Computable analysis and control synthesis over complex dynamical systems via formal verification

Alessandro Abate

Department of Computer Science, University of Oxford
Delft Center for Systems and Control, TU Delft

September 25, 2013
Outline

1. Formal abstractions for verification of complex models

2. Formal verification of stochastic hybrid systems
   - Analysis and control synthesis problems
   - Computable analysis and control synthesis via formal abstractions

3. Formal verification of max-plus linear models
   - Analysis and control synthesis problems
   - Computable analysis and control synthesis via formal abstractions

4. Concluding remarks

Key references will appear here
Outline

1. Formal abstractions for verification of complex models

2. Formal verification of stochastic hybrid systems
   - Analysis and control synthesis problems
   - Computable analysis and control synthesis via formal abstractions

3. Formal verification of max-plus linear models
   - Analysis and control synthesis problems
   - Computable analysis and control synthesis via formal abstractions

4. Concluding remarks
Formal abstractions for verification of complex models

- concrete complex model
- property, specification, cost or reward

\[
\epsilon\text{-spec holds yes/no} \quad \mu \rightarrow \epsilon\text{-spec}
\]
Formal abstractions for verification of complex models

\[ \varepsilon \text{-quantitative abstraction} \]

| concrete complex model | property, specification, cost or reward | \[ \varepsilon \text{-quantitative abstraction} \] |
Formal abstractions for verification of complex models

Abstract simple model

\( \varepsilon \)-specification

\( \varepsilon \)-quantitative abstraction

Concrete complex model

Property, specification, cost or reward
Formal abstractions for verification of complex models

- Abstract simple model
- $\epsilon$-specification
- Automatic verification
- Control synthesis
- $\epsilon$-quantitative abstraction
- Concrete complex model
- Property, specification, cost or reward
Formal abstractions for verification of complex models

- Abstract simple model
  - $\epsilon$-specification

- $\epsilon$-quantitative abstraction

- Concrete complex model
  - Property, specification, cost or reward

model checking

automatic verification

control synthesis
Formal abstractions for verification of complex models

abstract simple model

$\epsilon$-specification

$\epsilon$-quantitative abstraction

model checking

automatic verification

control synthesis

$\epsilon$-spec holds yes/no

policy $\mu \rightarrow \epsilon$-spec

concrete complex model

property, specification, cost or reward

Alessandro Abate
Formal abstractions for verification of complex models

Abstract simple model

$\varepsilon$-specification

Model checking

Automatic verification

Control synthesis

$\varepsilon$-spec holds yes/no

Policy $\mu \rightarrow \varepsilon$-spec

$\varepsilon$-quantitative abstraction

Concrete complex model

Property, specification, cost or reward

Refine back
Formal abstractions for verification of complex models

- Formal abstractions are used to simplify complex models.
- Abstract, simple models are used for verification.
- ε-specification is used for automatic verification.
- ε-spec holds yes/no policy μ → ε-spec
- ε-quantitative abstraction is used to refine complex models.
- Concrete, complex models are used to represent properties, specifications, costs, or rewards.
- Spec holds yes/no policy μ → spec (correct by design)
Formal abstractions for verification of complex models

abstract simple model

$\epsilon$-specification

model checking

automatic verification

control synthesis

$\epsilon$-spec holds yes/no

policy $\mu \rightarrow \epsilon$-spec

$\epsilon$-quantitative abstraction

refine back

concrete complex model

property, specification, cost or reward

spec holds yes/no

policy $\mu \rightarrow$ spec

(correct by design)

if no, tune $\epsilon$
Outline

1. Formal abstractions for verification of complex models

2. Formal verification of stochastic hybrid systems
   - Analysis and control synthesis problems
   - Computable analysis and control synthesis via formal abstractions

3. Formal verification of max-plus linear models
   - Analysis and control synthesis problems
   - Computable analysis and control synthesis via formal abstractions

4. Concluding remarks
Formal abstractions for verification of complex models

- **abstract simple model**
  - $\epsilon$-specification

- **concrete complex model**
  - property, specification, cost or reward

- **model checking**
  - automatic verification
  - control synthesis

- $\epsilon$-spec holds yes/no
  - policy $\mu \rightarrow \epsilon$-spec

- $\epsilon$-quantitative abstraction

- **refine back**
  - spec holds yes/no
    - policy $\mu \rightarrow \text{spec}$ (correct by design)

- if no, tune $\epsilon$
Formal abstractions for verification of dtSHS

- dtMC
- dtMDP
- relax’d/strenght’d PCTL
- inflated LTL – $\epsilon$-spec

PRISM
MRMC
prob. model
checking
dynamic
programming

$\epsilon$-spec holds
policy max/min $\epsilon$-spec

adaptive, sequential abstractions
approximate probabilistic bisimulations

refine back

spec holds policy max/min spec

PCTL
LTL – spec
automata
Stochastic hybrid (discrete/continuous) systems

PRISM
MRMC
prob. model checking
\( \epsilon \)-spec holds
policy max/min \( \epsilon \)-spec

dtMC
relax’d/strenght’d PCTL
inflated LTL – \( \epsilon \)-spec

dtMDP

dtSHS
PCTL
LTL – spec automata

adaptive, sequential abstractions
approximate probabilistic bisimulations

refine back

spec holds policy max/min spec
Stochastic hybrid (discrete/continuous) systems

- discrete-time models

  finite-space Markov chain

  \((Z, T)\)

  \(Z = (z_1, z_2, z_3)\)

  \(T = \begin{bmatrix}
p_{11} & p_{12} & p_{13} \\
p_{21} & \cdots & \cdots \\
\cdots & \cdots & \cdots
\end{bmatrix}\)

