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

ϕ ::= p | ¬p | ϕ ∧ ϕ | ϕ ∨ ϕ |
Xϕ | ϕ U ϕ | ϕ R ϕ

- **Boolean operators**
- **Temporal operators**
Linear Temporal Logic (LTL)
[Pnueli 1977]

Syntax

\[ \varphi ::= p \mid \neg p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X \varphi \mid \varphi U \varphi \mid \varphi R \varphi \]

Boolean operators
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**

\[ \varphi ::= p \mid \neg p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X \varphi \mid \varphi U \varphi \mid \varphi R \varphi \]

- **Boolean operators**
- **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

\[ \varphi ::= p \mid \neg p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X \varphi \mid \varphi U \varphi \mid \varphi R \varphi \]

Boolean operators
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.

\[ \alpha \models p \text{ iff } p \in \alpha[0] \]
Linear Temporal Logic (LTL)

[1] Pnueli 1977

Syntax

\[ \varphi ::= p \mid \neg p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X\varphi \mid \varphi U \varphi \mid \varphi R \varphi \]

Boolean operators

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.

- \( \alpha \models p \iff p \in \alpha[0] \)
- \( \alpha \models X\varphi \iff \alpha[1 : \infty] \models \varphi \)
Introduction
Safety Properties
General Properties

Need for new translations

Linear Temporal Logic (LTL)
[Pnueli 1977]

Syntax

$$\varphi ::= p \mid \neg p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X\varphi \mid \varphi U \varphi \mid \varphi R \varphi$$

- Boolean operators
- 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

\[ \phi ::= p \mid \neg p \mid \phi \land \phi \mid \phi \lor \phi \mid \phi X \mid \phi U \mid \phi R \phi \]

- Boolean operators
- 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 \( p \in \alpha[0] \)
- \( \alpha \models X\phi \) iff \( \alpha[1 : \infty] \models \phi \)
- \( \alpha \models \phi U \psi \) iff there is \( j \) such that \( \alpha[j : \infty] \models \psi \) and for all \( i < j \) \( \alpha[i : \infty] \models \phi \)
- \( \alpha \models \phi R \psi \) iff either for every \( i \), \( \alpha[i : \infty] \models \psi \) or
Linear Temporal Logic (LTL)
[Pnueli 1977]

Syntax

\[ \varphi ::= p \mid \neg p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \varphi_X \mid \varphi_U \mid \varphi_R \]

Boolean operators

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 \( i < j \), \( \alpha[i : \infty] \models \psi \)
Theorem (Sistla-Vardi-Wolper 1985)

For every LTL formula $\varphi$, there is a nondeterministic Büchi automaton $M$ of size $O(2^{\left|\varphi\right|})$ such that $L(M) = \llbracket \varphi \rrbracket$.
Translating LTL to Automata

Theorem (Sistla-Vardi-Wolper 1985)

For every LTL formula $\varphi$, there is a nondeterministic Büchi automaton $M$ of size $O(2^{\mid \varphi \mid})$ such that $L(M) = \llbracket \varphi \rrbracket$

Applications

Gave first non-elementary decision procedure for

- Satisfiability and validity of LTL
Theorem (Sistla-Vardi-Wolper 1985)

For every LTL formula \( \varphi \), there is a nondeterministic Büchi automaton \( M \) of size \( O(2^{\|\varphi\|}) \) such that \( L(M) = \llbracket \varphi \rrbracket \)

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
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.
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
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?
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?
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

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.
Dynamic Analysis of Systems

System

Environment

Monitor

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.
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.

- 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

System → Monitor → Environment
Randomized Monitoring

The monitor has access to a private source of randomness. The system itself is not probabilistic.
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 $\Sigma$ 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 $a \in \Sigma$, $\sum_{q' \in Q} \delta(q, a, q') = 1$. 
For $\alpha \in \Sigma^\omega$, let $\alpha[0:j]$ denote the prefix of length $j + 1$. The probability of rejecting and accepting $\alpha$ is defined as follows.

