

An Optimal Automata Approach to LTL Model Checking of Probabilistic Systems

Jean-Michel Couvreur, Nasser Saheb, Grégoire Sutre



LSV, Ecole Normale Sup. de Cachan, France

LaBRI, Bordeaux University, France



Outline

1. Introduction
2. Probabilistic systems
3. Linear Temporal Logic and ω -automata
4. LTL verification and evaluation problems
5. Our approach
6. Conclusion

Outline

1. Introduction
2. Probabilistic systems
3. Linear Temporal Logic and ω -automata
4. LTL verification and evaluation problems
5. Our approach
6. Conclusion

Motivations (practical)

- Need for **probabilistic** modeling:
 - randomized algorithms (distributed systems)
 - message loss in protocols, stochastic delays...
 - biological systems

Motivations (practical)

- Need for **probabilistic** modeling:
 - randomized algorithms (distributed systems)
 - message loss in protocols, stochastic delays...
 - biological systems

- Verification:
 - is a given property **almost surely** satisfied by the system?

- Focus on **linear time temporal properties**

Motivations (practical)

- Need for **probabilistic** modeling:
 - randomized algorithms (distributed systems)
 - message loss in protocols, stochastic delays...
 - biological systems

- Verification:
 - is a given property **almost surely** satisfied by the system?

- Focus on **linear time temporal properties**

- Evaluation:
 - with which probability is a given property satisfied by the system?

LTL verification on probabilistic systems

- The best known automata-based algorithm runs in double exponential time [Var85]
- There is a non-automata based algorithm running in single exponential time and polynomial space [CY95]
- Open problem [Var99]:

automata-based algorithm running in single exponential time?

LTL verification on probabilistic systems

- The best known automata-based algorithm runs in double exponential time [Var85]
- There is a non-automata based algorithm running in single exponential time and polynomial space [CY95]
- Open problem [Var99]:

automata-based algorithm running in single exponential time?

- Yes
- On-the-fly implementation

Outline

1. Introduction
2. Probabilistic systems
3. Linear Temporal Logic and ω -automata
4. LTL verification and evaluation problems
5. Our approach
6. Conclusion

Measuring sets of infinite words

- Σ : finite alphabet

Measuring sets of infinite words

- Σ : finite alphabet
- \mathcal{C}_Σ : set of all **basic cylindric sets** $w \cdot \Sigma^\omega$ with $w \in \Sigma^*$

Measuring sets of infinite words

- Σ : finite alphabet
- \mathcal{C}_Σ : set of all **basic cylindric sets** $w \cdot \Sigma^\omega$ with $w \in \Sigma^*$
- \mathcal{B}_Σ : **σ -algebra** (on Σ^ω) generated by \mathcal{C}_Σ
 - \mathcal{B}_Σ closed under complementation
 - \mathcal{B}_Σ closed under countable union (and intersection)

Measuring sets of infinite words

- Σ : finite alphabet
- \mathcal{C}_Σ : set of all **basic cylindric sets** $w \cdot \Sigma^\omega$ with $w \in \Sigma^*$
- \mathcal{B}_Σ : **σ -algebra** (on Σ^ω) generated by \mathcal{C}_Σ
 - \mathcal{B}_Σ closed under complementation
 - \mathcal{B}_Σ closed under countable union (and intersection)
- $(\Sigma^\omega, \mathcal{B}_\Sigma)$: considered **measurable space** (Σ will depend on the context)

Measuring sets of infinite words

- Σ : finite alphabet
- \mathcal{C}_Σ : set of all **basic cylindric sets** $w \cdot \Sigma^\omega$ with $w \in \Sigma^*$
- \mathcal{B}_Σ : **σ -algebra** (on Σ^ω) generated by \mathcal{C}_Σ
 - \mathcal{B}_Σ closed under complementation
 - \mathcal{B}_Σ closed under countable union (and intersection)
- $(\Sigma^\omega, \mathcal{B}_\Sigma)$: considered **measurable space** (Σ will depend on the context)
- probability measure defined on \mathcal{C}_Σ and **extended** to \mathcal{B}_Σ

Probabilistic systems

$$\blacksquare M = \langle S, T, \alpha, \beta, \lambda, P_0, P \rangle$$

1. $\langle S, T, \alpha, \beta, \lambda \rangle$ finite labeled graph over Σ

■ S : **states** and T : **transitions**

■ $\alpha : T \rightarrow V$ and $\beta : T \rightarrow V$: **source** and **target** mappings ($\bullet \cdot$ and $\cdot \bullet$)

■ $\lambda : T \rightarrow \Sigma$: **transition labeling**

Probabilistic systems

- $M = \langle S, T, \alpha, \beta, \lambda, P_0, P \rangle$
 1. $\langle S, T, \alpha, \beta, \lambda \rangle$ finite labeled graph over Σ
 - S : **states** and T : **transitions**
 - $\alpha : T \rightarrow V$ and $\beta : T \rightarrow V$: **source** and **target** mappings ($\bullet \cdot$ and $\cdot \bullet$)
 - $\lambda : T \rightarrow \Sigma$: **transition labeling**
 2. $P_0 : S \rightarrow [0, 1]$: **initial probability distribution** s.t. $\sum_{s \in S} P_0(s) = 1$