  \(P(z_1, \{z_2, z_3\}) = p_{12} + p_{13}\)

  uncountable-space Markov process

  \((S, T_s)\)

  \(S = \mathbb{R}^2\)

  \(T_s(x|s) = e^{-\frac{1}{2}(x-m(s))^T \Sigma^{-1}(s)(x-m(s))} \frac{1}{\sqrt{2\pi|\Sigma(s)|^{1/2}}}\)

  \(P(s, A) = \int_A T_s(dx|s), \quad A \in \mathcal{B}(S)\)
Stochastic hybrid (discrete/continuous) systems

- **discrete-time models**

  finite-space Markov chain

  \[(\mathcal{Z}, T)\]

  \[\mathcal{Z} = (z_1, z_2, z_3)\]

  \[T = \begin{bmatrix}
  p_{11} & p_{12} & p_{13} \\
  p_{21} & \cdots & \cdots \\
  \cdots & \cdots & \cdots
  \end{bmatrix}\]

  \[P(z_1, \{z_2, z_3\}) = p_{12} + p_{13}\]

  uncountable-space Markov process

  \[(\mathcal{S}, T_s)\]

  \[\mathcal{S} = \mathbb{R}^2\]

  \[T_s(x|s) = \frac{e^{-\frac{1}{2}(x-m(s))^T\Sigma^{-1}(s)(x-m(s))}}{\sqrt{2\pi|\Sigma(s)|^{1/2}}}\]

  \[P(s, A) = \int_A T_s(dx|s), \quad A \in \mathcal{B}(\mathcal{S})\]

  \[\Rightarrow \text{discrete-time, stochastic hybrid systems}\]
Stochastic hybrid (discrete/continuous) systems

Definition

A discrete-time **stochastic hybrid system** is a pair \((S, T_s)\), where

- \(S = \bigcup_{q \in Q} (\{q\} \times \mathbb{R}^{n(q)})\), \(Q\) a discrete set of modes, \(n : Q \rightarrow \mathbb{N}\)
- \(T_s : S \times S \rightarrow [0, 1]\) specifies the dynamics of process at point \(s = (q, x)\):

\[
T_s(ds' | s) = \begin{cases} 
T_x(dx'|(q, x)) T_q(q|(q, x)), & \text{if } q' = q \text{ (no transition)} \\
T_r(dx'|(q, x), q') T_q(q'|(q, x)), & \text{if } q' \neq q \text{ (transition)}
\end{cases}
\]

- **initial state** \(\pi : S \rightarrow [0, 1]\)

[AA et al - Automatica 08]
Stochastic hybrid (discrete/continuous) systems

**Definition**

A discrete-time stochastic hybrid system is a pair \((S, T_s)\), where

- \(S = \bigcup_{q \in Q} \{q\} \times \mathbb{R}^{n(q)}\), \(Q\) a discrete set of modes, \(n : Q \to \mathbb{N}\)
- \(T_s : S \times S \to [0, 1]\) specifies the dynamics of process at point \(s = (q, x)\):

\[
T_s(ds' | s) = \begin{cases} 
    T_x(dx'|(q, x)) T_q(q|(q, x)), & \text{if } q' = q \text{ (no transition)} \\
    T_r(dx'|(q, x), q') T_q(q'|(q, x)), & \text{if } q' \neq q \text{ (transition)} 
\end{cases}
\]

- initial state \(\pi : S \to [0, 1]\)
- can be control dependent \((u \in U)\):

\[
T_s(ds' | s, u) = \begin{cases} 
    T_x(dx'|(q, x), u) T_q(q|(q, x), u), & \text{if } q' = q \text{ (no transition)} \\
    T_r(dx'|(q, x), u, q') T_q(q'|(q, x), u), & \text{if } q' \neq q \text{ (transition)} 
\end{cases}
\]

- policy \(\mu\): “string” of controls
- equivalent dynamical representation: \(s_{k+1} = f(s_k, \xi_k, u_k)\)
- related to other models, e.g. LMP

[AA et al - Automatica 08]
Stochastic hybrid systems in risk analysis

\[
\begin{align*}
Z_{n+1} &= g(Z_n, \theta_n) \quad Z_n \in \mathbb{R},
\theta_{n+1} &= h(Z_n, \theta_n, \xi_n) \quad \theta_n \in \{\Theta_1, \ldots, \Theta_N\},
\end{align*}
\]

where \(\xi_n\) i.i.d. random variables; \(g, h\) measurable; \((Z_0, \theta_0)\) given

![Diagram of stochastic hybrid system]

[1. Tkachev, AA - CDC 11]
Stochastic hybrid systems in risk analysis

\[
\begin{align*}
Z_{n+1} &= g(Z_n, \theta_n) \quad Z_n \in \mathbb{R}, \\
\theta_{n+1} &= h(Z_n, \theta_n, \xi_n) \quad \theta_n \in \{\Theta_1, \ldots, \Theta_N\},
\end{align*}
\]

where \( \xi_n \) i.i.d. random variables; \( g, h \) measurable; \((Z_0, \theta_0)\) given

- **objective:** what is the probability that, starting from initial capital \( Z_0 = x \), high capitalization \( y \) is reached, while company’s bankruptcy is avoided

[I. Tkachev, AA - CDC 11]
Outline

1. Formal abstractions for verification of complex models

2. Formal verification of stochastic hybrid systems
   - Analysis and control synthesis problems
   - Computable analysis and control synthesis via formal abstractions

3. Formal verification of max-plus linear models
   - Analysis and control synthesis problems
   - Computable analysis and control synthesis via formal abstractions

