ iff $p \in \alpha[0]$
- $\alpha \models X\varphi$ iff $\alpha[1:\infty] \models \varphi$
- $\alpha \models \varphi\, U\, \psi$ iff there is j such that $\alpha[j:\infty] \models \psi$ and for all $i < j$ $\alpha[i:\infty] \models \varphi$
- $\alpha \models \varphi\, R\, \psi$ iff either for every i, $\alpha[i:\infty] \models \psi$ or

Linear Temporal Logic (LTL)

[Pnueli 1977]

Syntax

$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid$	Boolean operators
$X\varphi \mid \varphi \, U \, \varphi \mid \varphi \, R \, \varphi$	Temporal operators

Semantics

For $\alpha \in (2^P)^\omega$, $\alpha[i:\infty]$ is the suffix starting at position i and $\alpha[0]$ is the first element of sequence.

- $\alpha \models p$ iff $p \in \alpha[0]$
- $\alpha \models X\varphi$ iff $\alpha[1:\infty] \models \varphi$
- $\alpha \models \varphi \, U \, \psi$ iff there is j such that $\alpha[j:\infty] \models \psi$ and for all $i < j$ $\alpha[i:\infty] \models \varphi$
- $\alpha \models \varphi \, R \, \psi$ iff either for every i, $\alpha[i:\infty] \models \psi$ or there is j such that $\alpha[j:\infty] \models \varphi$ and for all $i < j$ $\alpha[i:\infty] \models \psi$

Translating LTL to Automata

Theorem (Sistla-Vardi-Wolper 1985)

For every LTL formula φ, there is a nondeterministic Büchi automaton $\mathcal{M}$ of size $O(2^{|\varphi|})$ such that $\mathcal{L}(\mathcal{M}) = [\![\varphi]\!]$

Translating LTL to Automata

Theorem (Sistla-Vardi-Wolper 1985)

For every LTL formula φ, there is a nondeterministic Büchi automaton $\mathcal{M}$ of size $O(2^{|\varphi|})$ such that $\mathcal{L}(\mathcal{M}) = [\![\varphi]\!]$

Applications

Gave first non-elementary decision procedure for

- Satisfiability and validity of LTL

Translating LTL to Automata

Theorem (Sistla-Vardi-Wolper 1985)

For every LTL formula φ, there is a nondeterministic Büchi automaton $\mathcal{M}$ of size $O(2^{|\varphi|})$ such that $\mathcal{L}(\mathcal{M}) = [\![\varphi]\!]$

Applications

Gave first non-elementary decision procedure for

- Satisfiability and validity of LTL
- Verifying system designs

Why translate LTL to probabilistic automata?

Understanding the power of randomization

Understanding the power of randomization

Central Question: What computational power do nondeterminism and randomization provide?

Understanding the power of randomization

Central Question: What computational power do nondeterminism and randomization provide?

- Nondeterminism, in the context of finite automata, reasonably well understood
  - Nondeterminism (in most cases) provides no additional computational power,

Understanding the power of randomization

Central Question: What computational power do nondeterminism and randomization provide?

- Nondeterminism, in the context of finite automata, reasonably well understood
  - Nondeterminism (in most cases) provides no additional computational power, but nondeterministic machines can have exponentially fewer states.

Understanding the power of randomization

Central Question: What computational power do nondeterminism and randomization provide?

- Nondeterminism, in the context of finite automata, reasonably well understood
  - Nondeterminism (in most cases) provides no additional computational power, but nondeterministic machines can have exponentially fewer states.
- What about probabilistic automata?

Understanding the power of randomization

Central Question: What computational power do nondeterminism and randomization provide?

- Nondeterminism, in the context of finite automata, reasonably well understood
  - Nondeterminism (in most cases) provides no additional computational power, but nondeterministic machines can have exponentially fewer states.
- What about probabilistic automata?
  - Probabilistic finite state machines can solve problems that cannot be solved on deterministic/nondeterministic automata

Understanding the power of randomization

Central Question: What computational power do nondeterminism and randomization provide?

- Nondeterminism, in the context of finite automata, reasonably well understood
  - Nondeterminism (in most cases) provides no additional computational power, but nondeterministic machines can have exponentially fewer states.
- What about probabilistic automata?
  - Probabilistic finite state machines can solve problems that cannot be solved on deterministic/nondeterministic automata
  - What about from the perspective of memory/states?

Applications

Translation from LTL to nondeterministic automata not good for certain applications

- Monitoring
- Solving games
- MDP model checking

Applications

Translation from LTL to nondeterministic automata not good for certain applications

- Monitoring
- Solving games
- MDP model checking

For such applications one usually translates the logic to deterministic automata.

Applications

Translation from LTL to nondeterministic automata not good for certain applications

- Monitoring
- Solving games
- MDP model checking

For such applications one usually translates the logic to deterministic automata.
Can probabilistic automata help?

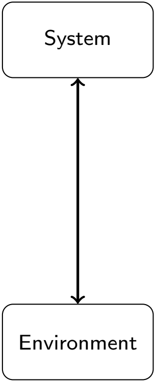
Applications

Translation from LTL to nondeterministic automata not good for certain applications

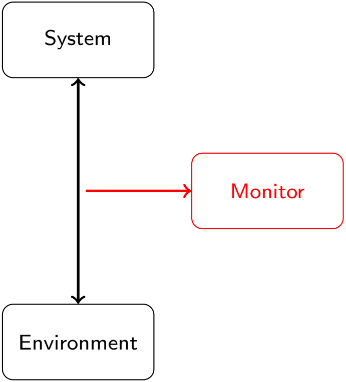
- Monitoring
- Solving games
- MDP model checking

For such applications one usually translates the logic to deterministic automata.
Can probabilistic automata help?