Probabilistic systems

- $M = \langle S, T, \alpha, \beta, \lambda, P_0, P \rangle$
 1. $\langle S, T, \alpha, \beta, \lambda \rangle$ finite labeled graph over Σ
 - S : **states** and T : **transitions**
 - $\alpha : T \rightarrow V$ and $\beta : T \rightarrow V$: **source** and **target** mappings (\bullet and \cdot)
 - $\lambda : T \rightarrow \Sigma$: **transition labeling**
 2. $P_0 : S \rightarrow [0, 1]$: **initial probability distribution** s.t. $\sum_{s \in S} P_0(s) = 1$
 3. $P : T \rightarrow]0, 1]$ is a **transition probability function** s.t. $\sum_{t \in s \bullet} P(t) = 1$

Probabilistic systems

- $M = \langle S, T, \alpha, \beta, \lambda, P_0, P \rangle$
 1. $\langle S, T, \alpha, \beta, \lambda \rangle$ finite labeled graph over Σ
 - S : **states** and T : **transitions**
 - $\alpha : T \rightarrow V$ and $\beta : T \rightarrow V$: **source** and **target** mappings (\bullet and \cdot)
 - $\lambda : T \rightarrow \Sigma$: **transition labeling**
 2. $P_0 : S \rightarrow [0, 1]$: **initial probability distribution** s.t. $\sum_{s \in S} P_0(s) = 1$
 3. $P : T \rightarrow]0, 1]$ is a **transition probability function** s.t. $\sum_{t \in s \bullet} P(t) = 1$
- μ_M **probability measure** over $(T^\omega, \mathcal{B}_T)$ defined by $\mu_M(T^\omega) = 1$, and

$$\mu_M(t_0 t_1 \cdots t_n \cdot T^\omega) = \begin{cases} P_0(\bullet t_0) P(t_0) P(t_1) \cdots P(t_n) & \text{if } t_0 t_1 \cdots t_n \in \text{Path}^*(M) \\ 0 & \text{otherwise.} \end{cases}$$

Properties

- initial states: $S_0 = \{s \in S \mid P_0(s) \neq 0\}$
- $Path^\omega(M)$ is measurable and $\mu_M(Path^\omega(M)) = 1$

Properties

- initial states: $S_0 = \{s \in S \mid P_0(s) \neq 0\}$
- $Path^\omega(M)$ is measurable and $\mu_M(Path^\omega(M)) = 1$

Notation. $M[s] \hat{=} M$ where $P_0(s) = 1$ (i.e. s unique initial state)

Properties

- initial states: $S_0 = \{s \in S \mid P_0(s) \neq 0\}$
- $Path^\omega(M)$ is measurable and $\mu_M(Path^\omega(M)) = 1$

Notation. $M[s] \hat{=} M$ where $P_0(s) = 1$ (i.e. s unique initial state)

- $$\mu_M = \sum_{s \in S_0} P_0(s) \cdot \mu_{M[s]}$$

Properties

- initial states: $S_0 = \{s \in S \mid P_0(s) \neq 0\}$
- $Path^\omega(M)$ is measurable and $\mu_M(Path^\omega(M)) = 1$

Notation. $M[s] \hat{=} M$ where $P_0(s) = 1$ (i.e. s unique initial state)

- $\mu_M = \sum_{s \in S_0} P_0(s) \cdot \mu_{M[s]}$
- when L measurable, $\mu_{M[s]}(L) = \sum_{t \in s \bullet} P(t) \cdot \mu_{M[t \bullet]}(t^{-1}L)$

Properties

- initial states: $S_0 = \{s \in S \mid P_0(s) \neq 0\}$
- $Path^\omega(M)$ is measurable and $\mu_M(Path^\omega(M)) = 1$

Notation. $M[s] \hat{=} M$ where $P_0(s) = 1$ (i.e. s unique initial state)

- $\mu_M = \sum_{s \in S_0} P_0(s) \cdot \mu_{M[s]}$
- when L measurable, $\mu_{M[s]}(L) = \sum_{t \in s \bullet} P(t) \cdot \mu_{M[t \bullet]}(t^{-1}L)$

Proposition

- Let $Path_{max}^*$ denote the set of all finite paths ending in a maximal SCC.
We have $\mu_M(Path_{max}^* \cdot T^\omega) = 1$

Properties

- initial states: $S_0 = \{s \in S \mid P_0(s) \neq 0\}$
- $Path^\omega(M)$ is measurable and $\mu_M(Path^\omega(M)) = 1$

Notation. $M[s] \hat{=} M$ where $P_0(s) = 1$ (i.e. s unique initial state)

- $\mu_M = \sum_{s \in S_0} P_0(s) \cdot \mu_{M[s]}$
- when L measurable, $\mu_{M[s]}(L) = \sum_{t \in s \bullet} P(t) \cdot \mu_{M[t \bullet]}(t^{-1}L)$

Proposition

- Let $Path_{max}^*$ denote the set of all finite paths ending in a maximal SCC.
We have $\mu_M(Path_{max}^* \cdot T^\omega) = 1$
- Let ρ be a finite path contained in some maximal SCC C , and let $s \in C$.
We have $\mu_{M[s]}((T^* \cdot \rho)^\omega) = 1$.