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

$$\text{acc}(\alpha) = 1 - \text{rej}(\alpha)$$
For $\alpha \in \Sigma^\omega$, let $\alpha[0:j]$ denote the prefix of length $j + 1$. The probability of rejecting and accepting $\alpha$ is defined as follows.

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

$$\text{acc}(\alpha) = 1 - \text{rej}(\alpha)$$
For $\alpha \in \Sigma^\omega$, let $\alpha[0 : j]$ denote the prefix of length $j + 1$. The probability of rejecting and accepting $\alpha$ is defined as follows.

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

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

Property $L$ is monitorable

- **strongly** if there is an $M$ such that $L_{=1}(M) = L$
- **weakly** if there is an $M$ such that $L > 0 (M) = L$
Strong and Weak Monitors

Property $L$ is monitorable

- **strongly** if there is an $M$ such that $L_{=1}(M) = L$; 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 $M$ such that $L_{\geq 1}(M) = L$; no false alarms
- **weakly** if there is an $M$ such that $L_{>0}(M) = L$; no missed alarms
Deterministic Monitoring [Schneider]

Properties monitored deterministically are safety properties

- \( L \subseteq \Sigma^\omega \) is a safety property if \( \alpha \not\in L \) 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 \( \alpha \not\in L \) 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 \( L = \mathcal{L}_{\leq 1}(\mathcal{M}) \) 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 \( \alpha \not\in L \) iff there is a prefix \( \alpha[0 : i] \) such that \( \alpha[0 : i]\Sigma^\omega \subseteq \overline{L} \).

#### Randomized Monitoring [Chadha-Sistla-V. 2008]

<table>
<thead>
<tr>
<th>Strong</th>
<th>There is FPM ( M ) such that ( L = \mathcal{L}_{\leq 1}(M) ) iff ( L ) is a regular, safety property.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weak</td>
<td>There are FPMs ( M ) such that ( \mathcal{L}_{&gt;0}(M) ) is a non-regular, persistence property.</td>
</tr>
</tbody>
</table>
Expressive Power of Randomized Monitors

Deterministic Monitoring [Schneider]

Properties monitored deterministically are safety properties

- $L \subseteq \Sigma^\omega$ is a safety property if $\alpha \not\in L$ iff there is a prefix $\alpha[0 : i]$ such that $\alpha[0 : i]\Sigma^\omega \subseteq L$.

Randomized Monitoring [Chadha-Sistla-V. 2008]

**Strong** There is FPM $\mathcal{M}$ such that $L = \mathcal{L}_{\geq 1}(\mathcal{M})$ 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]

\[ \varphi ::= p \mid \neg p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \varphi X \varphi \mid \varphi R \varphi \mid \varphi U \varphi \]

Boolean operators
Restricted to \( R \)
Safe LTL

[Sistla 1985]

\[ \varphi ::= p \mid \neg p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X \varphi \mid \varphi \mathrel{R} \varphi \mid \varphi \mathrel{U} \varphi \]

Boolean operators

Restricted to \( R \)
Proposition (Kini-V. 2014)

For every Safe LTL formula $\varphi$, there is $M_{\varphi}$ of size $O(2^{\varphi})$ such that $\llbracket \varphi \rrbracket = L_{=1}(M)$. 

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.
Proposition (Kini-V. 2014)

For every Safe LTL formula $\varphi$, there is $M_\varphi$ of size $O(2^{\lvert \varphi \rvert})$ such that $\llbracket \varphi \rrbracket = L_{=1}(M)$.

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.
Theorem (Kini-V. 2014)

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

Theorem (Kini-V. 2014)

There are Safe LTL formulas $\varphi$ such that the smallest FPM $M$ with $\mathcal{L}_{>0}(M) = [\varphi]$ 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\((G)\)

\[\varphi ::= p \mid \neg p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X \varphi \mid G \varphi\]

Boolean operations

Restricted to \(G\)

where \(G \varphi \equiv \bot \land R \varphi\)
Weakly monitoring LTL($G$)

\[ \phi ::= p \mid \neg p \mid \phi \land \phi \mid \phi \lor \phi \mid X \phi \mid G \phi \]