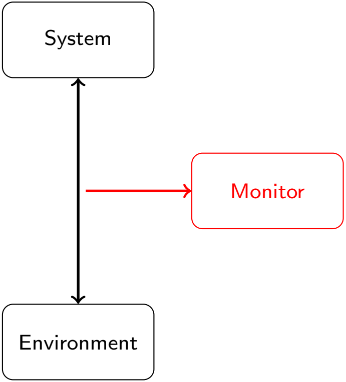
Dynamic Analysis of Systems



Dynamic Analysis of Systems

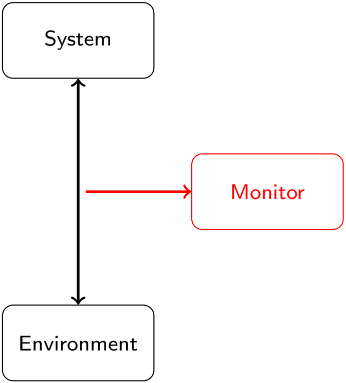


Dynamic Analysis of Systems



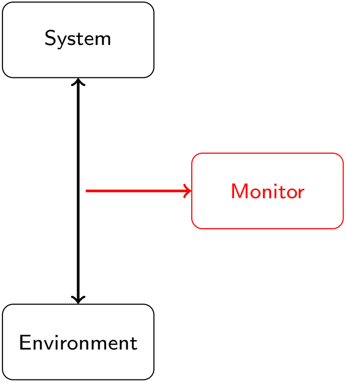
- Monitor passively observes system behavior

Dynamic Analysis of Systems



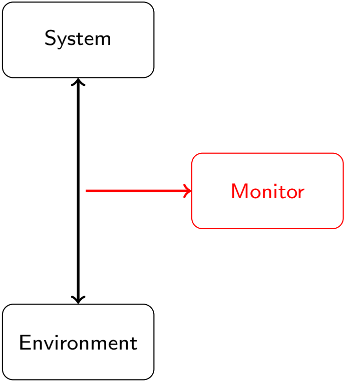
- Monitor passively observes system behavior which is an unbounded stream of events

Dynamic Analysis of Systems



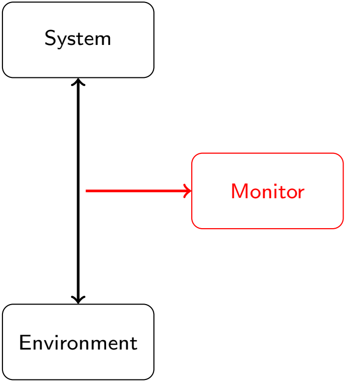
- Monitor passively observes system behavior which is an unbounded stream of events
- Alarm raised when a problem is discovered

Dynamic Analysis of Systems



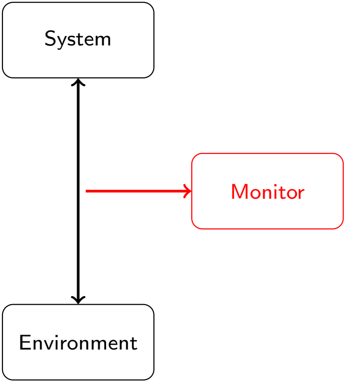
- Monitor passively observes system behavior which is an unbounded stream of events
- Alarm raised when a problem is discovered; correctness indicated implicitly by the absence of alarms

Dynamic Analysis of Systems

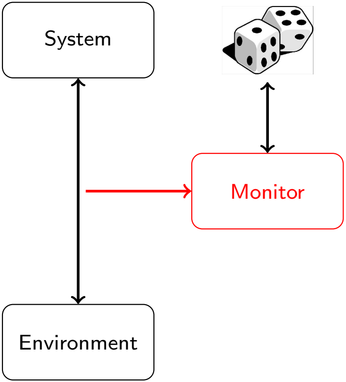


- Monitor passively observes system behavior which is an unbounded stream of events
- Alarm raised when a problem is discovered; correctness indicated implicitly by the absence of alarms
- Application: Discovery of errors and intrusions in deployed systems

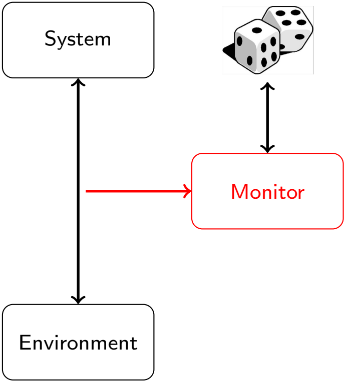
Monitoring



Randomized Monitoring

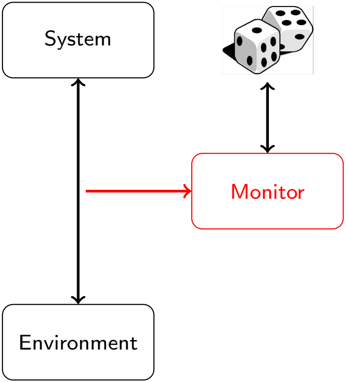


Randomized Monitoring



- The monitor has access to private source of randomness

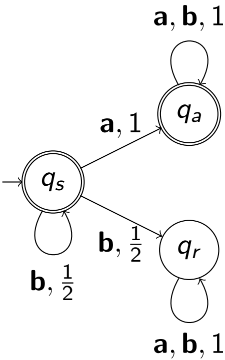
Randomized Monitoring



- The monitor has access to private source of randomness
- The system itself is not probabilistic

Finite State Probabilistic Monitors (FPM)

[Chadha-Sistla-V. 2008]



Definition

A FPM over alphabet Σ is $\mathcal{M} = (Q, q_s, q_r, \delta)$, where Q is a finite set of states, $q_s \in Q$ is the initial state, $q_r \in Q$ is the absorbing reject state, and $\delta : Q \times \Sigma \times Q \rightarrow [0, 1]$ is such that for any $q \in Q$ and $a \in \Sigma$, $\sum_{q' \in Q} \delta(q, a, q') = 1$.

Acceptance/Rejection Probability

For $\alpha \in \Sigma^\omega$, let $\alpha[0:j]$ denote the prefix of length $j+1$. The probability of rejecting and accepting α is defined as follows.

$$\mathrm{rej}(\alpha) = \lim_{j\to\infty} \delta_{\alpha[0:j]}(q_s, q_r)$$

Acceptance/Rejection Probability

For $\alpha \in \Sigma^\omega$, let $\alpha[0:j]$ denote the prefix of length $j+1$. The probability of rejecting and accepting α is defined as follows.

$$\mathrm{rej}(\alpha) = \lim_{j \to \infty} \delta_{\alpha[0:j]}(q_s, q_r)$$
$$\mathrm{acc}(\alpha) = 1 - \mathrm{rej}(\alpha)$$

Acceptance/Rejection Probability

For $\alpha \in \Sigma^\omega$, let $\alpha[0:j]$ denote the prefix of length $j+1$. The probability of rejecting and accepting α is defined as follows.

$$\mathrm{rej}(\alpha) = \lim_{j \to \infty} \delta_{\alpha[0:j]}(q_s, q_r)$$
$$\mathrm{acc}(\alpha) = 1 - \mathrm{rej}(\alpha)$$

Given $\lambda \in [0,1]$, $\mathcal{L}_{>\lambda}(\mathcal{M})$ is the set of words α accepted with probability $> \lambda$.

