Department of Computer Science
University of Oxford

Time-Bounded Verification of CTMCs Against Real-Time Specifications

Marco Diciolla

30/09/2011

ICCSW Phd Student Conference
Imperial College

Joint work with : Taolue Chen, Marta Kwiatkowska and Alexandru Mereacre

Overview

- Introduction
- Motivations
- Continuous-Time Markov Chains (CTMCs)
- Metric Temporal Logic (MTL)
- Time-Bounded Verification of MTL against CTMCs
- Conclusions

Introduction: Model Checking

Why Model Checking? ⇨ Errors might be costly ...

Toyota Camry (2010): Software glitch found in anti-lock braking system.
Loss: Over 185.000 cars recalled.

Mars Polar Lander (1999): The spacecraft failed to phone home after its attempted landing at the planet's south pole on December 3, 1999. The crash was caused by a flaw in the code which generated spurious signals when the craft's legs were deployed during descent.
Loss: $112 million.

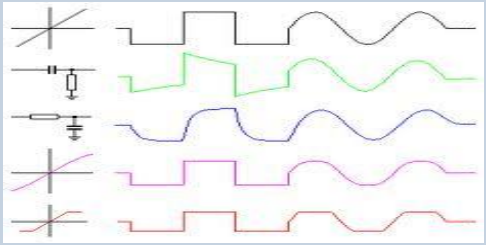
Introduction: Probabilistic Model Checking

Can I check that my airbag will deploy on time?

Complex system ...

Stochasticity ...

Interaction



Time constraints ...

UNIVERSITY OF OXFORD

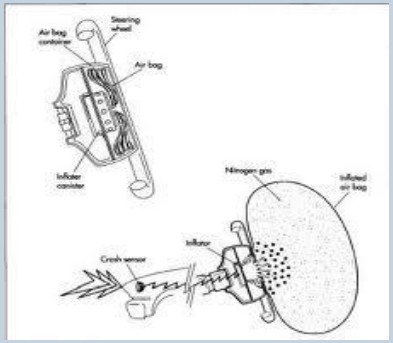
Marco Diciolla

Introduction:
Probabilistic Model Checking

What can we do?

Model of the System :
Airbag

Property :
Airbag will always deploy on time



Model Checker

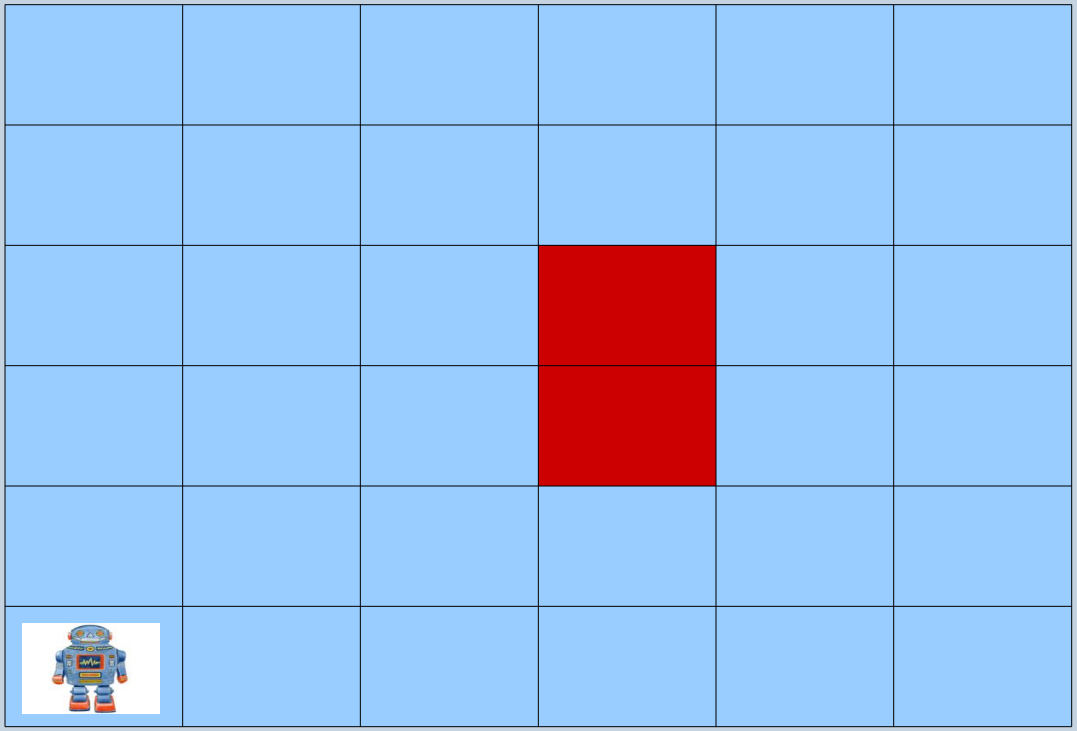
YES,NO

UNIVERSITY OF OXFORD

Marco Diciolla

Motivations

Can the robot R reach the red squares in less than 2 seconds and stay there forever?



