An Extension of the Inverse Method to Probabilistic Timed Automata

Étienne André, Laurent Fribourg

Laboratoire Spécification et Vérification
LSV, ENS de Cachan & CNRS, France

Jeremy Sproston

Dipartimento di Informatica
Università di Torino, Italy

Context: Verification of Timed Probabilistic Systems

- Verification of timed systems with stochastic behaviour
  - Need to express probabilities
  - Use of Probabilistic Timed Automata [Jen96, KNSS02]
- Need for adjusting some delays of the system
  - Use of parameters (unknown constants)
  - Definition of a zone of good behaviour for the parameters

Motivation: Model reduction

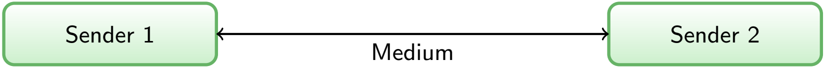
- Model checking Probabilistic Timed Automata
  - Use of the Prism model checker [HKNP06, wp]
  - Difficult to model-check systems with large constants
- Use rescaling of constants
  - Consider smaller values for all the constants of the system
  - Problem of discrete time
  - No formal justification for correctness
- Require a formal justification for rescaling of constants

Outline

Outline

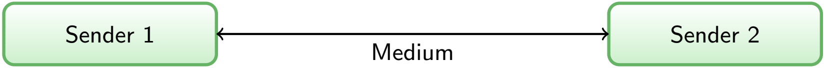
The CSMA/CD Protocol (1/2)

- Protocol of communication between 2 stations through 1 medium
  - Carrier Sense Multiple Access with Collision Detection [CSM02, KNSW07]



The CSMA/CD Protocol (1/2)

- Protocol of communication between 2 stations through 1 medium
  - Carrier Sense Multiple Access with Collision Detection [CSM02, KNSW07]



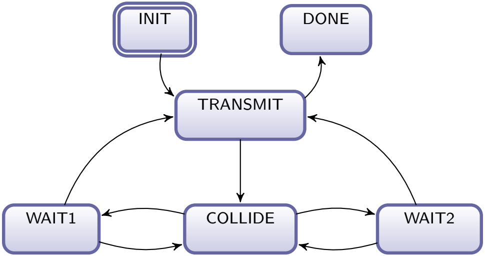
- Overall principle: Sender 1 tries to communicate
  1. Sender 1 listens to the medium
  2. If the medium is free, Sender 1 starts to communicate (duration λ)
  3. Since there is a non-null delay for a signal to go through the medium (duration σ), Sender 2 may have started to communicate in the meanwhile, which leads to a collision
  4. Both senders then wait a random number of time slots (duration *slot*) before trying again

Timed Parameters of the System

- Parameters of the system
  - σ: propagation time between 2 stations
  - λ: time to send a data
  - *slot*: time unit for the random time to wait before retransmitting
- Classical problem: Computation of minimum and maximum probabilities of reaching a certain state
  - P_1: Minimum probability that sender 1 transmits its message after exactly 1 collision.
  - $P_{\leq 3}$: Minimum probability that sender 1 transmits its message after 3 collisions or less.
  - Depend on the values of the parameters

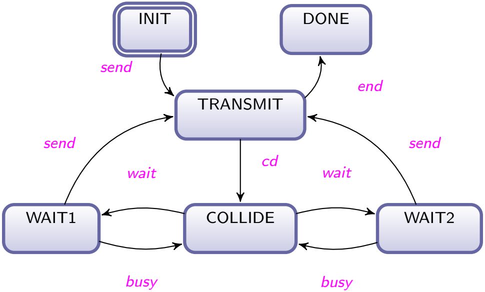
Probabilistic Timed Automaton (PTA)

- Timed Automaton (TA) [AD94]
  - Set of locations



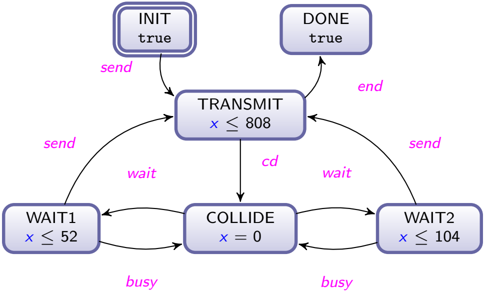
Probabilistic Timed Automaton (PTA)