Strong and Weak Monitors

Property L is monitorable

- strongly if there is an $\mathcal{M}$ such that $\mathcal{L}_{=1}(\mathcal{M}) = L$

Strong and Weak Monitors

Property L is monitorable

- strongly if there is an $\mathcal{M}$ such that $\mathcal{L}_{=1}(\mathcal{M}) = L$; no false alarms

Strong and Weak Monitors

Property L is monitorable

- strongly if there is an $\mathcal{M}$ such that $\mathcal{L}_{=1}(\mathcal{M}) = L$; no false alarms
- weakly if there is an $\mathcal{M}$ such that $\mathcal{L}_{>0}(\mathcal{M}) = L$

Strong and Weak Monitors

Property L is monitorable

- strongly if there is an $\mathcal{M}$ such that $\mathcal{L}_{=1}(\mathcal{M}) = L$; no false alarms
- weakly if there is an $\mathcal{M}$ such that $\mathcal{L}_{>0}(\mathcal{M}) = L$; no missed alarms

Expressive Power of Randomized Monitors

Deterministic Monitoring [Schneider]

Properties monitored deterministically are safety properties

- $L \subseteq \Sigma^\omega$ is a safety property if $\alpha \notin L$ iff there is a prefix $\alpha[0:i]$ such that $\alpha[0:i]\Sigma^\omega \subseteq \overline{L}$.

Expressive Power of Randomized Monitors

Deterministic Monitoring [Schneider]

Properties monitored deterministically are safety properties

- $L \subseteq \Sigma^\omega$ is a safety property if $\alpha \notin L$ iff there is a prefix $\alpha[0:i]$ such that $\alpha[0:i]\Sigma^\omega \subseteq \overline{L}$.

Randomized Monitoring [Chadha-Sistla-V. 2008]

Strong There is FPM $\mathcal{M}$ such that $L = \mathcal{L}_{=1}(\mathcal{M})$ iff L is a regular, safety property.

Expressive Power of Randomized Monitors

Deterministic Monitoring [Schneider]

Properties monitored deterministically are safety properties

- $L \subseteq \Sigma^\omega$ is a safety property if $\alpha \notin L$ iff there is a prefix $\alpha[0:i]$ such that $\alpha[0:i]\Sigma^\omega \subseteq \overline{L}$.

Randomized Monitoring [Chadha-Sistla-V. 2008]

Strong There is FPM $\mathcal{M}$ such that $L = \mathcal{L}_{=1}(\mathcal{M})$ iff L is a regular, safety property.

Weak There are FPMs $\mathcal{M}$ such that $\mathcal{L}_{>0}(\mathcal{M})$ is a non-regular, persistence property.

Expressive Power of Randomized Monitors

Deterministic Monitoring [Schneider]

Properties monitored deterministically are safety properties

- $L \subseteq \Sigma^\omega$ is a safety property if $\alpha \notin L$ iff there is a prefix $\alpha[0:i]$ such that $\alpha[0:i]\Sigma^\omega \subseteq \overline{L}$.

Randomized Monitoring [Chadha-Sistla-V. 2008]

Strong There is FPM $\mathcal{M}$ such that $L = \mathcal{L}_{=1}(\mathcal{M})$ iff L is a regular, safety property.

Weak There are FPMs $\mathcal{M}$ such that $\mathcal{L}_{>0}(\mathcal{M})$ is a non-regular, persistence property.

- L is a persistence property if it is a countable union of safety properties, i.e., "eventually always"-type properties

Safe LTL

[Sistla 1985]

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid$$ Boolean operators

$$X\varphi \mid \varphi \, R \, \varphi \mid \varphi \, U \, \varphi$$ Restricted to R

Safe LTL

[Sistla 1985]

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid$$ Boolean operators

$$X\varphi \mid \varphi\, R\, \varphi \mid \varphi\, U\, \varphi$$ Restricted to R

Strong Monitors for Safe LTL

Proposition (Kini-V. 2014)

For every Safe LTL formula φ, there is $\mathcal{M}_\varphi$ of size $O(2^{|\varphi|})$ such that $[\![\varphi]\!] = \mathcal{L}_{=1}(\mathcal{M})$.

Strong Monitors for Safe LTL

Proposition (Kini-V. 2014)

For every Safe LTL formula φ, there is $\mathcal{M}_\varphi$ of size $O(2^{|\varphi|})$ such that $[\![\varphi]\!] = \mathcal{L}_{=1}(\mathcal{M})$.

Proof.

- Construct nondeterministic Büchi automaton using [Vardi 1996]-method for $\neg\varphi$; the automaton has a single, absorbing accept state.
- Assign arbitrary probability to nondeterministic choices, and make accept state the unique reject state of FPM. □

Weak Monitors for Safe LTL

Theorem (Kini-V. 2014)

There are Safe LTL formulas φ such that the smallest FPM $\mathcal{M}$ with $\mathcal{L}_{>0}(\mathcal{M}) = [\![\varphi]\!]$ has at least doubly exponential states.

Weak Monitors for Safe LTL

Theorem (Kini-V. 2014)

There are Safe LTL formulas φ such that the smallest FPM $\mathcal{M}$ with $\mathcal{L}_{>0}(\mathcal{M}) = [\![\varphi]\!]$ has at least doubly exponential states.

Weak monitors are computationally more powerful than strong monitors but only as "efficient" as deterministic monitors for Safe LTL.

Weakly monitoring LTL(G)

LTL(G)

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \quad \text{Boolean operations}$$
$$X\varphi \mid G\varphi \quad \text{Restricted to } G$$

where $G\varphi \equiv \bot\, R\, \varphi$

Weakly monitoring LTL(G)

LTL(G)

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \qquad \text{Boolean operations}$$
$$X\varphi \mid G\varphi \qquad \text{Restricted to } G$$

where $G\varphi \equiv \bot \; R \; \varphi$

- [Alur-LaTorre 2004] Smallest deterministic machines for LTL(G) has doubly exponential states.