UNIVERSITY OF OXFORD

Marco Diciolla

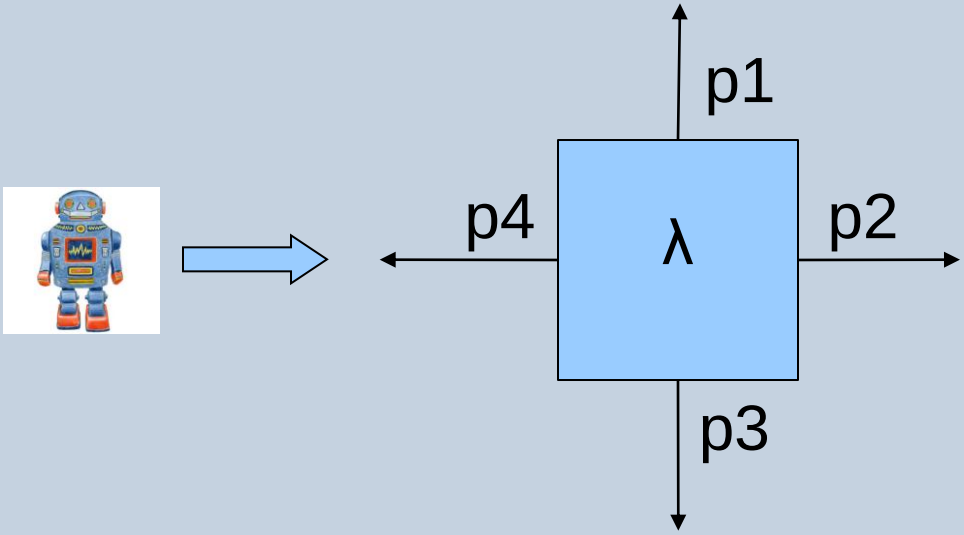
Specifications

Metric Temporal Logic Formula

$$\Diamond_{[0,2]}\Box(Red\_Zone)$$

Model

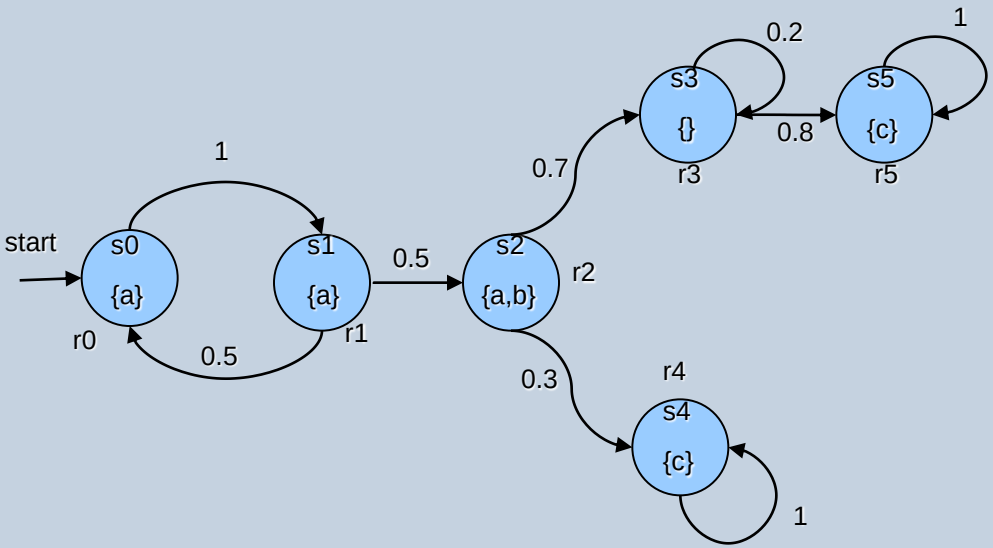
Single cell of the grid



- p1 is the probability of moving up;
- p2 is the probability of moving right;
- p3 is the probability of moving down;
- p4 is the probability of moving left; and
- 1/λ is the average residence time.

Continuous-Time Markov Chain

C = (S,AP,L,α,P,E)



- S is a finite set of states;
- AP is a finite set of atomic propositions;
- L is the labeling function;
- α is the initial distribution;
- **P** is a stochastic matrix; and
- E is the exit rate function.

Model Checking Problem

$$\Pr(\text{Robot} \models \text{MTL formula}) = ?$$

MTL
Syntax

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 U_I \varphi_2,$$

with $p \in AP$, $I = [a, b]$ with $a, b \in \mathbb{N} \cup \{\infty\}$and φ_1, φ_2 are MTL formulas.