- Timed Automaton (TA) [AD94]
  - Set of locations, set of actions



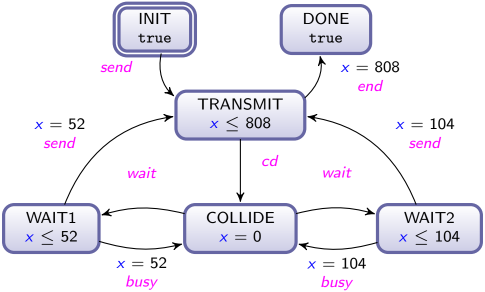
Probabilistic Timed Automaton (PTA)

- Timed Automaton (TA) [AD94]
  - Set of locations, set of actions
  - Set of clocks (real-valued variables increasing at the same linear rate)
    - ★ Operations: Location invariant



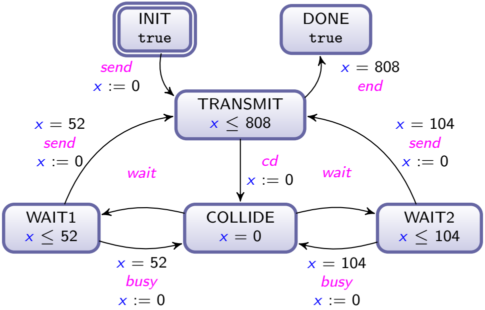
Probabilistic Timed Automaton (PTA)

- Timed Automaton (TA) [AD94]
  - Set of locations, set of actions
  - Set of clocks (real-valued variables increasing at the same linear rate)
    - ⋆ Operations: Location invariant, transition guard



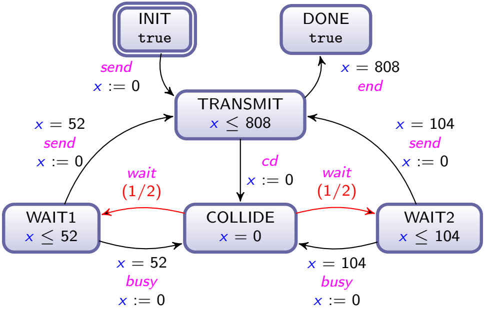
Probabilistic Timed Automaton (PTA)

- Timed Automaton (TA) [AD94]
  - Set of locations, set of actions
  - Set of clocks (real-valued variables increasing at the same linear rate)
    - ★ Operations: Location invariant, transition guard, clock reset



Probabilistic Timed Automaton (PTA)

- Timed Automaton (TA) [AD94]
  - Set of locations, set of actions
  - Set of clocks (real-valued variables increasing at the same linear rate)
    - ★ Operations: Location invariant, transition guard, clock reset
- Augmented with probabilities [Jen96, KNSS02]
  - The sum of the probabilities leaving a given location through a given action is equal to 1



Problem (1/2)

- Model CSMA/CD with Probabilistic Timed Automata
- Instantiation of the parameters
  - IEEE standard 802.3 for 10 Mbps Ethernet
    $\pi_0 := \{\lambda = 808\,\mu s\ ,\quad slot = 52\,\mu s\ ,\quad \sigma = 26\,\mu s\}$
  - Values too large for Prism (state-space explosion)
  - Use a set of rescaled values
    $\pi_1 := \{\lambda = 95\,\mu s\ ,\quad slot = 6\,\mu s\ ,\quad \sigma = 3\,\mu s\}$
- Computation of min/max probabilities using Prism with π_1
  - P_1: Minimum probability that sender 1 transmits its message after exactly 1 collision. $P_1 = 0.5$
  - $P_{\leq 3}$: Minimum probability that sender 1 transmits its message after 3 collisions or less. $P_{\leq 3} = 0.96875$

Problem (2/2)

- Prism does not formally guarantee that the rescaling does not affect the probabilities
  - Are the probabilities for π_1 the same as for π_0?
  - Need for a formal justification for rescaling

Problem (2/2)

- Prism does not formally guarantee that the rescaling does not affect the probabilities
  - Are the probabilities for π_1 the same as for π_0?
  - Need for a formal justification for rescaling
- More generally:

Goal

Given an instantiation π_0, compute a constraint K_0 on the parameters s.t.