Weakly monitoring LTL(G)

LTL(G)

$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid$	Boolean operations
$X\varphi \mid G\varphi$	Restricted to G

where $G\varphi \equiv \bot \; R \; \varphi$

- [Alur-LaTorre 2004] Smallest deterministic machines for LTL(G) has doubly exponential states.
- [Kini-V. 2014] For every LTL(G) formula φ there is an FPM $\mathcal{M}$ such that $\mathcal{L}_{>0}(\mathcal{M}) = [\![\varphi]\!]$ and $\mathcal{M}$ has $O(2^{|\varphi|})$ states.

Communication Complexity
[Yao 1982]

Setup

Problem described by function $f : X \times Y \rightarrow \{0, 1\}$, where X, Y are finite sets.

- Alice is given input $x \in X$ and Bob is given input $y \in Y$
- Alice and Bob arbitrary computational devices and can toss coins
- Alice and Bob can send and receive messages

Goal

How bits need to be communicated for Bob to compute $f(x, y)$?

Set Membership

Problem

For a set S, take $X = 2^S$ and $Y = S$. Define $g^S : X \times Y \to \{0, 1\}$ such that $g^S(x, y) = 1$ iff $y \in x$.

Set Membership

Problem

For a set S, take $X = 2^S$ and $Y = S$. Define $g^S : X \times Y \to \{0, 1\}$ such that $g^S(x, y) = 1$ iff $y \in x$.

One Round Randomized Protocol

In this model, both Alice and Bob can toss coins, but Bob has to compute the answer based on single message sent by Alice.

- $R_\epsilon^{A \to B}(f)$ is the fewest number of bits that Alice needs to send to Bob so that Bob can compute f with error at most ϵ.

Set Membership

Problem

For a set S, take $X = 2^S$ and $Y = S$. Define $g^S : X \times Y \rightarrow \{0, 1\}$ such that $g^S(x, y) = 1$ iff $y \in x$.

One Round Randomized Protocol

In this model, both Alice and Bob can toss coins, but Bob has to compute the answer based on single message sent by Alice.

- $R_\epsilon^{A \rightarrow B}(f)$ is the fewest number of bits that Alice needs to send to Bob so that Bob can compute f with error at most ϵ.

Theorem (Kremer-Nisan-Ron 1995)

$R_\epsilon^{A \rightarrow B}(g^S) = \Omega(2^{|S|})$.

Hard Property to Weakly Monitor

For alphabet $\Sigma = \{0, 1, \#, \$\}$ define the following languages

$S_n = (\#(0+1)^n)^+\$(0+1)^n$	membership query
$R'_n = \{(\#(0+1)^n)^*(\#w)(\#(0+1)^n)^*\$w \mid w \in (0+1)^n\}$	positive query
$R_n - S_n \setminus R'_n$	negative query
$L_n = R_n^\omega + R_n^*(\#(0+1)^n)^\omega$	

Hard Property to Weakly Monitor

For alphabet $\Sigma = \{0, 1, \#, \$\}$ define the following languages

$S_n = (\#(0+1)^n)^+ \$(0+1)^n$	membership query
$R'_n = \{(\#(0+1)^n)^* (\#w)(\#(0+1)^n)^* \$w \mid w \in (0+1)^n\}$	positive query
$R_n - S_n \setminus R'_n$	negative query
$L_n = R_n^\omega + R_n^* (\#(0+1)^n)^\omega$	

[Kupferman-Rosenberg 2010] There is φ_n such that $[\![\varphi_n]\!] = L_n$ and $|\varphi_n| = n \log n$

Protocol from Monitor

Lemma

For any ϵ and $\mathcal{M}_n$ such that $\mathcal{L}_{>0}(\mathcal{M}_n) = L_n$, there is a state q_ϵ, reachable through an input in R_n^* such that every $\beta \in L_n$ is accepted with probability $\geq 1 - \epsilon$ from q_ϵ.

Protocol from Monitor

Lemma

For any ϵ and $\mathcal{M}_n$ such that $\mathcal{L}_{>0}(\mathcal{M}_n) = L_n$, there is a state q_ϵ, reachable through an input in R_n^* such that every $\beta \in L_n$ is accepted with probability $\geq 1 - \epsilon$ from q_ϵ.

Protocol

For $S = (0+1)^n$, a protocol for g^S from $\mathcal{M}_n$ is as follows.

1. Let w_x be input corresponding to Alice's input x. Alice runs $\mathcal{M}_n$ on w_x from q_ϵ and sends the state q reached to Bob.
2. Bob checks if $\$y(\#0^n)^\omega$ is accepted from q

Protocol from Monitor

Lemma

For any ϵ and $\mathcal{M}_n$ such that $\mathcal{L}_{>0}(\mathcal{M}_n) = L_n$, there is a state q_ϵ, reachable through an input in R_n^* such that every $\beta \in L_n$ is accepted with probability $\geq 1 - \epsilon$ from q_ϵ.

Protocol

For $S = (0+1)^n$, a protocol for g^S from $\mathcal{M}_n$ is as follows.

1. Let w_x be input corresponding to Alice's input x. Alice runs $\mathcal{M}_n$ on w_x from q_ϵ and sends the state q reached to Bob.
2. Bob checks if $\$y(\#0^n)^\omega$ is accepted from q

Bits communicated $= \log |\mathcal{M}_n| \geq 2^n$

Fragments of LTL

LTL(F, G)

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \qquad \text{Boolean operations}$$
$$X\varphi \mid F\varphi \mid G\varphi \qquad \text{Restricted to } F \text{ and } G$$

where $F\varphi \equiv \top \; U \; \varphi$

Fragments of LTL

LTL(F, G)

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \qquad \text{Boolean operations}$$
$$X\varphi \mid F\varphi \mid G\varphi \qquad \text{Restricted to } F \text{ and } G$$

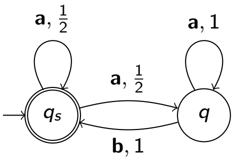
where $F\varphi \equiv \top\ U\ \varphi$

LTL $\setminus$ GU [Kretinsky-Esparza 2012]

$$\psi ::= \varphi \mid \psi \wedge \psi \mid \psi \vee \psi \mid \qquad \varphi \in \text{LTL}(F, G)$$
$$X\psi \mid \psi\ U\ \psi \qquad U \text{ above } G$$

Probabilistic Büchi Automata

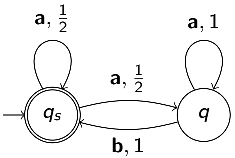
[Baier-Größer 2005]



A PBA is like an FPM except that it does not have a reject state and instead as final states.