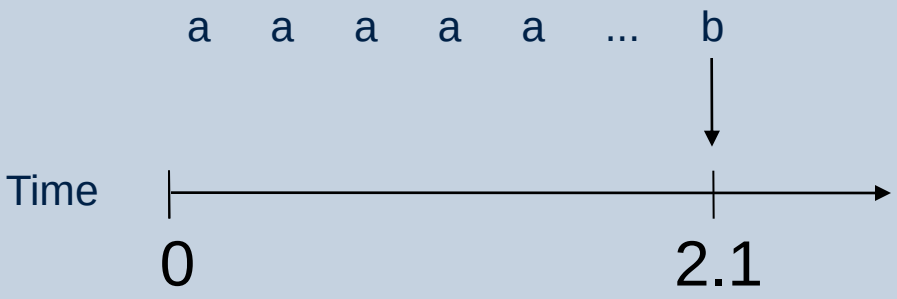
Time-Bounded Semantics

$$\begin{array}{lcl}
(\rho, t) \models p & \Leftrightarrow & p \in L(\rho@t) \wedge t \leq T \\
(\rho, t) \models \neg\varphi_1 & \Leftrightarrow & (\rho, t) \not\models \varphi_1 \\
(\rho, t) \models \varphi_1 \wedge \varphi_2 & \Leftrightarrow & (\rho, t) \models \varphi_1 \wedge (\rho, t) \models \varphi_2 \\
(\rho, t) \models \varphi_1 U_I \varphi_2 & \Leftrightarrow & \exists t'.\ t \leq t' \leq T \text{ s.t. } t' - t \in I \wedge (\rho, t') \models \varphi_2 \wedge \\
 & & \forall t''.\ t \leq t'' < t' \Rightarrow (\rho, t'') \models \varphi_1
\end{array}$$

where $p \in AP$ and φ_1, φ_2 are MTL formulas.

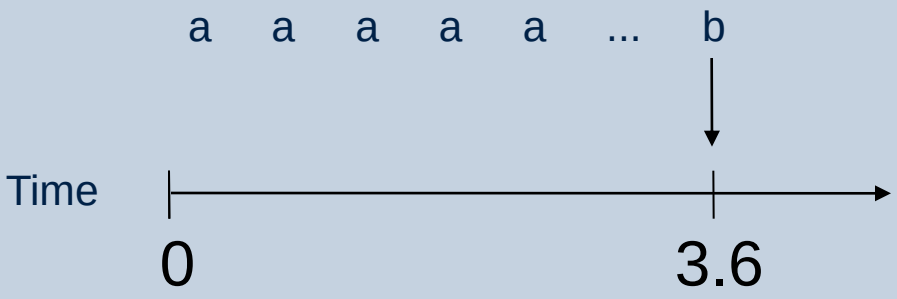
MTL : Example

$$\varphi = a\ U_{[2,4]}\ b \quad \Rightarrow \quad \text{True}$$



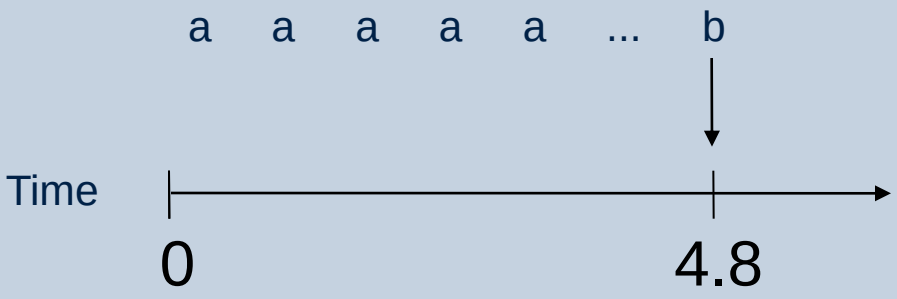
MTL : Example

$$\varphi = a\ U_{[2,4]}\ b \quad \Rightarrow \quad \text{True}$$



MTL : Example

$$\varphi = a\ U_{[2,4]}\ b \quad \Rightarrow \quad \text{False}$$



Time-Bounded Verification of MTL

The main algorithm is composed of 6 steps:

1) Bound the number of jumps N in the interval of time [0,T] in order to get the desired error

2) Transform the MTL formula φ in the untimed LTL version φ' (to reduce the complexity of the model checking algorithm)

3) Construct the NFA automaton $A_{\varphi'}$ out of φ' and build the product $CxA_{\varphi'}$

4) Search all the discrete paths σ in $(CxA_{\varphi'})_1$ of length at most N and for each of those, generate the set of linear inequalities S

5) Calculate the probability of σ under the constraints in S

6) Sum up all the probabilities

Time-Bounded Verification of MTL

The main algorithm is composed of 6 steps:

1) Bound the number of jumps N in the interval of time [0,T] in order to get the desired error