Boolean operations

Restricted to $G$

where $G \phi \equiv \perp R \phi$

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

\[
\phi ::= p \mid \neg p \mid \psi \land \psi \mid \psi \lor \psi \mid X\psi \mid G\psi
\]

where \(G\psi \equiv \bot R \psi\)

- **[Alur-LaTorre 2004]** Smallest deterministic machines for LTL(\(G\)) has doubly exponential states.
- **[Kini-V. 2014]** For every LTL(\(G\)) formula \(\psi\) there is an FPM \(M\) such that \(\mathcal{L}_{>0}(M) = \llbracket \psi \rrbracket\) and \(M\) has \(O(2^{\mid \psi \mid})\) 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 $g^S : X \times Y \rightarrow \{0, 1\}$ 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.

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

**Theorem (Kremer-Nisan-Ron 1995)**
$$R_{\epsilon}^{A\rightarrow B}(g^S) = \Omega(2^{|S|}).$$
For alphabet $\Sigma = \{0, 1, \#, $\}$ define the following languages

\[ S_n = (\#(0 + 1)^n)^+ $(0 + 1)^n \]
\[ R'_n = \{((\#(0 + 1)^n)^* \#w)(\#(0 + 1)^n)^*$w \mid w \in (0 + 1)^n \} \text{ positive query} \]
\[ R_n - S_n \setminus R'_n \text{ negative query} \]
\[ L_n = R_n^\omega + R_n^* (\#(0 + 1)^n)^\omega \]
Hard Property to Weakly Monitor

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

- $S_n = (\#(0 + 1)^n)^+$(0 + 1)^n$
- $R'_n = \{((\#(0 + 1)^n)^*(\#w)(\#(0 + 1)^n)^*$w | w $\in (0 + 1)^n\}$
- $R_n = S_n \setminus R'_n$
- $L_n = R_\omega + R^*(\#(0 + 1)^n)\omega$

[Kupferman-Rosenberg 2010] There is $\varphi_n$ such that $[\varphi_n] = L_n$ and $|\varphi_n| = n \log n$
Protocol from Monitor

Lemma

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

**Lemma**

For any $\epsilon$ and $M_n$ such that $L_{>0}(M_n) = L_n$, there is a state $q_\epsilon$, reachable through an input in $R_n^*$ such that every $\beta \in L_n$ is accepted with probability $\geq 1 - \epsilon$ from $q_\epsilon$.

**Protocol**

For $S = (0 + 1)^n$, a protocol for $g^S$ from $M_n$ is as follows.

1. Let $w_x$ be input corresponding to Alice’s input $x$. Alice runs $M_n$ on $w_x$ from $q_\epsilon$ and sends the state $q$ reached to Bob.
2. Bob checks if $y(\#0^n) \omega$ is accepted from $q$.
Lemma

For any $\epsilon$ and $M_n$ such that $L_{>0}(M_n) = L_n$, there is a state $q_\epsilon$, reachable through an input in $R_n^*$ such that every $\beta \in L_n$ is accepted with probability $\geq 1 - \epsilon$ from $q_\epsilon$.

Protocol

For $S = (0 + 1)^n$, a protocol for $g^S$ from $M_n$ is as follows.

1. Let $w_x$ be input corresponding to Alice’s input $x$. Alice runs $M_n$ on $w_x$ from $q_\epsilon$ and sends the state $q$ reached to Bob.

2. Bob checks if $y(#0^n)^\omega$ is accepted from $q$

Bits communicated $= \log |M_n| \geq 2^n$
Fragments of LTL

LTL($F$, $G$)

$\varphi ::= p \mid \neg p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X\varphi \mid F\varphi \mid G\varphi$