Probabilistic Büchi Automata

[Baier-Größer 2005]

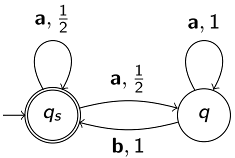


A PBA is like an FPM except that it does not have a reject state and instead as final states.

- An execution is accepting if it visits some final state infinitely often
- The acceptance probability of a word α, $\mathrm{acp}(\alpha)$, is the measure of all accepting executions on α.

Probabilistic Büchi Automata

[Baier-Größer 2005]



A PBA is like an FPM except that it does not have a reject state and instead as final states.

- An execution is accepting if it visits some final state infinitely often
- The acceptance probability of a word α, $\mathrm{acp}(\alpha)$, is the measure of all accepting executions on α.
- $\mathcal{L}_{>0}(\mathcal{M})$ and $\mathcal{L}_{=1}(\mathcal{M})$ defined similarly.

PBA for LTL \ *GU*

Theorem (Kini-V. 2015)

For every φ in LTL \ GU there is a PBA $\mathcal{M}_\varphi$ such that $\mathcal{M}_\varphi$ has $O(2^{|\varphi|})$ states and $\mathcal{L}_{>0}(\mathcal{M}) = [\![\varphi]\!]$.

Simplifying Assumptions

- Focus on LTL(F, G)

Simplifying Assumptions

- Focus on LTL(F, G)
- Also, assume there are no X operators

Limit Deterministic Automata

Courcoubetis-Yannakakis 1995

Limit deterministic automata: Nondeterministic automata such that every state reachable from a final state is deterministic

Limit Deterministic Automata

Courcoubetis-Yannakakis 1995

Limit deterministic automata: Nondeterministic automata such that every state reachable from a final state is deterministic

Theorem (Baier-Größer 2005)

Let $\mathcal{N}$ be a nondeterministic Büchi automaton. Let $\mathcal{M}$ be the PBA obtained assigning some probabilities to the nondeterministic choices. Then $\mathcal{L}_{>0}(\mathcal{M}) = \mathcal{L}(\mathcal{N})$.

Limit Deterministic Automata

Courcoubetis-Yannakakis 1995

Limit deterministic automata: Nondeterministic automata such that every state reachable from a final state is deterministic

Theorem (Baier-Größer 2005)

Let $\mathcal{N}$ be a nondeterministic Büchi automaton. Let $\mathcal{M}$ be the PBA obtained assigning some probabilities to the nondeterministic choices. Then $\mathcal{L}_{>0}(\mathcal{M}) = \mathcal{L}(\mathcal{N})$.

We will construct a limit deterministic automaton for LTL $\setminus$ GU.

"Standard" LTL to NBA translation

Idea: Automaton guesses which temporal subformulas are true at each step.

"Standard" LTL to NBA translation

Idea: Automaton guesses which temporal subformulas are true at each step. Consider $\varphi = G(a \vee Fb)$.

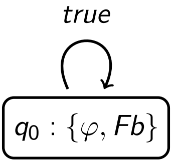
"Standard" LTL to NBA translation

Idea: Automaton guesses which temporal subformulas are true at each step. Consider $\varphi = G(a \vee Fb)$.

$$q_0 : \{\varphi, Fb\}$$

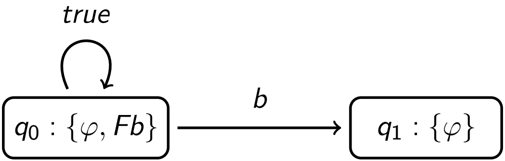
"Standard" LTL to NBA translation

Idea: Automaton guesses which temporal subformulas are true at each step. Consider $\varphi = G(a \vee Fb)$.



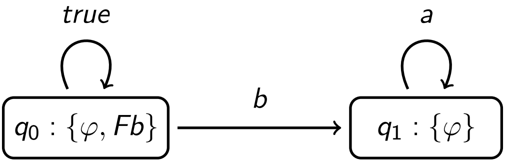
"Standard" LTL to NBA translation

Idea: Automaton guesses which temporal subformulas are true at each step. Consider $\varphi = G(a \vee Fb)$.



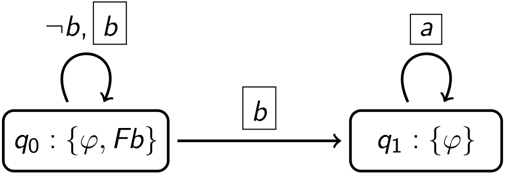
"Standard" LTL to NBA translation

Idea: Automaton guesses which temporal subformulas are true at each step. Consider $\varphi = G(a \vee Fb)$.



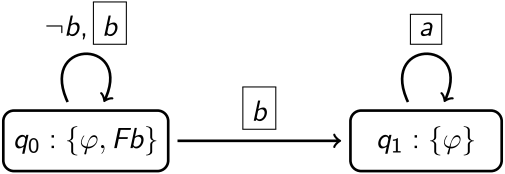
"Standard" LTL to NBA translation

Idea: Automaton guesses which temporal subformulas are true at each step. Consider $\varphi = G(a \vee Fb)$.



"Standard" LTL to NBA translation

Idea: Automaton guesses which temporal subformulas are true at each step. Consider $\varphi = G(a \vee Fb)$.



Automaton is not limit deterministic!

Construction for LTL(F, G)

Intuition

Observation

For any formula φ over propositions P, any word $w \in (2^P)^\omega$ satisfies exactly one of the following

$$G\varphi \quad F\varphi \wedge \neg G\varphi \quad \neg F\varphi$$

What does this mean for F, G subformulas?

$$G\varphi \quad F\varphi \wedge \neg G\varphi \quad \neg F\varphi$$

What does this mean for F, G subformulas?

$$G\varphi \quad F\varphi \wedge \neg G\varphi \quad \neg F\varphi$$

	A	B	C
$G\psi$			
$F\psi$			

What does this mean for F, G subformulas?

$G\varphi$ $\quad F\varphi \wedge \neg G\varphi \quad \neg F\varphi$

	A	B	C
$G\psi$	$G\psi$		
$F\psi$			

What does this mean for F, G subformulas?

$G\varphi$ $\quad F\varphi \wedge \neg G\varphi$ $\quad \neg F\varphi$

	A	B	C
$G\psi$	$G\psi$	$\neg G\psi \wedge FG\psi$	
$F\psi$			

What does this mean for F, G subformulas?