2) Transform the MTL formula φ in the untimed LTL version φ' (to reduce the complexity of the model checking algorithm)

3) Construct the NFA automaton $A_{\varphi'}$ out of φ' and build the product $CxA_{\varphi'}$

4) Search all the discrete paths σ in $(CxA_{\varphi'})_1$ of length at most N and for each of those, generate the set of linear inequalities S

5) Calculate the probability of σ under the constraints in S

6) Sum up all the probabilities

Time-Bounded Verification of MTL

The main algorithm is composed of 6 steps:

1) Bound the number of jumps N in the interval of time [0,T] in order to get the desired error

2) Transform the MTL formula φ in the untimed LTL version φ'
(to reduce the complexity of the model checking algorithm)

3) Construct the NFA automaton $A_{\varphi'}$ out of φ' and build the product $CxA_{\varphi'}$

4) Search all the discrete paths σ in $(CxA_{\varphi'})_1$ of length at most N and for each of those, generate the set of linear inequalities S

5) Calculate the probability of σ under the constraints in S

6) Sum up all the probabilities

Time-Bounded Verification of MTL

The main algorithm is composed of 6 steps:

1) Bound the number of jumps N in the interval of time [0,T] in order to get the desired error

2) Transform the MTL formula φ in the untimed LTL version φ'
(to reduce the complexity of the model checking algorithm)

3) Construct the NFA automaton $A_{\varphi'}$ out of φ' and build the product $CxA_{\varphi'}$

4) Search all the discrete paths σ in $(CxA_{\varphi'})_1$ of length at most N and for each of those, generate the set of linear inequalities S

5) Calculate the probability of σ under the constraints in S

6) Sum up all the probabilities

Time-Bounded Verification of MTL

The main algorithm is composed of 6 steps:

1) Bound the number of jumps N in the interval of time [0,T] in order to get the desired error

2) Transform the MTL formula φ in the untimed LTL version φ'
(to reduce the complexity of the model checking algorithm)

3) Construct the NFA automaton $A_{\varphi'}$ out of φ' and build the product $CxA_{\varphi'}$

4) Search all the discrete paths σ in $(CxA_{\varphi'})_1$ of length at most N and for each of those, generate the set of linear inequalities S

5) Calculate the probability of σ under the constraints in S

6) Sum up all the probabilities

Time-Bounded Verification of MTL

The main algorithm is composed of 6 steps:

1) Bound the number of jumps N in the interval of time [0,T] in order to get the desired error

2) Transform the MTL formula φ in the untimed LTL version φ'
(to reduce the complexity of the model checking algorithm)

3) Construct the NFA automaton $A_{\varphi'}$ out of φ' and build the product $CxA_{\varphi'}$

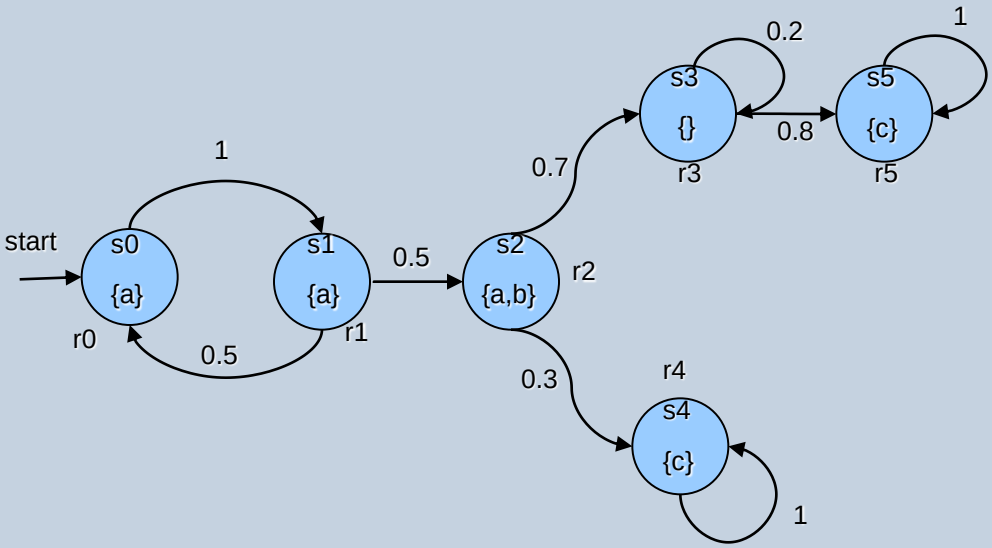
4) Search all the discrete paths σ in $(CxA_{\varphi'})_1$ of length at most N and for each of those, generate the set of linear inequalities S