1. $\pi_0 \models K_0$, *and*
2. *for all* $\pi \models K_0$, *the minimum and maximum probabilities for reachability properties are the same for* π_0 *and* π.

Problem (2/2)

- Prism does not formally guarantee that the rescaling does not affect the probabilities
  - Are the probabilities for π_1 the same as for π_0?
  - Need for a formal justification for rescaling
- More generally:

Goal

Given an instantiation π_0, compute a constraint K_0 on the parameters s.t.

1. $\pi_0 \models K_0$, *and*
2. *for all $\pi \models K_0$, the minimum and maximum probabilities for reachability properties are the same for π_0 and π.*

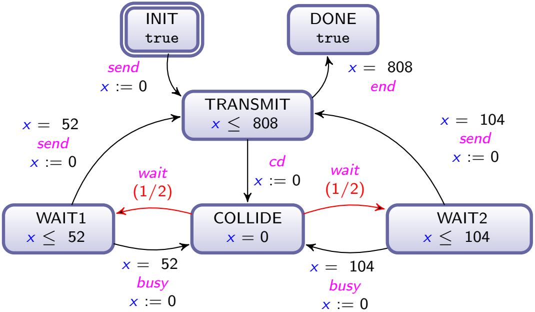
Inst.	λ	*slot*	σ	$\models K_0$	P_1	$P_{\leq 3}$
π_0	808	52	26	yes	0.5	0.96875
π_1	95	6	3	yes	0.5	0.96875

Outline

1. The CSMA/CD Protocol
   - Description
   - The Model of Probabilistic Timed Automata
   - The Problem
2. The Extension of the Inverse Method
   - Probabilistic Parametric Timed Automata
   - Our Method
   - Correctness
3. Implementation and Case Studies
4. Final Remarks

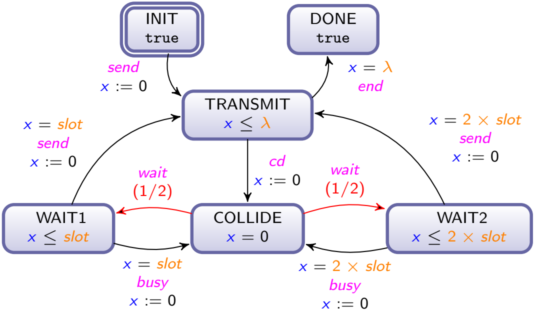
Probabilistic Timed Automaton

- Probabilistic Timed Automaton [Jen96, KNSS02]
  - Set of locations, set of actions, set of clocks
  - probabilities



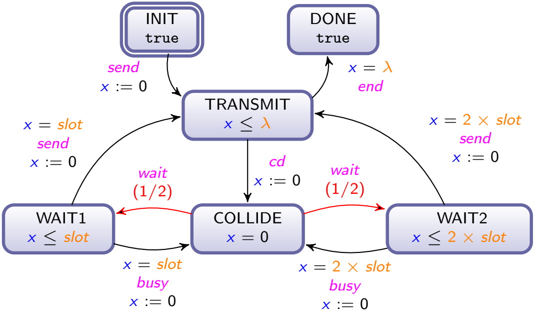
Probabilistic Parametric Timed Automaton (PPTA)

- Probabilistic Timed Automaton [Jen96, KNSS02]
  - Set of locations, set of actions, set of clocks
  - probabilities
  - Set of parameters (unknown constants) [AFS09]



Probabilistic Parametric Timed Automaton (PPTA)

- Probabilistic Timed Automaton [Jen96, KNSS02]
  - Set of locations, set of actions, set of clocks
  - probabilities
  - Set of parameters (unknown constants) [AFS09]



  - Given a PPTA $\mathcal{A}$ and an instantiation π of the parameters, we denote by $\mathcal{A}[\pi]$ the (non-parametric) PTA where all parameters where replaced by their value as defined by π