Outline

1. Introduction
2. Probabilistic systems
3. Linear Temporal Logic and ω -automata
4. LTL verification and evaluation problems
5. Our approach
6. Conclusion

LTL

$f ::= a \mid f \vee f \mid \neg f \mid Xf \mid fUf$ (with $a \in \Sigma$)

LTL

$$f ::= a \mid f \vee f \mid \neg f \mid Xf \mid fUf \quad (\text{with } a \in \Sigma)$$

- models are in Σ^ω
- \vee and \neg interpreted as usual

LTL

$$f ::= a \mid f \vee f \mid \neg f \mid Xf \mid fUf \quad (\text{with } a \in \Sigma)$$

- models are in Σ^ω
- \vee and \neg interpreted as usual
- $abca \cdots \models a$ but $bbca \cdots \not\models a$

LTL

$f ::= a \mid f \vee f \mid \neg f \mid Xf \mid fUf$ (with $a \in \Sigma$)

- models are in Σ^ω
- \vee and \neg interpreted as usual
- $abca \dots \models a$ but $bbca \dots \not\models a$
- $abca \dots \models Xb$ but $abca \dots \not\models Xc$

LTL

$$f ::= a \mid f \vee f \mid \neg f \mid Xf \mid fUf \quad (\text{with } a \in \Sigma)$$

- models are in Σ^ω
- \vee and \neg interpreted as usual
- $abca \dots \models a$ but $bbca \dots \not\models a$
- $abca \dots \models Xb$ but $abca \dots \not\models Xc$
- $aaaaaaaaabca \dots \models aUb$ but $aaaaaaaaabca \dots \not\models aUc$

LTL

$$f ::= a \mid f \vee f \mid \neg f \mid Xf \mid fUf \quad (\text{with } a \in \Sigma)$$

- models are in Σ^ω
- \vee and \neg interpreted as usual
- $abca \dots \models a$ but $bbca \dots \not\models a$
- $abca \dots \models Xb$ but $abca \dots \not\models Xc$
- $aaaaaaaaabca \dots \models aUb$ but $aaaaaaaaabca \dots \not\models aUc$
- $L(f)$: set of words $w \in \Sigma^\omega$ such that $w \models f$

ω -automata

- $A = \langle Q, T, \alpha, \beta, \lambda, Q_0, Acc \rangle$
 1. $\langle Q, T, \alpha, \beta, \lambda \rangle$ finite labeled graph over Σ
 2. $Q_0 \subseteq Q$: initial locations
 3. $acc \subseteq 2^T$: acceptance condition

ω -automata

- $A = \langle Q, T, \alpha, \beta, \lambda, Q_0, Acc \rangle$
 1. $\langle Q, T, \alpha, \beta, \lambda \rangle$ finite labeled graph over Σ
 2. $Q_0 \subseteq Q$: **initial locations**
 3. $acc \subseteq 2^T$: **acceptance condition**

- a run $\rho \in Path^\omega(A)$ is **accepting** if $\{t \mid \rho \in (T^* \cdot t)^\omega\} \in Acc$

ω -automata

- $A = \langle Q, T, \alpha, \beta, \lambda, Q_0, Acc \rangle$
 1. $\langle Q, T, \alpha, \beta, \lambda \rangle$ finite labeled graph over Σ
 2. $Q_0 \subseteq Q$: **initial locations**
 3. $acc \subseteq 2^T$: **acceptance condition**
- a run $\rho \in Path^\omega(A)$ is **accepting** if $\{t \mid \rho \in (T^* \cdot t)^\omega\} \in Acc$
- a word w is **accepted** if $w = \lambda(\rho)$ for some accepting run ρ

ω -automata

- $A = \langle Q, T, \alpha, \beta, \lambda, Q_0, Acc \rangle$
 1. $\langle Q, T, \alpha, \beta, \lambda \rangle$ finite labeled graph over Σ
 2. $Q_0 \subseteq Q$: **initial locations**
 3. $acc \subseteq 2^T$: **acceptance condition**
- a run $\rho \in Path^\omega(A)$ is **accepting** if $\{t \mid \rho \in (T^* \cdot t)^\omega\} \in Acc$
- a word w is **accepted** if $w = \lambda(\rho)$ for some accepting run ρ
- $L(A)$: set of accepted words

ω -automata

- $A = \langle Q, T, \alpha, \beta, \lambda, Q_0, Acc \rangle$
 1. $\langle Q, T, \alpha, \beta, \lambda \rangle$ finite labeled graph over Σ
 2. $Q_0 \subseteq Q$: **initial locations**
 3. $acc \subseteq 2^T$: **acceptance condition**
- a run $\rho \in Path^\omega(A)$ is **accepting** if $\{t \mid \rho \in (T^* \cdot t)^\omega\} \in Acc$
- a word w is **accepted** if $w = \lambda(\rho)$ for some accepting run ρ
- $L(A)$: set of accepted words

Theorem [Var85]

For any ω -automaton A , $L(A)$ is measurable.