5) Calculate the probability of σ under the constraints in S

6) Sum up all the probabilities

1) Bounding the number of jumps

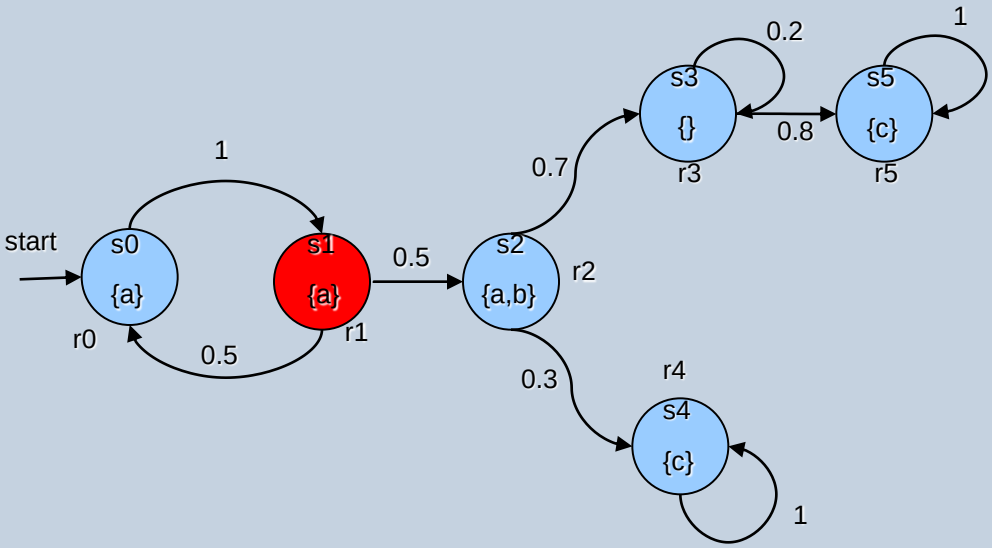
In a finite interval of time [0,T] there might be an arbitrary number of jumps in the CTMC.



How many jumps can we have in T?

1) Bounding the number of jumps

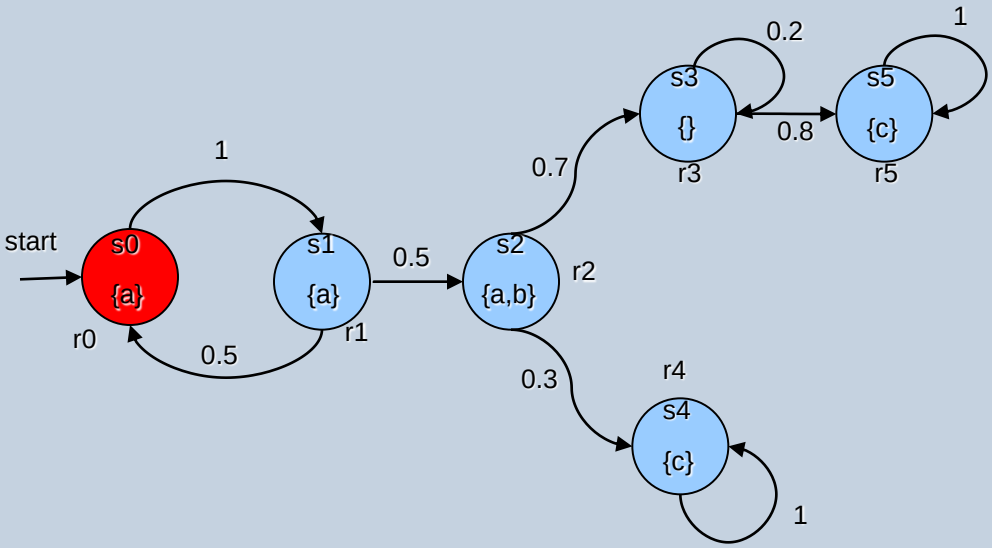
In a finite interval of time [0,T] there might be an arbitrary number of jumps in the CTMC.



How many jumps can we have in T?

1) Bounding the number of jumps

In a finite interval of time [0,T] there might be an arbitrary number of jumps in the CTMC.



How many jumps can we have in T?

1) Bounding the number of jumps

1.1) Pick a time bound T and the error bound ε;

1.2) Choose N such that :

$$N > \Lambda T e^2 + ln\left(\frac{1}{\epsilon}\right)$$

where Λ is the maximum exit rate of the CTMC.

Intuition: If N satisfies the inequality above, then the error produced by truncating the Poisson series is smaller than ε.

2) Transform the MTL formula φ in the untimed LTL formula φ'

$$
\begin{array}{lcl}
\varphi = p & \Rightarrow & \varphi' = p \\
\varphi = \neg p & \Rightarrow & \varphi' = \neg p \\
\varphi = \varphi_1 \vee \varphi_2 & \Rightarrow & \varphi' = \varphi_1' \vee \varphi_2' \\
\varphi = \varphi_1 \wedge \varphi_2 & \Rightarrow & \varphi' = \varphi_1' \wedge \varphi_2' \\
\varphi = \varphi_1 U_I \varphi_2 & \Rightarrow & \varphi' = \varphi_1' U \varphi_2' \\
\varphi = \Box_I \varphi_1 & \Rightarrow & \varphi' = \text{TRUE } U \ \varphi_1'
\end{array}
$$

where φ_1 and φ_2 are MTL formulas and φ_1' and φ_2' are LTL formulas.