Boolean operations

Restricted to $F$ and $G$

where $F\varphi \equiv \top \cup U \varphi$
Fragments of LTL

\[
\begin{align*}
\text{LTL}(F, G) & \\
\varphi & ::= p \mid \neg p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid X\varphi \mid F\varphi \mid G\varphi \\
\text{where } F\varphi & \equiv \top U \varphi
\end{align*}
\]

\[
\begin{align*}
\text{LTL} \setminus \text{GU} [\text{Kretinsky-Esparza 2012}] & \\
\psi & ::= \varphi \mid \psi \land \psi \mid \psi \lor \psi \mid X\psi \mid \psi U \psi \\
\varphi & \in \text{LTL}(F, G) \quad U \text{ above } G
\end{align*}
\]
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 $\alpha$, $\text{acp}(\alpha)$, is the measure of all accepting executions on $\alpha$. 
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 $\alpha$, $\text{acp}(\alpha)$, is the measure of all accepting executions on $\alpha$.
- $\mathcal{L}_{>0}(\mathcal{M})$ and $\mathcal{L}_{\geq 1}(\mathcal{M})$ defined similarly.
Theorem (Kini-V. 2015)

For every $\varphi$ in $\text{LTL} \setminus \text{GU}$ there is a PBA $M_\varphi$ such that $M_\varphi$ has $O(2^{\lvert \varphi \rvert})$ states and $\mathcal{L}_{>0}(M) = [\varphi]$. 
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* $\mathcal{L}_{>0}(\mathcal{M}) = \mathcal{L}(\mathcal{N})$. 
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 $\varphi = G(a \lor Fb)$.
“Standard” LTL to NBA translation

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

$q_0 : \{\varphi, Fb\}$
“Standard” LTL to NBA translation

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

$$q_0 : \{ \varphi, Fb \}$$
"Standard" LTL to NBA translation

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

![Diagram](true)

$q_0 : \{\varphi, Fb\} \xrightarrow{b} q_1 : \{\varphi\}$
“Standard” LTL to NBA translation

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

$\varphi = G(a \lor Fb)$

\begin{align*}
q_0 : \{\varphi, Fb\} & \quad b \quad q_1 : \{\varphi\} \\
\text{true} & \quad a \\
\end{align*}
“Standard” LTL to NBA translation

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

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

$q_1 : \{\varphi\}$
**“Standard” LTL to NBA translation**

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

\[\begin{align*}
q_0 : \{\varphi, Fb\} & \xrightarrow{b} q_1 : \{\varphi\} \\
\neg b, \boxed{b} & \quad q_0 \quad \boxed{a}
\end{align*}\]

Automaton is not limit deterministic!
Construction for LTL(F, G)

Intuition

Observation

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

$$G\varphi \quad F\varphi \land \neg G\varphi \quad \neg F\varphi$$
What does this mean for $F, G$ subformulas?

\[ G\varphi \quad F\varphi \land \neg G\varphi \quad \neg F\varphi \]
What does this mean for $F$, $G$ subformulas?

\[ G\varphi \quad F\varphi \land \neg G\varphi \quad \neg F\varphi \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G\psi$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$F\psi$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What does this mean for $F$, $G$ subformulas?

$G\varphi 
\quad F\varphi \land \neg G\varphi 
\quad \neg F\varphi$

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G\psi$</td>
<td>$G\psi$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$F\psi$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What does this mean for $F$, $G$ subformulas?

$\begin{align*}
G\varphi & \quad F\varphi \land \neg G\varphi \quad \neg F\varphi \\
\hline
G\psi & \quad G\psi & \quad \neg G\psi \land FG\psi \\
F\psi & \quad & \quad
\end{align*}$
What does this mean for $F$, $G$ subformulas?

$$G\varphi \quad F\varphi \land \neg G\varphi \quad \neg F\varphi$$

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G\psi$</td>
<td>$G\psi$</td>
<td>$\neg G\psi \land FG\psi$</td>
<td>$\neg FG\psi$</td>
</tr>
<tr>
<td>$F\psi$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What does this mean for $F$, $G$ subformulas?

$$G\varphi \quad F\varphi \land \neg G\varphi \quad \neg F\varphi$$

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G\psi$</td>
<td>$G\psi$</td>
<td>$\neg G\psi \land FG\psi$</td>
<td>$\neg FG\psi$</td>
</tr>
<tr>
<td>$F\psi$</td>
<td>$\neg F\psi$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What does this mean for $F, G$ subformulas?

\[
G\varphi \quad F\varphi \land \neg G\varphi \quad \neg F\varphi
\]

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G\psi$</td>
<td>$G\psi$</td>
<td>$\neg G\psi \land FG\psi$</td>
<td>$\neg FG\psi$</td>
</tr>
<tr>
<td>$F\psi$</td>
<td>$\neg F\psi$</td>
<td>$F\psi \land \neg GF\psi$</td>
<td></td>
</tr>
</tbody>
</table>
What does this mean for $F, G$ subformulas?