$G\varphi \quad F\varphi \wedge \neg G\varphi \quad \neg F\varphi$

	A	B	C
$G\psi$	$G\psi$	$\neg G\psi \wedge FG\psi$	$\neg FG\psi$
$F\psi$			

What does this mean for F, G subformulas?

$G\varphi \quad F\varphi \wedge \neg G\varphi \quad \neg F\varphi$

	A	B	C
$G\psi$	$G\psi$	$\neg G\psi \wedge FG\psi$	$\neg FG\psi$
$F\psi$	$\neg F\psi$		

What does this mean for F, G subformulas?

$G\varphi$ $F\varphi \wedge \neg G\varphi$ $\neg F\varphi$

	A	B	C
$G\psi$	$G\psi$	$\neg G\psi \wedge FG\psi$	$\neg FG\psi$
$F\psi$	$\neg F\psi$	$F\psi \wedge \neg GF\psi$	

What does this mean for F, G subformulas?

$G\varphi$ $\quad F\varphi \wedge \neg G\varphi \quad \neg F\varphi$

	A	B	C
$G\psi$	$G\psi$	$\neg G\psi \wedge FG\psi$	$\neg FG\psi$
$F\psi$	$\neg F\psi$	$F\psi \wedge \neg GF\psi$	$GF\psi$

Construction for LTL(F, G)

Overview

- A state is a guess about how often each F, G subformula holds.

example

Construction for LTL(F, G)

Overview

- A state is a guess about how often each F, G subformula holds.
- The automaton checks if the guess is sound

example

Construction for LTL(F, G)

Overview

- A state is a guess about how often each F, G subformula holds.
- The automaton checks if the guess is sound
  - A guess is sound if every $G\psi \in \pi_A$ is true and every $F\psi \notin \pi_A$ is true.

example

Construction for LTL(F, G)

Evaluation

	A	B	C
$G\psi$	$G\psi$	$\neg G\psi \wedge FG\psi$	$\neg FG\psi$
$F\psi$	$\neg F\psi$	$F\psi \wedge \neg GF\psi$	$GF\psi$

Construction for LTL(F, G)

Evaluation

	A	B	C
$G\psi$	$G\psi$	$\neg G\psi \wedge FG\psi$	$\neg FG\psi$
$F\psi$	$\neg F\psi$	$F\psi \wedge \neg GF\psi$	$GF\psi$

The evaluation of φ denoted by $[\varphi]^{\pi}_{\nu}$ is the truth of φ at present with respect to the guess π and input $\nu \in 2^P$.

Construction for LTL(F, G)

Evaluation

	A	B	C
$G\psi$	$G\psi$	$\neg G\psi \wedge FG\psi$	$\neg FG\psi$
$F\psi$	$\neg F\psi$	$F\psi \wedge \neg GF\psi$	$GF\psi$

The evaluation of φ denoted by $[\varphi]^{\pi}_{\nu}$ is the truth of φ at present with respect to the guess π and input $\nu \in 2^P$.

- truth of propositions obtained from input ν

Construction for LTL(F, G)

Evaluation

	A	B	C
$G\psi$	$G\psi$	$\neg G\psi \wedge FG\psi$	$\neg FG\psi$
$F\psi$	$\neg F\psi$	$F\psi \wedge \neg GF\psi$	$GF\psi$

The evaluation of φ denoted by $[\varphi]_\nu^\pi$ is the truth of φ at present with respect to the guess π and input $\nu \in 2^P$.

- truth of propositions obtained from input ν
- boolean connectives evaluated using their semantics

Construction for LTL(F, G)

Evaluation

	A	B	C
$G\psi$	$G\psi$	$\neg G\psi \wedge FG\psi$	$\neg FG\psi$
$F\psi$	$\neg F\psi$	$F\psi \wedge \neg GF\psi$	$GF\psi$

The evaluation of φ denoted by $[\varphi]^{\pi}_{\nu}$ is the truth of φ at present with respect to the guess π and input $\nu \in 2^P$.

- truth of propositions obtained from input ν
- boolean connectives evaluated using their semantics
- $[G\psi]^{\pi}_{\nu}$ is true iff $G\psi \in \pi_A$ and $[F\psi]^{\pi}_{\nu}$ is true iff $F\psi \notin \pi_A$.

Construction for LTL(F, G)

	A	B	C
$G\psi$	$G\psi$	$\neg G\psi \wedge FG\psi$	$\neg FG\psi$
$F\psi$	$\neg F\psi$	$F\psi \wedge \neg GF\psi$	$GF\psi$

$$\pi \xrightarrow{\quad \nu \quad} \rho$$

Construction for LTL(F, G)

	A	B	C
$G\psi$	$G\psi$	$\neg G\psi \wedge FG\psi$	$\neg FG\psi$
$F\psi$	$\neg F\psi$	$F\psi \wedge \neg GF\psi$	$GF\psi$

π	$\xrightarrow{\nu}$	ρ
$G\psi \in \pi_A$		$G\psi \in \rho_A$

Ensure ψ is true by evaluating it

Construction for LTL(F, G)

	A	B	C
$G\psi$	$G\psi$	$\neg G\psi \wedge FG\psi$	$\neg FG\psi$
$F\psi$	$\neg F\psi$	$F\psi \wedge \neg GF\psi$	$GF\psi$

π	$\xrightarrow{\nu}$	ρ
$G\psi \in \pi_B$		$G\psi \in \rho_A \cup \rho_B$

No need to check $G\psi$ is false

Construction for LTL(F, G)

	A	B	C
$G\psi$	$G\psi$	$\neg G\psi \wedge FG\psi$	$\neg FG\psi$
$F\psi$	$\neg F\psi$	$F\psi \wedge \neg GF\psi$	$GF\psi$

π	$\xrightarrow{\nu}$	ρ
$G\psi \in \pi_C$		$G\psi \in \rho_C$

No need to check $FG\psi$ is false

Construction for LTL(F, G)

	A	B	C
$G\psi$	$G\psi$	$\neg G\psi \wedge FG\psi$	$\neg FG\psi$
$F\psi$	$\neg F\psi$	$F\psi \wedge \neg GF\psi$	$GF\psi$

π	$\xrightarrow{\nu}$	ρ
$F\psi \in \pi_A$		$F\psi \in \rho_A$

No need to check $F\psi$ is false

Construction for LTL(F, G)

	A	B	C
$G\psi$	$G\psi$	$\neg G\psi \wedge FG\psi$	$\neg FG\psi$
$F\psi$	$\neg F\psi$	$F\psi \wedge \neg GF\psi$	$GF\psi$

π	$\xrightarrow{\nu}$	ρ
$F\psi \in \pi_B$		$F\psi \in \rho_A \cup \rho_B$

If $F\psi$ moves to A ensure ψ is true

Construction for LTL(F, G)

	A	B	C
$G\psi$	$G\psi$	$\neg G\psi \wedge FG\psi$	$\neg FG\psi$
$F\psi$	$\neg F\psi$	$F\psi \wedge \neg GF\psi$	$GF\psi$

π	$\xrightarrow{\nu}$	ρ
$F\psi \in \pi_C$		$F\psi \in \rho_C$

Check that ψ holds infinitely often: use a counter!