From LTL to ω -automata

Theorem [VW94]

Given an LTL formula f , one can build a Büchi ω -automaton A_f , with at most $2^{O(|f|)}$ locations, such that $L(f) = L(A_f)$.

Outline

1. Introduction
2. Probabilistic systems
3. Linear Temporal Logic and ω -automata
4. LTL verification and evaluation problems
5. Our approach
6. Conclusion

Verification and evaluation problems

- LTL probabilistic verification problem:

Given M and f , does M almost surely satisfy f ?

Verification and evaluation problems

- LTL probabilistic verification problem:

Given M and f , does M almost surely satisfy f ?

- “ $M \models f$ almost surely” $\hat{=}$

$$\mu_M(\text{Path}^\omega(M) \cap \lambda^{-1}(L(f))) = 1$$

Verification and evaluation problems

- LTL probabilistic verification problem:

Given M and f , does M almost surely satisfy f ?

- “ $M \models f$ almost surely” $\hat{=}$

$$\mu_M(\text{Path}^\omega(M) \cap \lambda^{-1}(L(f))) = 1$$

- “ $M \models f$ with positive probability” $\hat{=}$

$$\mu_M(\text{Path}^\omega(M) \cap \lambda^{-1}(L(f))) > 0$$

Verification and evaluation problems

- LTL probabilistic verification problem:

Given M and f , does M almost surely satisfy f ?

- “ $M \models f$ almost surely” $\hat{=}$

$$\mu_M(\text{Path}^\omega(M) \cap \lambda^{-1}(L(f))) = 1$$

- “ $M \models f$ with positive probability” $\hat{=}$

$$\mu_M(\text{Path}^\omega(M) \cap \lambda^{-1}(L(f))) > 0$$

- LTL probabilistic verification problem is PSPACE-complete [CY95]

Verification and evaluation problems

- LTL probabilistic verification problem:

Given M and f , does M almost surely satisfy f ?

- “ $M \models f$ almost surely” $\hat{=}$

$$\mu_M(\text{Path}^\omega(M) \cap \lambda^{-1}(L(f))) = 1$$

- “ $M \models f$ with positive probability” $\hat{=}$

$$\mu_M(\text{Path}^\omega(M) \cap \lambda^{-1}(L(f))) > 0$$

- LTL probabilistic verification problem is PSPACE-complete [CY95]
- LTL probabilistic evaluation problem:

Given M and f , compute $\mu_M(\text{Path}^\omega(M) \cap \lambda^{-1}(L(f)))$

Existing automata-based approach [Var85]

To check whether $M \models f$ with positive probability:

- compute a Büchi automaton A_f with $L(A_f) = L(f)$

$$|A_f| \text{ in } 2^{O(|f|)}$$

Existing automata-based approach [Var85]

To check whether $M \models f$ with positive probability:

- compute a Büchi automaton A_f with $L(A_f) = L(f)$

$$|A_f| \text{ in } 2^{O(|f|)}$$

- compute a **deterministic** Street automaton A'_f with $L(A'_f) = L(f)$

$$|A'_f| \text{ in } 2^{2^{O(|f|)}}$$

Existing automata-based approach [Var85]

To check whether $M \models f$ with positive probability:

- compute a Büchi automaton A_f with $L(A_f) = L(f)$

$$|A_f| \text{ in } 2^{O(|f|)}$$

- compute a **deterministic** Street automaton A'_f with $L(A'_f) = L(f)$

$$|A'_f| \text{ in } 2^{2^{O(|f|)}}$$

- compute the **probabilistic system** $M \otimes A'_f$ (synchronized product)

$$|M \otimes A'_f| \text{ in } O(|M|) \cdot 2^{2^{O(|f|)}}$$

Existing automata-based approach [Var85]

To check whether $M \models f$ with positive probability:

- compute a Büchi automaton A_f with $L(A_f) = L(f)$

$$|A_f| \text{ in } 2^{O(|f|)}$$

- compute a **deterministic** Street automaton A'_f with $L(A'_f) = L(f)$

$$|A'_f| \text{ in } 2^{2^{O(|f|)}}$$

- compute the **probabilistic system** $M \otimes A'_f$ (synchronized product)

$$|M \otimes A'_f| \text{ in } O(|M|) \cdot 2^{2^{O(|f|)}}$$

- check whether $M \otimes A'_f$ has an **accepted maximal SCC**

$$\text{in time } O(|M|) \cdot 2^{2^{O(|f|)}}$$

Outline

1. Introduction
2. Probabilistic systems
3. Linear Temporal Logic and ω -automata
4. LTL verification and evaluation problems
5. **Our approach**
6. Conclusion