4. Concluding remarks
Analysis and control synthesis problems

PRISM
MRMC
prob. model checking

\( \epsilon \)-spec holds
policy max/min \( \epsilon \)-spec

dynamic programming

\begin{align*}
dtMC & \quad \text{relax'd/strenght'd PCTL} \\
dtMDP & \quad \text{inflated LTL} - \ \epsilon \text{-spec}
\end{align*}

adaptive, sequential
abstractions

approximate probabilistic
bisimulations

\begin{align*}
dtSHS & \quad \text{PCTL} \\
& \quad \text{LTL - spec}
\end{align*}

\text{spec holds}
policy max/min spec

refine back
Analysis and control synthesis problems

- **reachability**
  - (safety/invariance)

- **reach-avoid**
  - (constrained reachability)

- **sequential reachability**
  - (trajectory planning)

- **∞-horizon objectives**
  - (i.o., eventually always)

- Properties expressed via PCTL, LTL (DFA or Büchi automata)
Analysis and control synthesis problems

- synthesis for reachability games (2 – 1/2 players)
- synthesis for reach-avoid (pursuit evasion games)
- sequential reachability (trajectory planning)
- ∞-horizon objectives (i.o., eventually always)

- properties expressed via PCTL, LTL (DFA or Büchi automata)
Probabilistic safety/invariance: characterization

- probabilistic invariance is the probability that the execution associated with an initial distribution $\pi$ stays in $S$ (safe set) during the time horizon $[0, N]$:

$$\mathcal{P}_\pi(S) := P_\pi(s_k \in S, \forall k \in [0, N])$$
Probabilistic safety/invariance: characterization

- probabilistic invariance is \textit{the probability that the execution associated with an initial distribution } \pi \textit{ stays in } S \textit{ (safe set) during the time horizon } [0, N]:

\[ P_{\pi}(S) := P_{\pi}(s_k \in S, \forall k \in [0, N]) \]

- consider realization \( s_k \in S, k \in [0, N] \) – then

\[ \prod_{k=0}^{N} 1_{S}(s_k) = \begin{cases} 1, & \text{if } \forall k \in [0, N] : s_k \in S \\ 0, & \text{otherwise} \end{cases} \]

\[ \Rightarrow P_{\pi}(S) = P_{\pi}\left(\prod_{k=0}^{N} 1_{S}(s_k) = 1\right) = E_{\pi}\left[\prod_{k=0}^{N} 1_{S}(s_k)\right] \]

[AA et al. - Automatica 08]
Probabilistic safety/invariance: characterization

- **probabilistic invariance** is the probability that the execution associated with an initial distribution $\pi$ stays in $S$ (safe set) during the time horizon $[0, N]$:

$$\mathcal{P}_{\pi}(S) := P_{\pi}(s_k \in S, \forall k \in [0, N])$$

- consider realization $s_k \in S$, $k \in [0, N]$ – then

$$\prod_{k=0}^{N} 1_{S}(s_k) = \begin{cases} 1, & \text{if } \forall k \in [0, N] : s_k \in S \\ 0, & \text{otherwise} \end{cases}$$

$$\Rightarrow \mathcal{P}_{\pi}(S) = P_{\pi} \left( \prod_{k=0}^{N} 1_{S}(s_k) = 1 \right) = E_{\pi} \left[ \prod_{k=0}^{N} 1_{S}(s_k) \right]$$

- select $\epsilon \in [0, 1]$ – probabilistic safe/invariant set with safety level $\epsilon$ is

$$S(\epsilon) \doteq \{ s \in S : \mathcal{P}_{s}(S) \geq \epsilon \} \quad \text{(here } \pi = \delta_s)$$

[AA et al. - Automatica 08]
Probabilistic invariance: computation

- computation of $\mathcal{P}_s(S)$ (and thus of $S(\epsilon)$) via **dynamic programming**: sequential update, backward in time, of multi-stage value function

$$V_k(s) : [0, N] \times S \rightarrow \mathbb{R}^+,$$

accounting for current and expected future rewards – in particular

$$V_N(s) = 1_S(s), \quad V_k(s) = \int_S V_{k+1}(x) T_s(dx|s),$$

$$V_0(s) = \mathcal{P}_s(S) \Rightarrow S(\epsilon)$$
Probabilistic invariance: computation

- computation of $\mathcal{P}_s(S)$ (and thus of $S(\epsilon)$) via dynamic programming: sequential update, backward in time, of multi-stage value function

$$V_k(s) : [0, N] \times S \to \mathbb{R}^+,$$

accounting for current and expected future rewards – in particular

$$V_N(s) = 1_S(s), \quad V_k(s) = \int_S V_{k+1}(x) T_s(dx|s)$$

$$V_0(s) = \mathcal{P}_s(S) \Rightarrow S(\epsilon)$$

- control dependent models: find optimal policy $\mu$, optimizing recursively over

$$V_k(s, u) : [0, N] \times S \times U \to \mathbb{R}^+$$
Computing probabilistic invariance: issues

- non-standard (max, multiplicative) value functions
- continuous control space
- hybrid state space

⇒ solution of DP is seldom analytical
Computing probabilistic invariance: issues

- issues
  1. non-standard (max, multiplicative) value functions
  2. continuous control space
  3. hybrid state space

⇒ solution of DP is seldom analytical

- numerical solutions are needed

⇒ problem # 1: difference between real solution and computed solution
  (in verification and correct-by-design controller synthesis)

⇒ problem # 2: Bellman’s curse of dimensionality
  (state/control space gridding)
Outline

1. Formal abstractions for verification of complex models

2. Formal verification of stochastic hybrid systems
   - Analysis and control synthesis problems
   - *Computable analysis and control synthesis via formal abstractions*

3. Formal verification of max-plus linear models
   - Analysis and control synthesis problems
   - *Computable analysis and control synthesis via formal abstractions*