The Inverse Problem for PPTAs

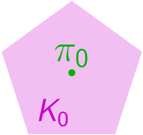
- Inputs
  - A Probabilistic Parametric Timed Automaton $\mathcal{A}$
  - A reference instantiation π_0 of all the parameters of $\mathcal{A}$

π_0

The Inverse Problem for PPTAs

- Inputs
  - A Probabilistic Parametric Timed Automaton $\mathcal{A}$
  - A reference instantiation π_0 of all the parameters of $\mathcal{A}$

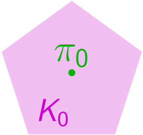
- Output: generalisation
  - A constraint K_0 on the parameters such that
    - $\pi_0 \models K_0$
    - For all instantiation $\pi \models K_0$, the sets of probabilistic traces (alternating sequences of locations with probabilities, and actions) of $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$ are equal



The Inverse Problem for PPTAs

- Inputs
  - A Probabilistic Parametric Timed Automaton $\mathcal{A}$
  - A reference instantiation π_0 of all the parameters of $\mathcal{A}$

- Output: generalisation
  - A constraint K_0 on the parameters such that
    - $\pi_0 \models K_0$
    - For all instantiation $\pi \models K_0$, the sets of probabilistic traces (alternating sequences of locations with probabilities, and actions) of $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$ are equal



As a consequence, the minimum and maximum probabilities for reachability properties in $\mathcal{A}[\pi]$ are the same as in $\mathcal{A}[\pi_0]$

Our Method: Overall Principle

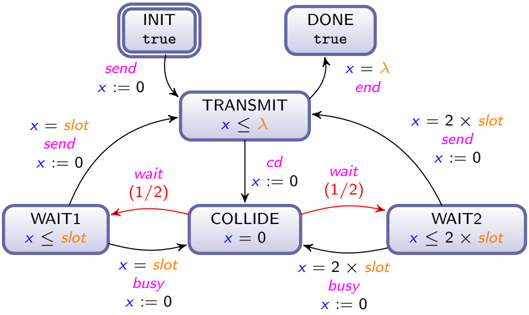
Starting with a PPTA $\mathcal{A}$, and an instantiation π_0 of the parameters:

1. Construct a non-probabilistic version $\mathcal{A}^*$ of $\mathcal{A}$
2. Compute a constraint K_0^* by applying the inverse method for classical parametric timed automata to $\mathcal{A}^*$ and π_0

Then, K_0^* also solves the inverse problem for $\mathcal{A}$
($K_0 = K_0^*$).

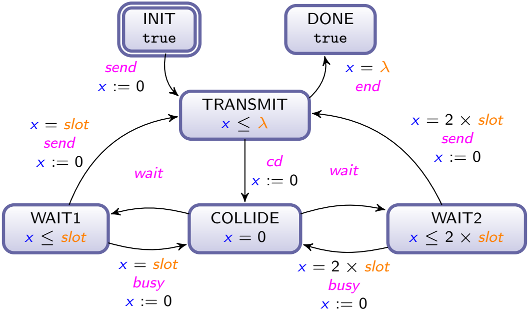
Non-probabilistic version $\mathcal{A}^*$ of a PPTA $\mathcal{A}$

- Replace stochastic distributions by non-determinism



Non-probabilistic version $\mathcal{A}^*$ of a PPTA $\mathcal{A}$

- Replace stochastic distributions by non-determinism

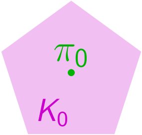


Overview of the Inverse Method for Classical TAs

- Algorithm *InverseMethod* [ACEF09]
- Inputs
  - A Parametric Timed Automaton $\mathcal{A}^*$
  - A reference instantiation π_0 of all the parameters of $\mathcal{A}^*$

Overview of the Inverse Method for Classical TAs

- Algorithm *InverseMethod* [ACEF09]
- Inputs
  - A Parametric Timed Automaton $\mathcal{A}^*$
  - A reference instantiation π_0 of all the parameters of $\mathcal{A}^*$
- Output: generalisation
  - A constraint K_0 on the parameters such that
    - $\pi_0 \models K_0$
    - For all instantiation $\pi \models K_0$, the set of traces (alternating sequences of locations and actions) under π is the same as the set of traces under π_0



Correctness of our Method

Theorem (Correctness)

Let $\mathcal{A}$ be a PPTA, and π_0 be an instantiation of the parameters of $\mathcal{A}$. Let $K_0 = InverseMethod(\mathcal{A}^*, \pi_0)$.
Then, for all $\pi \models K_0$, the sets of probabilistic traces (alternating sequences of locations with probabilities, and actions) of $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$ are equal.