Intuition: If a path does not satisfy the untimed LTL formula, then it does not satisfy the timed MTL formula.

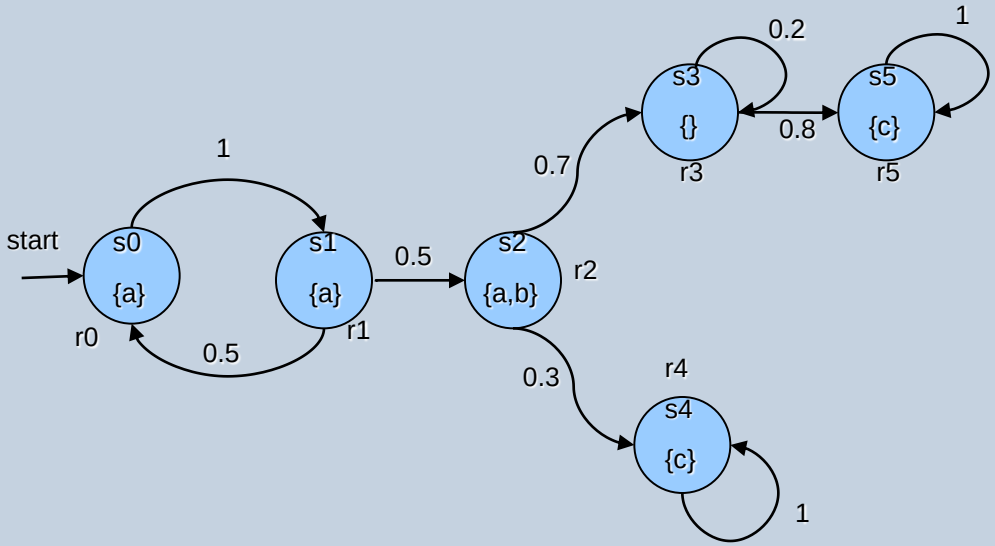
3) Construct $A_{\varphi'}$ and build $CxA_{\varphi'}$

- Construct $A_{\varphi'}$: Vardi-Wolper construction for LTL
- Build $CxA_{\varphi'}$:

$$C \otimes A_{\varphi'} = (Loc, l_0, Loc_{\mathbf{F}}, \rightsquigarrow) \text{ where :}$$

- $Loc = S \times Q$;
- $l_0 = \langle s_0, q_0 \rangle$;
- $Loc_{\mathbf{F}} = S \times F$;
- $\rightsquigarrow \subseteq Loc \times Loc$ s.t. $\quad \dfrac{P(s,s') > 0 \wedge q \xrightarrow{L(s)} q'}{\langle s, q \rangle \rightsquigarrow \langle s', q' \rangle}$.

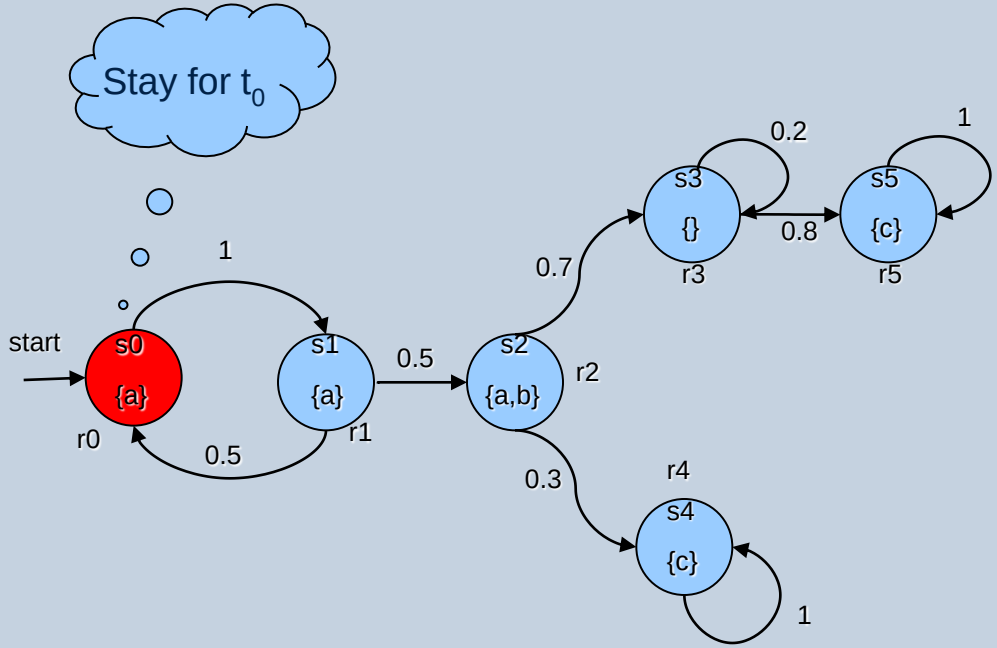
4) Set of linear inequalities S : Example