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G \psi$</td>
<td>$G \psi$</td>
<td>$\neg G \psi \land FG \psi$</td>
<td>$\neg FG \psi$</td>
</tr>
<tr>
<td>$F \psi$</td>
<td>$\neg F \psi$</td>
<td>$F \psi \land \neg GF \psi$</td>
<td>$GF \psi$</td>
</tr>
</tbody>
</table>
Construction for LTL($F, G$)

Overview

- A state is a **guess** about how often each $F, G$ subformula holds.
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.
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.
## Construction for LTL\((F, G)\)

### Evaluation

<table>
<thead>
<tr>
<th>(G\psi)</th>
<th>(A)</th>
<th>(B)</th>
<th>(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(G\psi)</td>
<td>(G\psi)</td>
<td>(¬G\psi \land FG\psi)</td>
<td>(¬FG\psi)</td>
</tr>
<tr>
<td>(F\psi)</td>
<td>(¬F\psi)</td>
<td>(F\psi \land ¬GF\psi)</td>
<td>(GF\psi)</td>
</tr>
</tbody>
</table>
### Construction for LTL($F, G$)

#### Evaluation

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G\psi$</td>
<td>$G\psi$</td>
<td>$\neg G\psi \land FG\psi$</td>
<td>$\neg FG\psi$</td>
</tr>
<tr>
<td>$F\psi$</td>
<td>$\neg F\psi$</td>
<td>$F\psi \land \neg GF\psi$</td>
<td>$GF\psi$</td>
</tr>
</tbody>
</table>

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

- truth of propositions obtained from input $\nu$
Construction for LTL($F, G$)

Evaluation

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G\psi$</td>
<td>$G\psi$</td>
<td>$\neg G\psi \land FG\psi$</td>
<td>$\neg FG\psi$</td>
</tr>
<tr>
<td>$F\psi$</td>
<td>$\neg F\psi$</td>
<td>$F\psi \land \neg GF\psi$</td>
<td>$GF\psi$</td>
</tr>
</tbody>
</table>

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

- truth of propositions obtained from input $\nu$
- boolean connectives evaluated using their semantics
Construction for $\text{LTL}(F, G)$

Evaluation

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G\psi$</td>
<td>$G\psi$</td>
<td>$\neg G\psi \land FG\psi$</td>
<td>$\neg FG\psi$</td>
</tr>
<tr>
<td>$F\psi$</td>
<td>$\neg F\psi$</td>
<td>$F\psi \land \neg GF\psi$</td>
<td>$GF\psi$</td>
</tr>
</tbody>
</table>

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

- truth of propositions obtained from input $\nu$
- boolean connectives evaluated using their semantics
- $[G\psi]_\nu^\pi$ is true iff $G\psi \in \pi_A$ and $[F\psi]_\nu^\pi$ is true iff $F\psi \not\in \pi_A$. 
Construction for LTL($F, G$)

<table>
<thead>
<tr>
<th>$G\psi$</th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G\psi$</td>
<td>$G\psi$</td>
<td>$\neg G\psi \land FG\psi$</td>
<td>$\neg FG\psi$</td>
</tr>
<tr>
<td>$F\psi$</td>
<td>$\neg F\psi$</td>
<td>$F\psi \land \neg GF\psi$</td>
<td>$GF\psi$</td>
</tr>
</tbody>
</table>

\[ \pi \xrightarrow{\nu} \rho \]
Construction for LTL\((F, G)\)