Idea of the Proof

- Given $\pi \models K_0$:

$\boxed{\mathcal{A}[\pi_0]}$	$\boxed{\mathcal{A}^*[\pi_0]}$
$\boxed{\mathcal{A}[\pi]}$	$\boxed{\mathcal{A}^*[\pi]}$

Idea of the Proof

- Given $\pi \models K_0$:

$$\begin{tikzcd} \mathcal{A}[\pi_0] \arrow[rr, leftrightarrow, "\text{Prop. 1}"] & & \mathcal{A}^*[\pi_0] \\ \mathcal{A}[\pi] & & \mathcal{A}^*[\pi] \end{tikzcd}$$

- Justification
  - Prop. 1: The sets of non-probabilistic traces of $\mathcal{A}[\pi_0]$ and $\mathcal{A}^*[\pi_0]$ are equal [AFS09]

Idea of the Proof

- Given $\pi \models K_0$:

$$\begin{tikzcd}
\mathcal{A}[\pi_0] \arrow[rr, leftrightarrow, "\text{Prop. 1}"] & & \mathcal{A}^*[\pi_0] \arrow[d, leftrightarrow, "\text{Th. 1}"] \\
\mathcal{A}[\pi] & & \mathcal{A}^*[\pi]
\end{tikzcd}$$

- Justification
  - Prop. 1: The sets of non-probabilistic traces of $\mathcal{A}[\pi_0]$ and $\mathcal{A}^*[\pi_0]$ are equal [AFS09]
  - Th. 1: The sets of (non-probabilistic) traces of $\mathcal{A}^*[\pi_0]$ and $\mathcal{A}^*[\pi]$ are equal [ACEF09]

Idea of the Proof

- Given $\pi \models K_0$:

$$\begin{tikzcd} \mathcal{A}[\pi_0] \arrow[rr, leftrightarrow, "\text{Prop. 1}"] & & \mathcal{A}^*[\pi_0] \arrow[d, leftrightarrow, "\text{Th. 1}"] \\ \mathcal{A}[\pi] \arrow[rr, leftrightarrow, "\text{Prop. 1}"] & & \mathcal{A}^*[\pi] \end{tikzcd}$$

- Justification
  - Prop. 1: The sets of non-probabilistic traces of $\mathcal{A}[\pi_0]$ and $\mathcal{A}^*[\pi_0]$ are equal [AFS09]
  - Th. 1: The sets of (non-probabilistic) traces of $\mathcal{A}^*[\pi_0]$ and $\mathcal{A}^*[\pi]$ are equal [ACEF09]

Idea of the Proof

- Given $\pi \models K_0$:

$$\begin{tikzcd}[column sep=huge, row sep=huge]
\mathcal{A}[\pi_0] \arrow[r, leftrightarrow, "\text{Prop. 1}"] \arrow[d, leftrightarrow, bend left, "\text{Transitivity}"] & \mathcal{A}^*[\pi_0] \arrow[d, leftrightarrow, "\text{Th. 1}"] \\
\mathcal{A}[\pi] \arrow[r, leftrightarrow, "\text{Prop. 1}"] & \mathcal{A}^*[\pi]
\end{tikzcd}$$

- Justification
  - Prop. 1: The sets of non-probabilistic traces of $\mathcal{A}[\pi_0]$ and $\mathcal{A}^*[\pi_0]$ are equal [AFS09]
  - Th. 1: The sets of (non-probabilistic) traces of $\mathcal{A}^*[\pi_0]$ and $\mathcal{A}^*[\pi]$ are equal [ACEF09]