4. Concluding remarks
Dynamical properties as temporal specifications

- PRISM
- MRMC

\[ \text{prob. model checking} \]

\[ \epsilon\text{-spec holds} \]

\[ \text{policy max/min } \epsilon\text{-spec} \]

- dtMC
- dtMDP

\[ \text{relax’d/strenght’d PCTL} \]

\[ \text{inflated LTL} - \epsilon\text{-spec} \]

- adaptive, sequential abstractions
- approximate probabilistic bisimulations

- dtSHS

\[ \text{PCTL} \]

\[ \text{LTL} - \text{spec automata} \]

- refine back

\[ \text{spec holds} \]

\[ \text{policy max/min spec} \]

Alessandro Abate
Approximate model checking of probabilistic invariance

- model \((\mathcal{S}, T_s)\), invariance set \(S \in \mathcal{S}\), finite time horizon \(N\), safety level \(\epsilon\)
Approximate model checking of probabilistic invariance

- model \((\mathcal{S}, T_s)\), invariance set \(\mathcal{S} \in \mathcal{S}\), finite time horizon \(N\), safety level \(\epsilon\)
- \(\delta\)-approximate \((\mathcal{S}, T_s)\) with finite-state dt-MC \((\mathcal{Z}, \mathcal{T})\)
- compute approximation error \(f(\delta, N)\)
- \(\mathcal{S} \rightarrow S_\delta\): define formula \(\Phi_{S_\delta}\) characterizing set \(S_\delta\), label states in \(\mathcal{Z}\)

\[ \text{probabilistic safe set } S(\epsilon) = \{ s \in S : P_s(S) \geq \epsilon \} = \{ s \in S : (1 - P_s(S)) \leq 1 - \epsilon \} \]

[AA et al. - EJC 11]
Approximate model checking of probabilistic invariance

- model \((S, T_S)\), invariance set \(S \in S\), finite time horizon \(N\), safety level \(\epsilon\)
- \(\delta\)-approximate \((S, T_S)\) with finite-state dt-MC \((Z, \mathcal{I})\)
- compute approximation error \(f(\delta, N)\)
- \(S \rightarrow S_\delta\): define formula \(\Phi_{S_\delta}\) characterizing set \(S_\delta\), label states in \(Z\)

\[ S(\epsilon) = \{ s \in S : P_s(S) \geq \epsilon \} \]
\[ = \{ s \in S : (1 - P_s(S)) \leq 1 - \epsilon \} \]
Approximate model checking of probabilistic invariance

- model \((\mathcal{S}, T_s)\), invariance set \(S \in \mathcal{S}\), finite time horizon \(N\), safety level \(\epsilon\)
- \(\delta\)-approximate \((\mathcal{S}, T_s)\) with finite-state dt-MC \((\mathcal{Z}, \mathcal{T})\)
- compute approximation error \(f(\delta, N)\)
- \(S \to S_\delta\): define formula \(\Phi_{S_\delta}\) characterizing set \(S_\delta\), label states in \(\mathcal{Z}\)

\[ S(\epsilon) = \{ s \in \mathcal{S} : P_s(S) \geq \epsilon \} \]
\[ = \{ s \in \mathcal{S} : (1 - P_s(S)) \leq 1 - \epsilon \} \]

\(\Rightarrow\) probabilistic safe set

\[ Z_\delta(\epsilon) \doteq \text{Sat} \left( P_{\leq 1-\epsilon} \left( \text{true} \cup_{\leq N} \neg \Phi_{S_\delta} \right) \right) \]
\[ = \{ z \in \mathcal{Z} : z \models P_{\leq 1-\epsilon} \left( \text{true} \cup_{\leq N} \neg \Phi_{S_\delta} \right) \} \]

[AA et al. - EJC 11]
Approximate model checking of probabilistic invariance

- model \((S, T_s)\), invariance set \(S \in S\), finite time horizon \(N\), safety level \(\epsilon\)
- \(\delta\)-approximate \((S, T_s)\) with finite-state dt-MC \((Z, T)\)
- compute approximation error \(f(\delta, N)\)
- \(S \rightarrow S_\delta\): define formula \(\Phi_{S_\delta}\) characterizing set \(S_\delta\), label states in \(Z\)

1. define

\[
S(\epsilon) = \{s \in S : P_s(S) \geq \epsilon\} \\
Z_\delta(\epsilon) = \text{Sat} \left( \mathbb{P}^{\leq 1 - \epsilon} (\text{true} \cup^{\leq N} \neg \Phi_{S_\delta}) \right)
\]

2. select \(\eta > 0 : \eta/2 \in (0, 1 - \epsilon)\)
3. pick \(\delta : f(\delta, N) \leq \eta/2\)
4. compute \(Z_\delta(\epsilon + \eta/2)\)
5. define \(\hat{S}_\eta(\epsilon) = \{s \in S \leftrightarrow z \in Z_\delta(\epsilon + \eta/2)\}\)

\[
\Rightarrow \quad S(\epsilon + \eta) \subseteq \hat{S}_\eta(\epsilon) \subseteq S(\epsilon)
\]

[AA et al. - EJC 11]
Verification of over- or under-specifications in PCTL

- any PCTL formula can be expressed via equivalent DP recursions

- consider PCTL formula $\mathbb{P} \sim \epsilon (\Psi)$ on SHS $(S, T_s)$
- $\delta$-approximate SHS $(S, T_s)$ as a dt-MC $(Z, T)$
- compute approximation error $f(\delta, N)$

[D’Innocenzo, AA, J.-P. Katoen - HSCC 12]
Verification of over- or under-specifications in PCTL