$$\sigma = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} s_3$$

$$\varphi = aU_{[1,2]}b$$

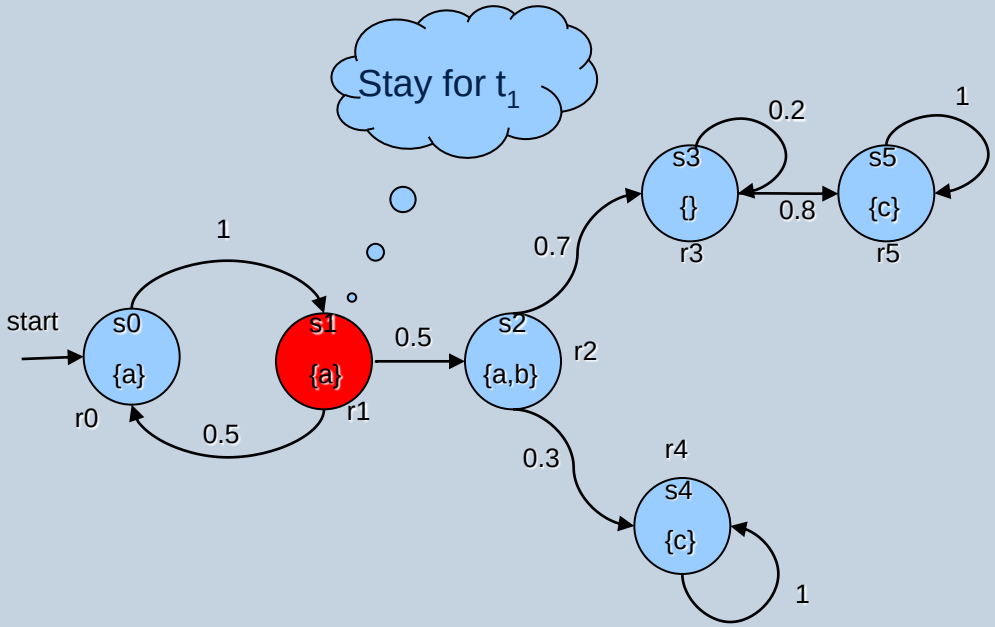
4) Set of linear inequalities S : Example



$$\sigma = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} s_3$$

$$\varphi = aU_{[1,2]}b$$

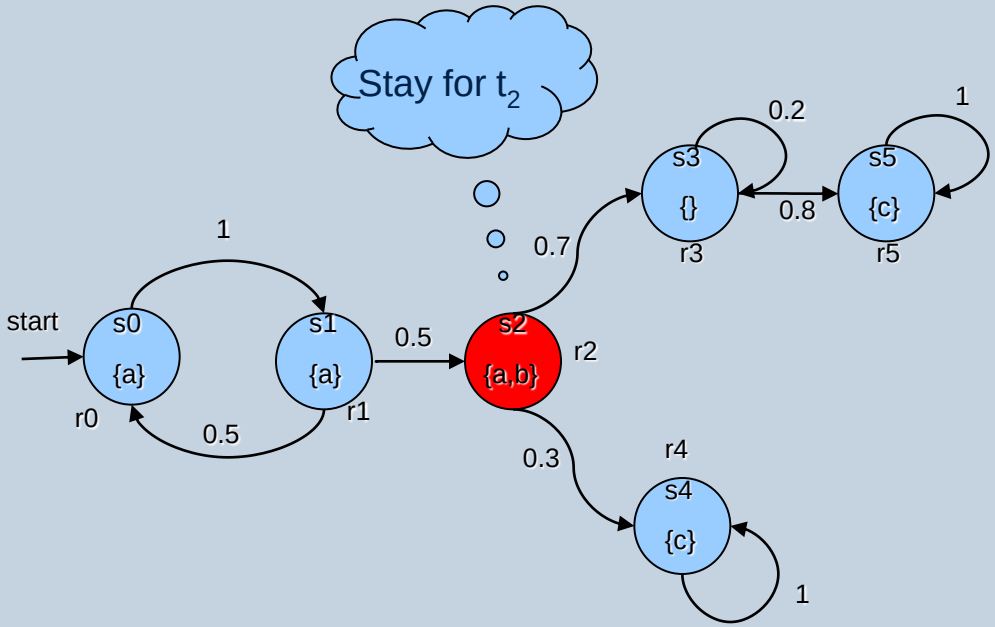
4) Set of linear inequalities S : Example