Overview

We use a similar approach:

- translate f into a **non-deterministic** ω -automaton

$$|A_f| \text{ in } O(2^{|f|})$$

Overview

We use a similar approach:

- translate f into a **non-deterministic** ω -automaton

$$|A_f| \text{ in } O(2^{|f|})$$

- compute the **ω -automaton** $M \otimes A_f$ (synchronized product)

$$|M \otimes A_f| \text{ in } O(|M| \cdot 2^{|f|})$$

Overview

We use a similar approach:

- translate f into a **non-deterministic** ω -automaton

$$|A_f| \text{ in } O(2^{|f|})$$

- compute the **ω -automaton** $M \otimes A_f$ (synchronized product)

$$|M \otimes A_f| \text{ in } O(|M| \cdot 2^{|f|})$$

- look for a “suitable” SCC in $M \otimes A_f$

$$\text{in time } O(|M| \cdot 2^{|f|})$$

Overview

We use a similar approach:

- translate f into a **non-deterministic** ω -automaton

$$|A_f| \text{ in } O(2^{|f|})$$

- compute the **ω -automaton** $M \otimes A_f$ (synchronized product)

$$|M \otimes A_f| \text{ in } O(|M| \cdot 2^{|f|})$$

- look for a “suitable” SCC in $M \otimes A_f$

$$\text{in time } O(|M| \cdot 2^{|f|})$$

- based on **properties of the translation** from LTL to ω -automaton

Properties of ω -automata coming from LTL

- Optimized tableau based translation (slight variation of [Cou00])

Proposition

- *Given an LTL formula f , one can build a multi-Büchi ω -automaton A_f such that $L(f) = L(A_f)$, and whose size and computation time are in $O(|\Sigma|) \cdot 2^{O(|f|)}$.*
- *Moreover A_f is **unambiguous** and **separated** on each SCC.*

Properties of ω -automata coming from LTL

- Optimized tableau based translation (slight variation of [Cou00])

Proposition

- Given an LTL formula f , one can build a multi-Büchi ω -automaton A_f such that $L(f) = L(A_f)$, and whose size and computation time are in $O(|\Sigma|) \cdot 2^{O(|f|)}$.
- Moreover A_f is *unambiguous* and *separated* on each SCC.

where:

- A is *unambiguous* $\hat{=}$

$$t_1 \neq t_2 \wedge \bullet t_1 = \bullet t_2 \wedge \lambda(t_1) = \lambda(t_2) \Rightarrow L(A[t_1 \bullet]) \cap L(A[t_2 \bullet]) = \emptyset$$

Properties of ω -automata coming from LTL

- Optimized tableau based translation (slight variation of [Cou00])

Proposition

- Given an LTL formula f , one can build a multi-Büchi ω -automaton A_f such that $L(f) = L(A_f)$, and whose size and computation time are in $O(|\Sigma|) \cdot 2^{O(|f|)}$.
- Moreover A_f is *unambiguous* and *separated* on each SCC.

where:

- A is *unambiguous* $\hat{=}$

$$t_1 \neq t_2 \wedge \bullet t_1 = \bullet t_2 \wedge \lambda(t_1) = \lambda(t_2) \Rightarrow L(A[t_1 \bullet]) \cap L(A[t_2 \bullet]) = \emptyset$$

- A is *separated* $\hat{=}$ $q_1 \neq q_2 \Rightarrow L(A[q_1]) \cap L(A[q_2]) = \emptyset$

Synchronized product of M and A_f

$M \otimes A = \langle S \times Q, T_{\otimes}, \alpha_{\otimes}, \beta_{\otimes}, \lambda_{\otimes}, S_0 \times Q_0, Acc_{\otimes} \rangle$ where:

- $T_{\otimes} = \{(t_M, t_A) \in T_M \times T_A \mid \lambda_M(t_M) = \lambda_A(t_A)\}$
- $\bullet(t_M, t_A) = (\bullet t_M, \bullet t_A)$ and $(t_M, t_A)^\bullet = (t_M^\bullet, t_A^\bullet)$
- λ_{\otimes} is the projection from T_{\otimes} to T_M
- $U \in Acc_{\otimes}$ iff the projection of U on A is in Acc

Synchronized product of M and A_f

$M \otimes A = \langle S \times Q, T_{\otimes}, \alpha_{\otimes}, \beta_{\otimes}, \lambda_{\otimes}, S_0 \times Q_0, Acc_{\otimes} \rangle$ where:

- $T_{\otimes} = \{(t_M, t_A) \in T_M \times T_A \mid \lambda_M(t_M) = \lambda_A(t_A)\}$
- $\bullet(t_M, t_A) = (\bullet t_M, \bullet t_A)$ and $(t_M, t_A)^\bullet = (t_M^\bullet, t_A^\bullet)$
- λ_{\otimes} is the projection from T_{\otimes} to T_M
- $U \in Acc_{\otimes}$ iff the projection of U on A is in Acc

Proposition

- $L(M \otimes A) = Path^\omega(M) \cap \lambda_M^{-1}(L(A))$

Synchronized product of M and A_f

$M \otimes A = \langle S \times Q, T_{\otimes}, \alpha_{\otimes}, \beta_{\otimes}, \lambda_{\otimes}, S_0 \times Q_0, Acc_{\otimes} \rangle$ where:

- $T_{\otimes} = \{(t_M, t_A) \in T_M \times T_A \mid \lambda_M(t_M) = \lambda_A(t_A)\}$
- $\bullet(t_M, t_A) = (\bullet t_M, \bullet t_A)$ and $(t_M, t_A)^{\bullet} = (t_M^{\bullet}, t_A^{\bullet})$
- λ_{\otimes} is the projection from T_{\otimes} to T_M
- $U \in Acc_{\otimes}$ iff the projection of U on A is in Acc

Proposition

- $L(M \otimes A) = Path^{\omega}(M) \cap \lambda_M^{-1}(L(A))$
- $M \models f$ with positive probability iff $\mu_{M[s]}(L(M \otimes A[s, q])) > 0$ from an initial location $(s, q) \in S_0 \times Q_0$

Synchronized product of M and A_f (cont'd)

- $L(s, q) \hat{=} L(M \otimes A[s, q])$

$$L(s, q) = \bigcup_{(t_M, t_A) \in (s, q)^\bullet} t_M \cdot L(t_M^\bullet, t_A^\bullet)$$

- $V(s, q) \hat{=} \mu_{M[s]}(L(s, q))$

$V(s, q) > 0$ iff $V(s', q') > 0$ for some (s', q') reachable from (s, q)

Synchronized product of M and A_f (cont'd)

- $L(s, q) \hat{=} L(M \otimes A[s, q])$

$$L(s, q) = \bigcup_{(t_M, t_A) \in (s, q)^\bullet} t_M \cdot L(t_M^\bullet, t_A^\bullet)$$

- $V(s, q) \hat{=} \mu_{M[s]}(L(s, q))$

$V(s, q) > 0$ iff $V(s', q') > 0$ for some (s', q') reachable from (s, q)

- An SCC C of $M \otimes A$ is called:

- **null** if $V(s, q) = 0$ for all $(s, q) \in C$

- **persistent** if C is an SCC which is maximal among the non null SCCs,

- **transient** otherwise.

Synchronized product of M and A_f (cont'd)

- $L(s, q) \hat{=} L(M \otimes A[s, q])$

$$L(s, q) = \bigcup_{(t_M, t_A) \in (s, q)^\bullet} t_M \cdot L(t_M^\bullet, t_A^\bullet)$$

- $V(s, q) \hat{=} \mu_{M[s]}(L(s, q))$

$V(s, q) > 0$ iff $V(s', q') > 0$ for some (s', q') reachable from (s, q)

- An SCC C of $M \otimes A$ is called:

- **null** if $V(s, q) = 0$ for all $(s, q) \in C$

- **persistent** if C is an SCC which is maximal among the non null SCCs,

- **transient** otherwise.

- Goal: check the existence of a reachable non null SCC

Local notions on SCCs

- $(M \otimes A)|_C \hat{=} \text{“restriction of } M \otimes A \text{ to } C$
- $L_C(s, q) = L((M \otimes A)|_C[s, q])$
- $V_C(s, q) = \mu_{M[s]}(L_C(s, q))$

Local notions on SCCs

- $(M \otimes A)|_C \hat{=} \text{“restriction of } M \otimes A \text{ to } C$
- $L_C(s, q) = L((M \otimes A)|_C[s, q])$
- $V_C(s, q) = \mu_{M[s]}(L_C(s, q))$
- C is **locally positive** if $V_C(s, q) > 0$ for all $(s, q) \in C$

Local notions on SCCs

- $(M \otimes A)|_C \hat{=} \text{“restriction of } M \otimes A \text{ to } C$
- $L_C(s, q) = L((M \otimes A)|_C[s, q])$
- $V_C(s, q) = \mu_{M[s]}(L_C(s, q))$
- C is **locally positive** if $V_C(s, q) > 0$ for all $(s, q) \in C$

persistent \Rightarrow locally positive \Rightarrow non null

Local notions on SCCs

- $(M \otimes A)|_C \hat{=} \text{“restriction of } M \otimes A \text{ to } C$
- $L_C(s, q) = L((M \otimes A)|_C[s, q])$
- $V_C(s, q) = \mu_{M[s]}(L_C(s, q))$
- C is **locally positive** if $V_C(s, q) > 0$ for all $(s, q) \in C$

persistent \Rightarrow locally positive \Rightarrow non null

- $M \models f$ with positive probability iff there is a locally positive SCC reachable from an initial location (s, q) in $S_0 \times Q_0$