<table>
<thead>
<tr>
<th>(G\psi)</th>
<th>(A)</th>
<th>(B)</th>
<th>(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(G\psi)</td>
<td>(G\psi)</td>
<td>(\neg G\psi \land FG\psi)</td>
<td>(\neg FG\psi)</td>
</tr>
<tr>
<td>(F\psi)</td>
<td>(\neg F\psi)</td>
<td>(F\psi \land \neg GF\psi)</td>
<td>(GF\psi)</td>
</tr>
</tbody>
</table>

\[\pi \xrightarrow{\nu} \rho\]

\(G\psi \in \pi_A\)

\(G\psi \in \rho_A\)

Ensure \(\psi\) is true by evaluating it.
Construction for LTL($F$, $G$)

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G\psi$</td>
<td>$G\psi$</td>
<td>$\neg G\psi \land FG\psi$</td>
<td>$\neg FG\psi$</td>
</tr>
<tr>
<td>$F\psi$</td>
<td>$\neg F\psi$</td>
<td>$F\psi \land \neg GF\psi$</td>
<td>$GF\psi$</td>
</tr>
</tbody>
</table>

\[ \pi \xrightarrow{\nu} \rho \]

\[ G\psi \in \pi_B \]
\[ G\psi \in \rho_A \cup \rho_B \]

No need to check $G\psi$ is false
Construction for $\text{LTL}(F, G)$

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G\psi$</td>
<td>$G\psi$</td>
<td>$\neg G\psi \land FG\psi$</td>
<td>$\neg FG\psi$</td>
</tr>
<tr>
<td>$F\psi$</td>
<td>$\neg F\psi$</td>
<td>$F\psi \land \neg GF\psi$</td>
<td>$GF\psi$</td>
</tr>
</tbody>
</table>

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

- $G\psi \in \pi_C$
- $G\psi \in \rho_C$

No need to check $FG\psi$ is false
Construction for LTL($F, G$)

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G\psi$</td>
<td>$G\psi$</td>
<td>$\neg G\psi \land FG\psi$</td>
<td>$\neg FG\psi$</td>
</tr>
<tr>
<td>$F\psi$</td>
<td>$\neg F\psi$</td>
<td>$F\psi \land \neg GF\psi$</td>
<td>$GF\psi$</td>
</tr>
</tbody>
</table>

\[
\pi \xrightarrow{\nu} \rho
\]

- $F\psi \in \pi_A$
- $F\psi \in \rho_A$

**No need to check $F\psi$ is false**
Construction for $\text{LTL}(F,G)$

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G\psi$</td>
<td>$G\psi$</td>
<td>$\neg G\psi \land FG\psi$</td>
<td>$\neg FG\psi$</td>
</tr>
<tr>
<td>$F\psi$</td>
<td>$\neg F\psi$</td>
<td>$F\psi \land \neg GF\psi$</td>
<td>$GF\psi$</td>
</tr>
</tbody>
</table>

\[
\pi \xrightarrow{\nu} \rho
\]

$F\psi \in \pi_B$

$F\psi \in \rho_A \cup \rho_B$

If $F\psi$ moves to $A$ ensure $\psi$ is true
Construction for LTL($F$, $G$)

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G\psi$</td>
<td>$G\psi$</td>
<td>$\neg G\psi \land FG\psi$</td>
<td>$\neg FG\psi$</td>
</tr>
<tr>
<td>$F\psi$</td>
<td>$\neg F\psi$</td>
<td>$F\psi \land \neg GF\psi$</td>
<td>$GF\psi$</td>
</tr>
</tbody>
</table>

Check that $\psi$ holds infinitely often: use a counter!
An automaton state is a pair \((\pi, n)\)
Construction for LTL($F, G$)

States

An automaton state is a pair $(\pi, n)$

- $\pi$ is current guess for $\varphi$
Construction for LTL($F, G$)

States

An automaton state is a pair $(\pi, n)$

- $\pi$ is current guess for $\varphi$
- $n \in \{0, 1, \ldots k\}$ where $k$ is the number of $F$ formulas in $\pi_C$
Transitions should help check if the guess is sound

\[(\pi, m) \xrightarrow{\nu} (\rho, n)\]
Construction for LTL\((F, G)\)