Construction for LTL(F, G)

States

An automaton state is a pair (π, n)

Construction for LTL(F, G)

States

An automaton state is a pair (π, n)

- π is current guess for φ

Construction for LTL(F, G)

States

An automaton state is a pair (π, n)

- π is current guess for φ
- $n \in \{0, 1, \ldots k\}$ where k is the number of F formulas in π_C

Construction for LTL(F, G)
Transitions

Transitions should help check if the guess is sound

$$(\pi, m) \quad \xrightarrow{\nu} \quad (\rho, n)$$

Construction for LTL(F, G)

Transitions

Transitions should help check if the guess is sound

$$(\pi, m) \xrightarrow{\nu} (\rho, n)$$

- $\pi_A \subseteq \rho_A \quad \pi_B \supseteq \rho_B \quad \pi_C = \rho_C$ (component π_B is non-increasing)

Construction for LTL(F, G)

Transitions

Transitions should help check if the guess is sound

$$(\pi, m) \quad \xrightarrow{\nu} \quad (\rho, n)$$

- $\pi_A \subseteq \rho_A \quad \pi_B \supseteq \rho_B \quad \pi_C = \rho_C$ (component π_B is non-increasing)
- for $G\psi \in \pi_A$, $[\psi]^{\pi}_{\nu}$ is true

Construction for LTL(F, G)

Transitions

Transitions should help check if the guess is sound

$$(\pi, m) \quad \xrightarrow{\nu} \quad (\rho, n)$$

- $\pi_A \subseteq \rho_A \quad \pi_B \supseteq \rho_B \quad \pi_C = \rho_C$ (component π_B is non-increasing)
- for $G\psi \in \pi_A$, $[\psi]^{\pi}_{\nu}$ is true
- for $F\psi \in \pi_B$, $[\psi]^{\pi}_{\nu}$ is false implies $F\psi \in \rho_B$

Construction for LTL(F, G)

Transitions

Transitions should help check if the guess is sound

$$(\pi, m) \xrightarrow{\nu} (\rho, n)$$

- $\pi_A \subseteq \rho_A \quad \pi_B \supseteq \rho_B \quad \pi_C = \rho_C$ (component π_B is non-increasing)
- for $G\psi \in \pi_A$, $[\psi]^{\pi}_{\nu}$ is true
- for $F\psi \in \pi_B$, $[\psi]^{\pi}_{\nu}$ is false implies $F\psi \in \rho_B$ (delayed forever?)

Construction for LTL(F, G)

Transitions

Transitions should help check if the guess is sound

$$(\pi, m) \quad \xrightarrow{\nu} \quad (\rho, n)$$

- $\pi_A \subseteq \rho_A \quad \pi_B \supseteq \rho_B \quad \pi_C = \rho_C$ (component π_B is non-increasing)
- for $G\psi \in \pi_A$, $[\psi]^{\pi}_{\nu}$ is true
- for $F\psi \in \pi_B$, $[\psi]^{\pi}_{\nu}$ is false implies $F\psi \in \rho_B$ (delayed forever?)
- increment counter if $m = 0$ or the m^{th} F-formula in π_C evaluates to true

Construction for LTL(F, G)

Acceptance Condition

Büchi Condition: A state $(\pi, 0)$ is final if π_B is empty.

Construction for LTL(F, G)

Acceptance Condition

Büchi Condition: A state $(\pi, 0)$ is final if π_B is empty.

- empty π_B ensure obligations are eventually met

Construction for LTL(F, G)

Acceptance Condition

Büchi Condition: A state $(\pi, 0)$ is final if π_B is empty.

- empty π_B ensure obligations are eventually met
- Büchi condition ensures counter incremented infinitely often

Construction for LTL(F, G)

Acceptance Condition

Büchi Condition: A state $(\pi, 0)$ is final if π_B is empty.

- empty π_B ensure obligations are eventually met
- Büchi condition ensures counter incremented infinitely often

Together they ensure that every guess in an accepting run is sound.

Construction for LTL(F, G)

Initial Conditions

A transition $(\pi, 0) \xrightarrow{\nu} (\rho, n)$ is initial if $[\varphi]^{\pi}_{\nu}$ is true.
Since initial guess is sound in an accepting run, the truth of φ is ensured.

Construction for LTL(F, g)

Limit Determinism

Limit determinism is ensured because

- Once π_B becomes empty, the guess π cannot change across transitions
- Counter is incremented deterministically