Idea of the Proof

- Given $\pi \models K_0$:

$$
\begin{tikzcd}
\mathcal{A}[\pi_0] \arrow[rr, leftrightarrow, "\text{Prop. 1}"] \arrow[d, leftrightarrow, "\text{Prop. 2}"'] \arrow[d, leftrightarrow, bend left, "\text{Transitivity}"] & & \mathcal{A}^*[\pi_0] \arrow[d, leftrightarrow, "\text{Th. 1}"] \\
\mathcal{A}[\pi] \arrow[rr, leftrightarrow, "\text{Prop. 1}"] & & \mathcal{A}^*[\pi]
\end{tikzcd}
$$

- Justification
  - Prop. 1: The sets of non-probabilistic traces of $\mathcal{A}[\pi_0]$ and $\mathcal{A}^*[\pi_0]$ are equal [AFS09]
  - Th. 1: The sets of (non-probabilistic) traces of $\mathcal{A}^*[\pi_0]$ and $\mathcal{A}^*[\pi]$ are equal [ACEF09]
  - Prop. 2: If the sets of non-probabilistic traces of $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$ are equal, then the sets of probabilistic traces of $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$ are equal [KNS02, KNS03]

Outline

Implementation

- Inverse method implemented in IMITATOR [And09]
  - IMITATOR: "Inverse Method for Inferring Time AbstracT BehaviOR"
  - 1500 lines of code in Python
  - 4 man-months of work
  - Calls the parametric model checker HYTECH [HHWT95]
    - Used by IMITATOR for the computation of the *Post* operation
  - Web page: http://www.lsv.ens-cachan.fr/~andre/IMITATOR

- Some case studies

Example	# of PTAs	loc. per PTA	# of clocks	# of param.	# of iter.	$\lvert Post^* \rvert$	$\lvert K_0 \rvert$	CPU time
CSMA/CD [CSM02, KNSW07, wp]	3	$[3, 8]$	3	3	17	218	3	44 s
RCP [SS01]	5	$[6, 11]$	6	5	18	154	2	70 s
WLAN [wp, KNS02]	3	$[1, 15]$	2	8	21	294	13	108 s

Case Studies (1/2)

- CSMA/CD Protocol [CSM02, KNSW07, wp]
  - IEEE standard 802.3 for 10 Mbps Ethernet
    $\pi_0 := \{\lambda = 808\,\mu s\ ,\quad slot = 52\,\mu s\ ,\quad \sigma = 26\,\mu s\}$
  - Constraint computed by IMITATOR:
    $K_0 :\quad \sigma < slot \quad \wedge \quad 15 slot < \lambda < 16 slot$
  - Recall that $\pi_1 \models K_0$
    $\pi_1 := \{\lambda = 95\,\mu s\ ,\quad slot = 6\,\mu s\ ,\quad \sigma = 3\,\mu s\}$
  - We can thus compute P_1 and $P_{\leq 3}$ using π_1, and apply the result to π_0 (by correction of our method)

Case Studies (2/2)

- Root Contention Protocol [SS01]
  - Reference instantiation π_0:
    $rc\_fast\_max = 85ns$ $rc\_fast\_min = 76ns$ $rc\_slow\_max = 167ns$
    $rc\_slow\_min = 159ns$ $delay = 30ns$
  - Constraint output by our method
    $K_0 : 2delay < rc\_fast\_min \wedge rc\_fast\_max + 2delay < rc\_slow\_min$

- Wireless Local Area Network Protocol [wp, KNS02]
  - Reference instantiation π_0:
    $ASLOTTIME = 1\mu s$ $DIFS = 2\mu s$ $VULN = 1\mu s$ $TTMAX = 315\mu s$
    $TTMIN = 4\mu s$ $ACK\_TO = 6\mu s$ $ACK = 4\mu s$ $SIFS = 1\mu s$
  - Constraint output by our method

$$
\begin{array}{rlrlrl}
 & VULN > 0 & \wedge & SIFS > 0 & \wedge & ACK\_TO + DIFS < 15ASLOTTIME \\
\wedge & DIFS > 0 & \wedge & ASLOTTIME > 0 & \wedge & TTMIN + DIFS \leq TTMAX \\
\wedge & ACK \leq 2DIFS & \wedge & DIFS < TTMIN & \wedge & ACK\_TO + DIFS \leq ACK + TTMIN \\
\wedge & SIFS < TTMIN & \wedge & TTMIN \geq ACK & \wedge & TTMIN \leq ACK\_TO \\
\wedge & VULN < ACK & & & &
\end{array}
$$