- any PCTL formula can be expressed via equivalent DP recursions

- consider PCTL formula $P \sim_\epsilon (\Psi)$ on SHS $(S, T_s)$
- $\delta$-approximate SHS $(S, T_s)$ as a dt-MC $(Z, T)$
- compute approximation error $f(\delta, N)$

- compute $g(\Psi, f)$, a function based on formula & error
- model check $P \sim_{\epsilon \pm g(\Psi, f)} (\Psi)$ on $(Z, T)$

1. if PCTL formula is “robust”, then conclusion holds for $P \sim_\epsilon (\Psi)$ on SHS
2. else refine $\delta \rightarrow$ reduce $f(\delta, N) \rightarrow$ decrease $g(\Psi, f)$

[D’Innocenzo, AA, J.-P. Katoen - HSCC 12]
Approximate model checking of automata specifications

- generalization to “richer” set of properties over dtSHS
- specifications expressed as a DFA or a Büchi automata
- probabilistic reachability-like computation over product construction
- recent extensions to controller synthesis

[AA et al. - HSCC 11; I. Tkachev et al. - HSCC13]
Characterization & computation of $\infty$-horizon properties

- consider target set $T$; invariant set $S = T^c = S \setminus T$; $\forall s \in S$:

$$P_s(\forall n \geq 0 : s_n \in S) \iff 1 - P_s(\text{true} \cup T)$$

[I. Tkachev, AA - CDC 11, HSCC 12, CDC12, TCS 13]
Characterization & computation of $\infty$-horizon properties

- consider target set $T$; invariant set $S = T^c = S \setminus T$; $\forall s \in S$:
  \[ P_s(\forall n \geq 0 : s_n \in S) \iff 1 - P_s(\text{true} \cup T) \]

- existence and computation of absorbing set $B$ : $\forall x \in B$, $T_s(B|x) = 1$

- characterization – study of existence/uniqueness of (non-trivial) solutions of Bellman equations
  convergence of Bellman recursions, contractivity of operators

- computation – formal reduction to finite-horizon problems

\[ l. \ Tkachev, \ AA \ - \ CDC \ 11, \ HSCC \ 12, \ CDC12,TCS \ 13 \]
On the approximation error $f(\delta, N)$

PRISM MRMC
prob. model checking
dynamic programming

$\epsilon$-spec holds policy max/min $\epsilon$-spec

adaptive, sequential abstractions
approximate probabilistic bisimulations

refine back

spec holds policy max/min spec

dtMC dtMDP
relax’d/strengthen’d PCTL inflated LTL – $\epsilon$-spec

PCTL LTL – spec automata

dtSHS
On the approximation error $f(\delta, N)$

- approximation via $\delta$-partitioning: $S = \bigcup_{i=1,\ldots,m} S^i$

- under Lip-continuity assumptions on density of kernel $T_s$,

\[ h(i, j), \quad i, j = 1, \ldots, m \]

- for any $z_q^i \in S_\delta$, $\forall s : s \wedge z^i \in S^i$, error is

\[ f(\delta, N) \doteq |P_s(S) - P_{z^i}(S_\delta)| \leq \max_{i=1,\ldots,m} N\delta_i \sum_{j=1,\ldots,m} h(i, j), \]

\[ \delta = \max_{i=1,\ldots,m} \delta_i, \quad \delta_{q,i} = \text{diam} (S^i) \]

error is linear in $N, \delta_i$ and depends on local constants $h(i, j) \rightarrow$ local tuning

[AA et al. - EJC 11, S. Soudjani, AA - QEST 11, TAC 13]
On the approximation error $f(\delta, N)$

- **formula-based abstractions**
- software (in the making) for sequential, adaptive grid generation based on approximation error
- from MATLAB/Simulink model to MRMC/PRISM input

[Figures showing numerical data and heatmaps related to approximation error]

[S. Soudjani, AA - QEST 11, HSCC 12, ATVA12, SIAM 13]
Approximate probabilistic bisimulations

- *dtMC* 
  - relax’d/strenght’d PCTL
  - inflated LTL – $\varepsilon$-spec

- *dtMDP*

- *dtSHS*
  - PCTL
  - LTL – spec automata

- *PRISM, MRMC*
  - prob. model checking

- $\varepsilon$-spec holds
  - policy max/min $\varepsilon$-spec

- adaptive, sequential abstractions
  - approximate probabilistic bisimulations

- refine back
  - spec holds
    - policy max/min spec

Alessandro Abate
Approximate probabilistic bisimulations

- above abstraction leads to approximate probabilistic bisimulation \([Larsen \& Skou, 91]\) - alternatively . . .

- consider models \((T_{s,i}, S_i)\) with solution processes \(s_i(k), i = 1, 2, k \geq 0\)
- parallel composition of models with output \(s_{1,2}(k) = s_1(k) - s_2(k)\)
Approximate probabilistic bisimulations

- above abstraction leads to approximate probabilistic bisimulation [Larsen & Skou, 91] - alternatively . . .

- consider models \((T_{s,i}, S_i)\) with solution processes \(s_i(k), i = 1, 2, k \geq 0\)
- parallel composition of models with output \(s_{1,2}(k) = s_1(k) - s_2(k)\)

**Definition**

A function \(\psi : S_1 \times S_2 \rightarrow \mathbb{R}^+\) is a **probabilistic bisimulation function** if

\[
\psi(s_{1,2}) \geq \|s_1 - s_2\|^2
\]

and if \(\psi_{s_0}(s_{1,2}(k))\) is a **supermartingale**.