Example

Consider $\varphi = G(a \vee Fb)$

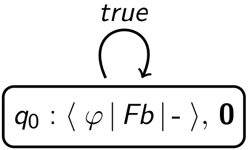
Example

Consider $\varphi = G(a \vee Fb)$

$$q_0 : \langle\ \varphi \,|\, Fb \,|\, \text{-}\ \rangle,\ \mathbf{0}$$

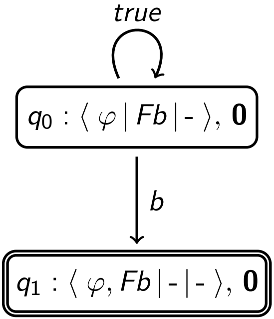
Example

Consider $\varphi = G(a \vee Fb)$



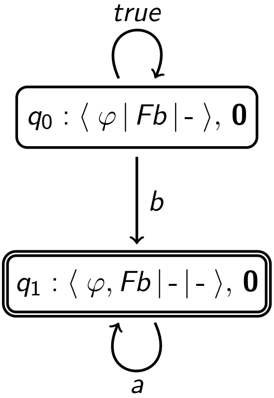
Example

Consider $\varphi = G(a \vee Fb)$



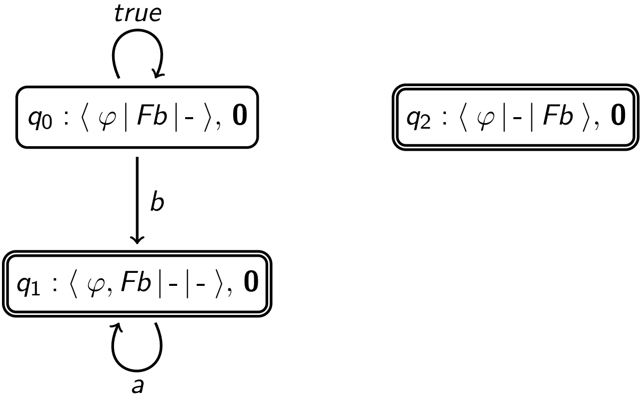
Example

Consider $\varphi = G(a \vee Fb)$



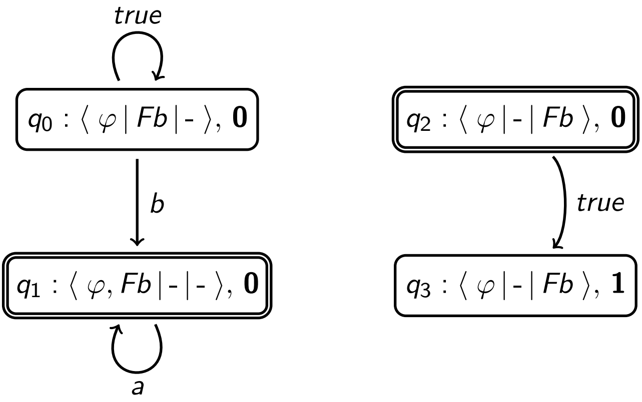
Example

Consider $\varphi = G(a \vee Fb)$



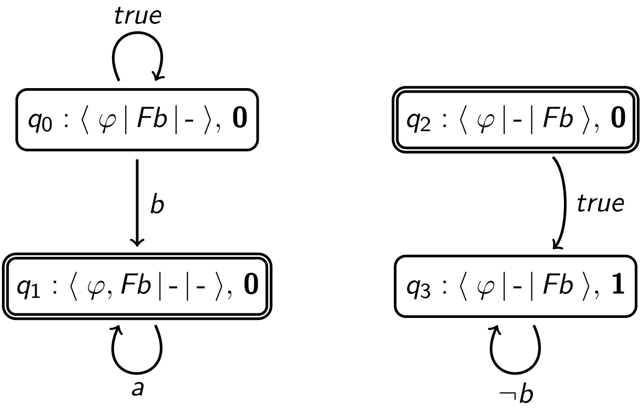
Example

Consider $\varphi = G(a \vee Fb)$



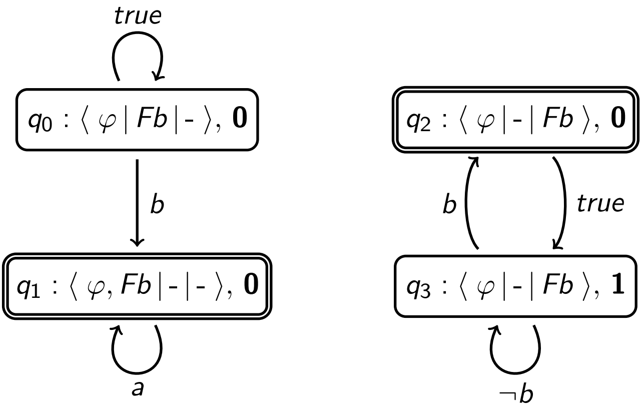
Example

Consider $\varphi = G(a \vee Fb)$



Example

Consider $\varphi = G(a \vee Fb)$



Markov Decision Processes

- States divided into probabilistic and nondeterministic. From a probabilistic state, the next state is chosen by tossing a coin, and from a nondeterministic state, the next state is chosen nondeterministically

Markov Decision Processes

- States divided into probabilistic and nondeterministic. From a probabilistic state, the next state is chosen by tossing a coin, and from a nondeterministic state, the next state is chosen nondeterministically
- Models (closed) concurrent, stochastic programs

Markov Decision Processes

- States divided into probabilistic and nondeterministic. From a probabilistic state, the next state is chosen by tossing a coin, and from a nondeterministic state, the next state is chosen nondeterministically
- Models (closed) concurrent, stochastic programs
- Nondeterminism resolved by a scheduler

Model Checking Problem

Given and MDP A and LTL formula φ, is there a scheduler S such that the set of executions of A^S that satisfy φ has probability > 0?

Model Checking Problem

Given and MDP A and LTL formula φ, is there a scheduler S such that the set of executions of A^S that satisfy φ has probability > 0?

- [Courcoubetis-Yannakakis 1995] The problem is 2EXPTIME-complete for LTL specs

Model Checking Problem

Given and MDP A and LTL formula φ, is there a scheduler S such that the set of executions of A^S that satisfy φ has probability > 0?

- [Courcoubetis-Yannakakis 1995] The problem is 2EXPTIME-complete for LTL specs
- Upper bound relies on analyzing the cross-product of the MDP with a limit deterministic automaton for φ.

Model Checking Problem

Given and MDP A and LTL formula φ, is there a scheduler S such that the set of executions of A^S that satisfy φ has probability > 0?

- [Courcoubetis-Yannakakis 1995] The problem is 2EXPTIME-complete for LTL specs
- Upper bound relies on analyzing the cross-product of the MDP with a limit deterministic automaton for φ.
- [Kini-V. 2015] The problem is EXPTIME-complete for LTL $\setminus$ GU specs

Wrapup

- Ideas can be generalized to construct limit deterministic automata for full LTL

Wrapup

- Ideas can be generalized to construct limit deterministic automata for full LTL but it is doubly exponential size

Wrapup

- Ideas can be generalized to construct limit deterministic automata for full LTL but it is doubly exponential size
- Can it be improved?

Wrapup

- Ideas can be generalized to construct limit deterministic automata for full LTL but it is doubly exponential size
- Can it be improved?
    - No lower bound proof, but unlikely

Wrapup

- Ideas can be generalized to construct limit deterministic automata for full LTL but it is doubly exponential size
- Can it be improved?
  - No lower bound proof, but unlikely
- Implementation of translation
  `http://web.engr.illinois.edu/ kini2/buchifier/`