Transitions

Transitions should help check if the guess is sound

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

- \(\pi_A \subseteq \rho_A\)
- \(\pi_B \supseteq \rho_B\)
- \(\pi_C = \rho_C\)

(component \(\pi_B\) is non-increasing)
Construction for LTL($F, G$)

Transitions

Transitions should help check if the guess is sound

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

- $\pi_A \subseteq \rho_A$  $\pi_B \supseteq \rho_B$  $\pi_C = \rho_C$  (component $\pi_B$ is non-increasing)
- for $G\psi \in \pi_A$, $[\psi]^{\pi}_{\nu}$ is true
Transitions should help check if the guess is sound

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

- \(\pi_A \subseteq \rho_A\), \(\pi_B \supseteq \rho_B\), \(\pi_C = \rho_C\)  
  (component \(\pi_B\) 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

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

- \(\pi_A \subseteq \rho_A\) \(\pi_B \supseteq \rho_B\) \(\pi_C = \rho_C\) (component \(\pi_B\) 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 $\text{LTL}(F, G)$

Transitions

Transitions should help check if the guess is sound

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

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

Acceptance Condition

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

- empty \(\pi_B\) ensure obligations are eventually met
Construction for LTL($F, G$)

Acceptance Condition

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

- empty $\pi_B$ 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 $\pi_B$ is empty.
- empty $\pi_B$ 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 $[\varphi]_\nu^{\pi}$ is true. Since initial guess is sound in an accepting run, the truth of $\varphi$ is ensured.
Construction for LTL($F, g$)

Limit Determinism

Limit determinism is ensured because

- Once $\pi_B$ becomes empty, the guess $\pi$ cannot change across transitions
- Counter is incremented deterministically
Consider $\varphi = G(a \lor Fb)$
Consider $\varphi = G(a \lor Fb)$
Consider $\varphi = G(a \lor Fb)$

$q_0 : \langle \varphi \mid Fb \mid - \rangle, 0$

true
Example

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

$q_0 : \langle \varphi \mid Fb \mid - \rangle, 0$

$q_1 : \langle \varphi, Fb \mid - \mid - \rangle, 0$

true

$b$

$true$
Example

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

$q_0 : \langle \varphi \mid Fb \mid - \rangle, 0$
$q_1 : \langle \varphi, Fb \mid -\mid - \rangle, 0$
$q_2 : \langle \varphi \mid -\mid Fb \rangle, 0$

true

$a$

$b$
Example

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

$q_0 : \langle \varphi \mid Fb \mid - \rangle, 0$

$q_1 : \langle \varphi, Fb \mid - - \rangle, 0$

$q_2 : \langle \varphi \mid - \mid Fb \rangle, 0$

$q_3 : \langle \varphi \mid - \mid Fb \rangle, 1$

true

$a$

$b$
Consider $\varphi = G(a \lor Fb)$
Example

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

$q_0 : \langle \varphi \mid Fb \mid - \rangle, \ 0$

$q_1 : \langle \varphi, Fb \mid - \mid - \rangle, \ 0$

$q_2 : \langle \varphi \mid - \mid Fb \rangle, \ 0$

$q_3 : \langle \varphi \mid - \mid Fb \rangle, \ 1$

$a$

$b$

$\neg b$

true
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 $\varphi$, is there a scheduler $S$ such that the set of executions of $A^S$ that satisfy $\varphi$ has probability $> 0$?
Model Checking Problem

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

- [Courcoubetis-Yannakakis 1995] The problem is 2EXPTIME-complete for LTL specs.
Model Checking Problem

Given an MDP $A$ and an LTL formula $\varphi$, is there a scheduler $S$ such that the set of executions of $A^S$ that satisfy $\varphi$ 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 $\varphi$. 
Model Checking Problem

Given and MDP $A$ and LTL formula $\varphi$, is there a scheduler $S$ such that the set of executions of $A^S$ that satisfy $\varphi$ 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 $\varphi$.
- [Kini-V. 2015] The problem is EXPTIME-complete for LTL \ GU specs
Ideas can be generalized to construct limit deterministic automata for full LTL
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
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/