- \(\psi\) is an upper bound on the distance btw solutions of two models:

\[
P_{s_0} \left( \sup_{k \geq 0} \|s_1(k) - s_2(k)\|^2 \geq \epsilon \right) \leq \psi_{s_0}(s_{1,2}(0))/\epsilon
\]
Outline

1. Formal abstractions for verification of complex models

2. Formal verification of stochastic hybrid systems
   - Analysis and control synthesis problems
   - Computable analysis and control synthesis via formal abstractions

3. Formal verification of max-plus linear models
   - Analysis and control synthesis problems
   - Computable analysis and control synthesis via formal abstractions

4. Concluding remarks
Formal abstractions for verification of complex models

- **abstract simple model**
  - \( \epsilon \)-specification

- **model checking**
  - automatic verification
  - control synthesis

- \( \epsilon \)-spec holds yes/no
  - policy \( \mu \rightarrow \epsilon \)-spec

- **\( \epsilon \)-quantitative abstraction**

- **concrete complex model**
  - property, specification, cost or reward

- **refine back**
  - if no, tune \( \epsilon \)

- **spec holds yes/no**
  - policy \( \mu \rightarrow \text{spec} \) (correct by design)
Formal abstractions for verification of MPL models

\[
\text{LTS} \quad \text{LTL} \quad \text{safe LTL}
\]

\[
\text{VeriSiMPL} \quad \text{bisimulations} \quad \text{simulations}
\]

\[
\text{MPL} \quad \text{transient or steady-state}
\]

\[
\text{SPIN} \quad \text{model checking}
\]

\[
(\exists \text{policy}) \text{ spec yes/no} \\
(\forall \text{policies}) \text{ spec yes}
\]

\[
(\exists \text{policy}) \text{ property yes/no} \\
(\forall \text{policies}) \text{ property yes}
\]

\[
\text{refine back}
\]

\[
\text{determ.}
\]
Introduction to MPL systems

LTS

- LTL
- safe LTL

VeriSiMPL

bisimulations simulations

MPL

- transient or steady-state

SPIN

model checking

(∃ policy) spec yes/no
(∀ policies) spec yes

refine back

(∃ policy) property yes/no
(∀ policies) property yes

determ.
Introduction to MPL systems

- **Max-Plus-Linear (MPL) systems** are event-driven models
- Applications: railway scheduling, planning of production lines, network calculus

\[ x(k) \text{ is the time of } k\text{-th event, } k \in \mathbb{N} \cup \{0\} \]
- Timing updates: maximization \((\oplus)\) and addition \((\otimes)\) operations

\[ \epsilon = -\infty, \quad \mathbb{R}_\epsilon = \mathbb{R} \cup \{\epsilon\}, \quad \alpha, \beta \in \mathbb{R}_\epsilon \]
- \[ \alpha \oplus \beta := \max(\alpha, \beta), \quad \alpha \otimes \beta := \alpha + \beta, \quad \text{and matrix operations} \]
Max-plus-linear models

Definition (Autonomous MPL model)

\[ x(k + 1) = A \otimes x(k), \]

where \( A \in \mathbb{R}^{n \times n} \) and \( k \in \mathbb{N} \cup \{0\} \)

Example

A simple railway model [Heidergott, 06]

\[
\begin{bmatrix}
2 & 5 \\
3 & 3
\end{bmatrix} \otimes x(k), \quad
\begin{bmatrix}
x_1(k + 1) \\
x_2(k + 1)
\end{bmatrix} = \begin{bmatrix}
\max\{2 + x_1(k), 5 + x_2(k)\} \\
\max\{3 + x_1(k), 3 + x_2(k)\}
\end{bmatrix}
\]
Max-plus-linear models

Definition (Autonomous MPL model)

\[ x(k + 1) = A \otimes x(k), \]
where \( A \in \mathbb{R}^{n \times n}_\epsilon \) and \( k \in \mathbb{N} \cup \{0\} \)

Example

A simple railway model [Heidergott, 06]

\[ x(k + 1) = \begin{bmatrix} 2 & 5 \\ 3 & 3 \end{bmatrix} \otimes x(k), \quad \begin{bmatrix} x_1(k + 1) \\ x_2(k + 1) \end{bmatrix} = \begin{bmatrix} \max\{2 + x_1(k), 5 + x_2(k)\} \\ \max\{3 + x_1(k), 3 + x_2(k)\} \end{bmatrix} \]

Definition (Non-autonomous MPL model)

\[ x(k + 1) = A \otimes x(k) \oplus B \otimes u(k), \]
where \( B \in \mathbb{R}^{n \times m}_\epsilon \) and \( u \in \mathbb{R}^m \) (synthesis = scheduling)

[Heidergott, 06]
Outline

1. Formal abstractions for verification of complex models

2. Formal verification of stochastic hybrid systems
   - Analysis and control synthesis problems
   - Computable analysis and control synthesis via formal abstractions

3. Formal verification of max-plus linear models
   - Analysis and control synthesis problems
   - Computable analysis and control synthesis via formal abstractions

4. Concluding remarks
Classical analysis of MPL models

- LTS
  - LTL
    - safe LTL

- VeriSiMPL
  - bisimulations simulations

- MPL
  - transient or steady-state

- SPIN
  - model checking
    - (∃ policy) spec yes/no
    - (∀ policies) spec yes

- refine back
  - (∃ policy) property yes/no
  - (∀ policies) property yes

- detterm.
Classical analysis of MPL models

- study of transient and periodic regimes, of asymptotics
- classical analysis based on algebraic or geometric properties

**Definition**

1. **max-plus eigenvector** $x \in \mathbb{R}^n$: $A \otimes x = \lambda \otimes x \Rightarrow x(k + 1) = \lambda \otimes x(k)$
2. **cycles on precedence graph** $\Rightarrow$ periodic regime with period $c$:
   $\forall k \geq k_0$, $x(k + c) = \lambda^{\otimes c} \otimes x(k)$