$$\sigma = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} s_3$$

$$\varphi = aU_{[1,2]}b$$

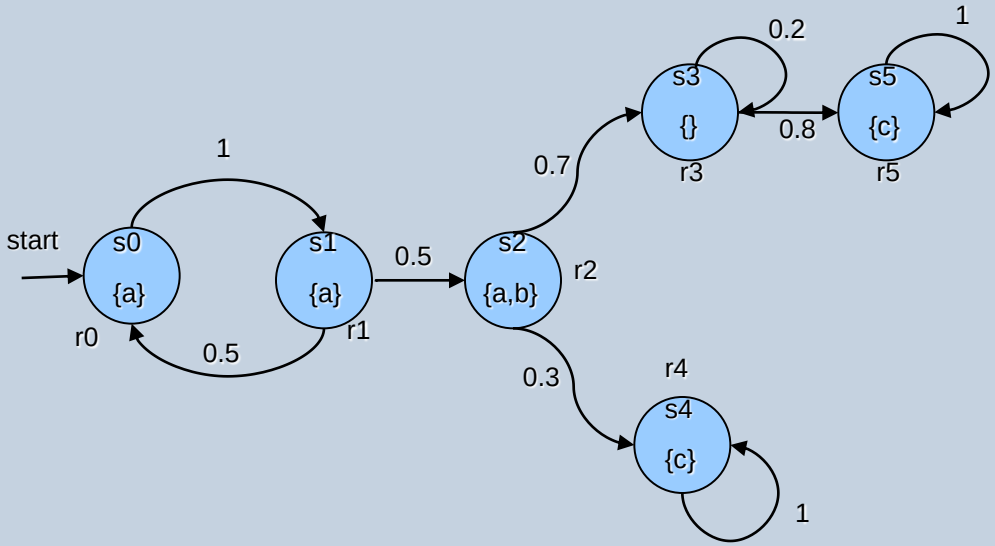
4) Set of linear inequalities S : Example



$$\sigma = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} s_3$$

$$\varphi = aU_{[1,2]}b$$

4) Set of linear inequalities S : Example



$$\sigma = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} s_3$$

$$\varphi = aU_{[1,2]}b$$

$$S = \begin{cases} t_0 + t_1 < 2 \\ t_0 + t_1 + t_2 \geq 1 \end{cases}.$$

5) Calculate the probability of σ under S (σ[S])

$$\Pr(\sigma[S]) = E(s_0)\mathbf{P}(s_0,s_1)\cdot E(s_1)\mathbf{P}(s_1,s_2)\cdot E(s_2)\mathbf{P}(s_2,s_3)\cdot \int\int\int_S e^{-(E(s_0)\tau_0\cdot E(s_1)\tau_1\cdot E(s_2)\tau_2)} d\tau_0 d\tau_1 d\tau_2$$

$$\sigma = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} s_3$$

$$S = \begin{cases} t_0 + t_1 < 2 \\ t_0 + t_1 + t_2 \geq 1 \end{cases}.$$

5) Calculate the probability of σ under S (σ[S])

$$\Pr(\sigma[S]) = \text{Volume Convex Polytope}$$

"A Laplace transform algorithm for computing the volume of convex polytope", J.B. Lasserre and E.S. Zeron. J. ACM, 48(6):1126-1140, 2001.

6) Sum up all the probabilities

$$\Pr(\text{Robot} \models \text{MTL formula})$$

$$=$$

$$\sum_{\substack{\sigma \text{ length} \\ \text{most } N}} \Pr(\sigma[S])$$

Summary

- First MTL verification algorithm against CTMC
- MTL verification is hard
- MTL verification problem reduced to constraint generation and volume computation

Future work:

- Detailed complexity analysis
- Implementation in PRISM and extension to time-unbounded verification

Summary

- First MTL verification algorithm against CTMC
- MTL verification is hard
- MTL verification problem reduced to constraint generation and volume computation

Future work:

- Detailed complexity analysis
- Implementation in PRISM and extension to time-unbounded verification

Summary

- First MTL verification algorithm against CTMC
- MTL verification is hard
- MTL verification problem reduced to constraint generation and volume computation

Future work:

- Detailed complexity analysis
- Implementation in PRISM and extension to time-unbounded verification

Summary

- First MTL verification algorithm against CTMC
- MTL verification is hard
- MTL verification problem reduced to constraint generation and volume computation

Future work:

- Detailed complexity analysis
- Implementation in PRISM and extension to time-unbounded verification

Summary

- First MTL verification algorithm against CTMC
- MTL verification is hard
- MTL verification problem reduced to constraint generation and volume computation

Future work:

- Detailed complexity analysis
- Implementation in PRISM and extension to time-unbounded verification

Thank you!!

Any Questions?