Characterisation of locally positive SCCs

- C is **accepted** if its set of transitions is in Acc_{\otimes}
- C is **complete** if every finite path of M starting in C is contained in C

Characterisation of locally positive SCCs

- C is **accepted** if its set of transitions is in Acc_{\otimes}
- C is **complete** if every finite path of M starting in C is contained in C

Proposition

If A is multi-Büchi or unambiguous, then

$$\text{locally positive} \iff \text{accepted} \wedge \text{complete}$$

Characterisation of locally positive SCCs

- C is **accepted** if its set of transitions is in Acc_{\otimes}
- C is **complete** if every finite path of M starting in C is contained in C

Proposition

If A is multi-Büchi or unambiguous, then

$$\text{locally positive} \iff \text{accepted} \wedge \text{complete}$$

Proposition

- *when A is multi-Büchi, acceptance checking is in $O(|Acc| \cdot |C|)$*
- *when A is unambiguous and separated on each SCC, completeness checking is in $O(|C|)$*

Main result

Theorem

Given an LTL formula f , checking whether $M \models f$ with positive probability can be done in $O(|M| \cdot |f| \cdot 2^{|f|})$.

Evaluation

Proposition

- If A is *unambiguous* then

$$V(s, q) = \sum_{(t_M, t_A) \in (s, q)^\bullet} P(t_M) \cdot V(t_M^\bullet, t_A^\bullet)$$

Evaluation

Proposition

- If A is *unambiguous* then

$$V(s, q) = \sum_{(t_M, t_A) \in (s, q)^\bullet} P(t_M) \cdot V(t_M^\bullet, t_A^\bullet)$$

- Moreover, for every persistent SCC C ,
 - if C is *deterministic* then $V(s, q) = 1$ for all $s, q \in C$
 - if C is *separated* then $\sum_{q: (s, q) \in C} V(s, q) = 1$ for all $s \in C$

Evaluation

Proposition

- If A is *unambiguous* then

$$V(s, q) = \sum_{(t_M, t_A) \in (s, q)^\bullet} P(t_M) \cdot V(t_M^\bullet, t_A^\bullet)$$

- Moreover, for every persistent SCC C ,
 - if C is *deterministic* then $V(s, q) = 1$ for all $s, q \in C$
 - if C is *separated* then $\sum_{q: (s, q) \in C} V(s, q) = 1$ for all $s \in C$
- Equation system decomposed and solved for each component