**Example**

1. eigenspace (periodic regime with period 1 and $\lambda = 4$):
   
   $\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 5 \\ 4 \end{bmatrix}, \begin{bmatrix} 9 \\ 8 \end{bmatrix}, \begin{bmatrix} 13 \\ 12 \end{bmatrix}, \begin{bmatrix} 17 \\ 16 \end{bmatrix}, \begin{bmatrix} 21 \\ 20 \end{bmatrix}, \begin{bmatrix} 25 \\ 24 \end{bmatrix}, \begin{bmatrix} 29 \\ 28 \end{bmatrix}, \begin{bmatrix} 33 \\ 32 \end{bmatrix}, \begin{bmatrix} 37 \\ 36 \end{bmatrix}, \begin{bmatrix} 41 \\ 40 \end{bmatrix}, \begin{bmatrix} 45 \\ 44 \end{bmatrix}, \ldots$

2. periodic regime with period $c = 2$ (transient $k_0 = 3$):
   
   $\begin{bmatrix} 4 \\ 0 \end{bmatrix}, \begin{bmatrix} 6 \\ 7 \end{bmatrix}, \begin{bmatrix} 12 \\ 10 \end{bmatrix}, \begin{bmatrix} 15 \\ 15 \end{bmatrix}, \begin{bmatrix} 20 \\ 18 \end{bmatrix}, \begin{bmatrix} 23 \\ 23 \end{bmatrix}, \begin{bmatrix} 28 \\ 26 \end{bmatrix}, \begin{bmatrix} 31 \\ 31 \end{bmatrix}, \begin{bmatrix} 36 \\ 34 \end{bmatrix}, \begin{bmatrix} 39 \\ 39 \end{bmatrix}, \begin{bmatrix} 44 \\ 42 \end{bmatrix}, \begin{bmatrix} 47 \\ 47 \end{bmatrix}, \ldots$
Outline

1. Formal abstractions for verification of complex models

2. Formal verification of stochastic hybrid systems
   - Analysis and control synthesis problems
   - Computable analysis and control synthesis via formal abstractions

3. Formal verification of max-plus linear models
   - Analysis and control synthesis problems
   - Computable analysis and control synthesis via formal abstractions

4. Concluding remarks
Labeled transition system (LTS)

- **LTS**
  - LTL
  - safe LTL

- **VeriSiMPL**
  - bisimulations simulations

- **MPL**
  - transient or steady-state

- **SPIN**
  - model checking
  - $(\exists \text{ policy}) \text{ spec yes/no}$
  - $(\forall \text{ policies}) \text{ spec yes}$

- **(∃ policy) property yes/no**
  - $(\forall \text{ policies}) \text{ property yes}$

- **refine back**

**Deterministic (determ.)**
Labeled transition system (LTS)

- set of states $S = \{1, 2, 3, 4\}$
- set of inputs $Act = \{\alpha, \beta\}$
- transitions $\xrightarrow{} = \{(1, \alpha, 4), (4, \alpha, 3), \ldots\}$
- set of outputs $AP = \{a, b\}$ and output map $L(1) = \emptyset$, $L(2) = \{b\}$, $\ldots$

- labels can be defined over states or transitions
- LTS can be deterministic vs non-deterministic
- LTS can be infinite vs finite

[Baier & Katoen, 08]
Finite LTS as abstractions of MPL models

- LTS
  - LTL
  - Safe LTL

- MPL
  - Transient or steady-state

- VeriSiMPL
  - Bisimulations simulations

- SPIN
  - Model checking
  - (∃ policy) spec yes/no
  - (∀ policies) spec yes

Procedure: need to compute
1. $S$: states of LTS
2. $\rightarrow$: LTS transitions
3. $L$: LTS labels

Refine back
- (∃ policy) property yes/no
- (∀ policies) property yes
LTS states: partitioning of state space

- state space $\mathbb{R}^n$ is partitioned in finitely many polytopic regions
- partition is not arbitrary, it is adapted to underlying dynamics
- obtained state-space partition defines states of LTS
- partition can be possibly refined (determinization – more later)

Example

- we obtain a total of 5 regions:
  - $R_1 = \{ x \in \mathbb{R}^2 : x_1 - x_2 < 0 \}$
  - $R_2 = \{ x \in \mathbb{R}^2 : x_1 - x_2 = 0 \}$
  - $R_3 = \{ x \in \mathbb{R}^2 : x_1 - x_2 > 3 \}$
  - $R_4 = \{ x \in \mathbb{R}^2 : x_1 - x_2 = 3 \}$
  - $R_5 = \{ x \in \mathbb{R}^2 : 0 < x_1 - x_2 < 3 \}$
Difference-bound matrices (DBM)

**Definition (DBM)**

A difference-bound matrix in $\mathbb{R}^n$ is the finite intersection of sets defined by

$$x_i - x_j \simeq i,j \alpha_{i,j},$$

where $\simeq_{i,j} \in \{<, \leq\}$, $\alpha_{i,j} \in \mathbb{R} \cup \{+\infty\}$, for $1 \leq i \neq j \leq n$.