Outline

Conclusion

- Generalisation method
  - Model a system with a probabilistic parametric timed automaton $\mathcal{A}$
  - Starting with an instantiation π_0 of the parameters, we synthesise a constraint K_0 on the parameters guaranteeing that, for any $\pi \models K_0$, the min/max probabilities of reaching some state are equal for $\mathcal{A}[\pi]$ and $\mathcal{A}[\pi_0]$
- Advantages
  - Useful to determine probabilities (e.g., using Prism) for systems with large constants
  - Avoid the repeated computation of probabilities for many different values of the parameters
- Applications: Probabilistic systems
  - Protocols of communication
  - Hardware verification

Future Works

- Deal with soft deadline properties
  - E.g., probability of reaching some state within some deadline
  - Fall beyond the class of properties considered here
- Consider methods to enlarge our constraint
  - The constraint output by the inverse method for classical parametric timed automata is not maximal
    - Not the weakest constraint solving the inverse problem
  - Consider iterative methods [ACEF09]
- Consider continuous probabilities
  - For now, we considered continuous time with discrete probabilities
  - Allow to model more classes of systems

References I

Étienne André, Thomas Chatain, Emmanuelle Encrenaz, and Laurent Fribourg.
An inverse method for parametric timed automata.
International Journal of Foundations of Computer Science, 2009.
To appear.

R. Alur and D. L. Dill.
A theory of timed automata.
TCS, 126(2):183–235, 1994.

É. André, L. Fribourg, and J. Sproston.
An extension of the inverse method to probabilistic timed automata.
In *AVOCS'09*, 2009.

Étienne André.
IMITATOR: A tool for synthesizing constraints on timing bounds of timed automata.
In Martin Leucker and Carroll Morgan, editors, *Proceedings of the 6th International Colloquium on Theoretical Aspects of Computing (ICTAC'09)*, volume 5684 of *Lecture Notes in Computer Science*, pages 336–342, Kuala Lumpur, Malaysia, August 2009. Springer.

IEEE 802.3-2002: Carrier sense multiple access with collision detection (CSMA/CD) standard, 2002.

References II

T. A. Henzinger, P. Ho, and H. Wong-Toi.
A user guide to HyTech.
In *TACAS*, pages 41–71, 1995.

A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker.
PRISM: A tool for automatic verification of probabilistic systems.
In *TACAS'06*, volume 3920 of *LNCS*, pages 441–444. Springer, 2006.

H. E. Jensen.
Model checking probabilistic real time systems.
In *Proc. of the 7th Nordic Work. on Progr. Theory*. Chalmers Institute of Technology, 1996.

M. Kwiatkowska, G. Norman, and J. Sproston.
Probabilistic model checking of the IEEE 802.11 wireless local area network protocol.
In *Proc. PAPM/PROBMIV'02*, volume 2399 of *LNCS*, pages 169–187. Springer, 2002.

M. Kwiatkowska, G. Norman, and J. Sproston.
Probabilistic model checking of deadline properties in the IEEE 1394 FireWire root contention protocol.
Formal Aspects of Computing, 14(3):295–318, 2003.

M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston.
Automatic verification of real-time systems with discrete probability distributions.
TCS, 282:101–150, 2002.

References III

M. Kwiatkowska, G. Norman, J. Sproston, and F. Wang.
Symbolic model checking for probabilistic timed automata.
Information and Computation, 205(7):1027–1077, 2007.

D. Simons and M. Stoelinga.
Mechanical verification of the IEEE 1394a Root Contention Protocol using UPPAAL2k.
International Journal on Software Tools for Technology Transfer, 3(4):469–485, 2001.

PRISM web page.
`http://www.prismmodelchecker.org/`.