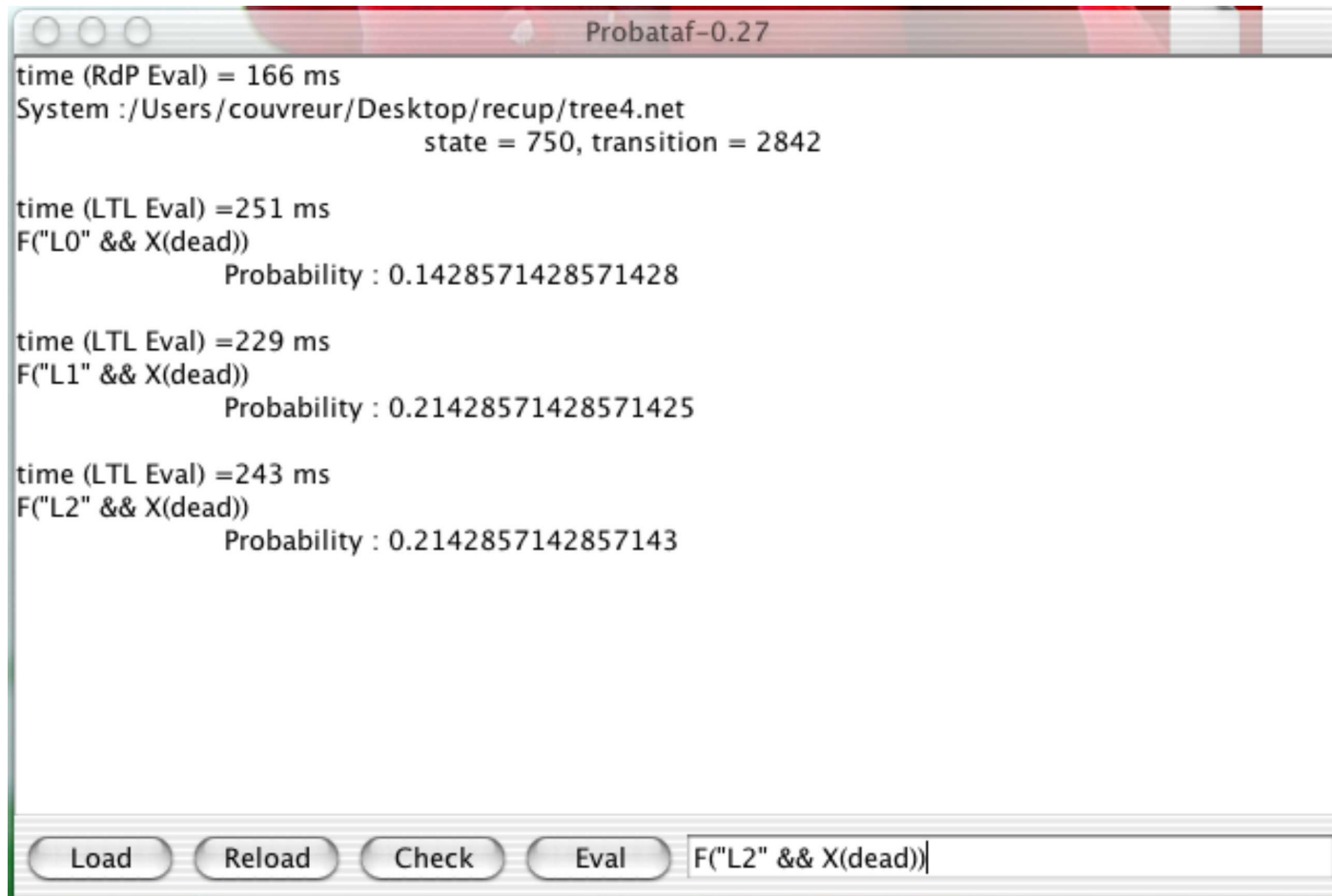
Outline

1. Introduction
2. Probabilistic systems
3. Linear Temporal Logic and ω -automata
4. LTL verification and evaluation problems
5. Our approach
6. **Conclusion**

Experimentation: the ProbaTaf tool

- Probabilistic systems described by bounded Petri nets
- LTL formulas on the Petri net: transitions, markings and “dead”
- **explicit description** of the probabilistic system
- **symbolic BDD-based** representation of ω -automata
- **on-the-fly** verification algorithm [Cou99]
- simple **Gauss** elimination algorithm for evaluation
- application to several examples:
 - biased dice game [KY76]
 - randomized election algorithm [MSZ03]

Experimentation: the ProbaTaf tool



The screenshot shows a window titled "Probatatf-0.27" with the following output:

```
time (RdP Eval) = 166 ms
System : /Users/couvreur/Desktop/recup/tree4.net
           state = 750, transition = 2842

time (LTL Eval) = 251 ms
F("L0" && X(dead))
           Probability : 0.1428571428571428

time (LTL Eval) = 229 ms
F("L1" && X(dead))
           Probability : 0.21428571428571425

time (LTL Eval) = 243 ms
F("L2" && X(dead))
           Probability : 0.2142857142857143
```

At the bottom, there are four buttons: "Load", "Reload", "Check", and "Eval". To the right of these buttons is a text input field containing the LTL formula: `F("L2" && X(dead))`.

Conclusion and perspectives

- Optimal automata-based approach for LTL verification
 - Allows evaluation
- Based on properties of ω -automata: **separation** and **unambiguity**
- Java implementation of the method

Conclusion and perspectives

- Optimal automata-based approach for LTL verification
 - Allows evaluation
- Based on properties of ω -automata: **separation** and **unambiguity**
- Java implementation of the method

Future work

- **precision** of the solver
- **infinite-state** probabilistic systems
- **stochastic** systems

References

- [Cou99] Jean-Michel Couvreur. On-the-fly verification of linear temporal logic. In *FM'99—Formal Methods, Volume 1*, volume 1708 of *Lecture Notes in Computer Science*, pages 253–271. Springer, 1999.
- [Cou00] Jean-Michel Couvreur. Un point de vue symbolique sur la logique temporelle linéaire. In *Actes du Colloque LaCIM 2000*, volume 27 of *Publications du LaCIM*, pages 131–140. Université du Québec à Montréal, August 2000.
- [CY95] Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, July 1995.
- [KY76] Knuth and Yao. The complexity of nonuniform random number generation. In *Algorithms and Complexity: New Directions and Recent Results*, Ed. J. F. Traub. Academic Press, 1976.
- [MSZ03] Yves Métivier, Nasser Saheb, and Akka Zemmari. A uniform randomized election in trees. In *SIROCCO 10*, volume 17 of *Proceedings in Informatics*, pages 259–274. Carleton Scientific, 2003.
- [Var85] M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proc. 26th IEEE Symp. Foundations of Computer Science (FOCS'85), Portland, OR, USA, Oct. 1985*, pages 327–338, 1985.
- [Var99] M. Y. Vardi. Probabilistic linear-time model checking: An overview of the automata-theoretic approach. In *Proc. 5th Int. AMAST Workshop Formal Methods for Real-Time and Probabilistic Systems (ARTS'99), Bamberg, Germany, May 1999*, volume 1601 of *Lecture Notes in Computer Science*, pages 265–276. Springer, 1999.
- [VW94] Moshe Y. Vardi and Pierre Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.