- DBM allow **compact matrix representation**
- DBM are **easy to manipulate** (projections, emptiness and inclusion check)

[Dill, 90]
Difference-bound matrices (DBM)

**Definition (DBM)**

A difference-bound matrix in $\mathbb{R}^n$ is the finite intersection of sets defined by

$$x_i - x_j \simeq_{i,j} \alpha_{i,j},$$

where $\simeq_{i,j} \in \{<, \leq\}$, $\alpha_{i,j} \in \mathbb{R} \cup \{+\infty\}$, for $1 \leq i \neq j \leq n$

- DBM allow compact matrix representation
- DBM are easy to manipulate (projections, emptiness and inclusion check)
- closure: image/inverse image of DBM over MPL dynamics is again a DBM

[Dill, 90]
LTS transitions: one-step reachability

- consider any two TS states (partitioning regions) $R, R'$
- $R \rightarrow R'$ iff there exists a $x(k) \in R$ such that $x(k + 1) \in R'$: check

\[
R' \cap \{ x(k + 1) : x(k) \in R \} \neq \emptyset
\]
LTS transitions: one-step reachability

- consider any two TS states (partitioning regions) $R, R'$
- $R \rightarrow R'$ iff there exists a $x(k) \in R$ such that $x(k + 1) \in R'$: check
  \[ R' \cap \{x(k + 1) : x(k) \in R\} \neq \emptyset \]

- computation of transitions:
  - use region representation via DBM, DBM forward-mapping via PWA dynamics, DBM emptiness check
- transitions are stored on sparse Boolean matrix
LTS transitions, an example

- **determinism vs non-determinism** of obtained TS
- above $R_i$ - *original* partitions, $R'_i$ - *refined* partitions (determinization)
Relationship between LTS and MPL

LTS

LTL
safe LTL

VeriSiMPL

bisimulations simulations

MPL

transient or steady-state

SPIN

model checking

(∃ policy) spec yes/no
(∀ policies) spec yes

(∃ policy) property yes/no
(∀ policies) property yes

refine back
determ.
Relationship between LTS and MPL

Theorem

- **TS simulates the original MPL model**
- **TS bisimulates the MPL model if and only if it is deterministic**

- non-deterministic TS can be “determinized” by refining partitioning regions
- however, refinement procedure may not terminate

Theorem

- if TS is deterministic over the periodic regime, then TS is globally deterministic
- every irreducible MPL model admits finite deterministic TS abstraction
LTS labels

**Definition**

- **state labels:**
  all possible values of $x_i(k) - x_j(k)$, for $1 \leq i < j \leq n$
  time difference of *same-event variables*

- **transition labels:**
  all possible values of $x_i(k + 1) - x_i(k)$, for $1 \leq i \leq n$
  time difference of *successive events*

- labels are *vectors of intervals*, can be represented as *DBM*
LTS labels, an example

Example

- LTS transition labels
Formal analysis of MPL models is now “very simple”
VeriSiMPL – Verification via biSimulation of MPL models

LTS

| LTL safe LTL |

VeriSiMPL

bisimulations simulations

MPL

transient or steady-state

SPIN

model checking

(∃ policy) spec yes/no
(∀ policies) spec yes

refine back

(∃ policy) property yes/no
(∀ policies) property yes

determ.
Formal analysis of MPL models is now “very simple”

VeriSiMPL – Verification via biSimulation of MPL models

- abstract MPL model as LTS (in MATLAB)
- export LTS abstraction (as PROMELA script) into SPIN model checker
- consider properties in LTL logic
- verify property via SPIN over LTS and export outcome back to MPL model

VeriSiMPL (“very simple”)
Verification via biSimulations of Max-Plus Linear Models

- is a software tool for concrete MPL models implemented in Matlab, which exports abstract LTS models to SPIN in Promela language

Documentation
  - comes as a text file: txt

Download
  - the toolbox as a compressed folder: zip

Contacts
  - for questions and queries, please send an email to
    - D. Adzkiya, d dot adzkiya at tudelft dot nl
    - A. Abate, a dot abate at tudelft dot nl

http://sourceforge.net/projects/verisimipl
MPL verification in practice

Example

- automatically identify MPL eigenspace: \( \bigvee_{\varphi \in L=AP} (\square \varphi \land |\varphi| = 0) \)
MPL verification in practice

Example

- automatically identify MPL periodic regime: \( \Psi = \bigvee_{\varphi \in \mathcal{L}=\mathcal{AP}} \Box (\varphi \land \Box^c \varphi) \)
Computational benchmark for abstraction

- coded in MATLAB, run over 12-core Intel Xeon, 3.47 GHz, 24 GB
- A randomly generated with elements taking values between 1 and 100
- 10 independent experiments per dimension – mean values are displayed:

<table>
<thead>
<tr>
<th>size of MPL model</th>
<th>time for generation of states</th>
<th>time for generation of transitions</th>
<th>time for generation of labels</th>
<th>total number of LTS states</th>
<th>total number of LTS transitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.1 [s]</td>
<td>0.4 [s]</td>
<td>0.1 [s]</td>
<td>3.6</td>
<td>4.3</td>
</tr>
<tr>
<td>5</td>
<td>0.2 [s]</td>
<td>0.4 [s]</td>
<td>0.1 [s]</td>
<td>8.6</td>
<td>13.8</td>
</tr>
<tr>
<td>7</td>
<td>0.9 [s]</td>
<td>0.5 [s]</td>
<td>0.3 [s]</td>
<td>37.2</td>
<td>289.3</td>
</tr>
<tr>
<td>9</td>
<td>4.1 [s]</td>
<td>0.8 [s]</td>
<td>1.6 [s]</td>
<td>120.0</td>
<td>1.7·10³</td>
</tr>
<tr>
<td>11</td>
<td>24.8 [s]</td>
<td>15.2 [s]</td>
<td>16.1 [s]</td>
<td>613.2</td>
<td>1.9·10⁴</td>
</tr>
<tr>
<td>13</td>
<td>3.5 [m]</td>
<td>5.5 [m]</td>
<td>2.8 [m]</td>
<td>1.9·10³</td>
<td>1.9·10⁵</td>
</tr>
<tr>
<td>15</td>
<td>53.6 [m]</td>
<td>2.0 [h]</td>
<td>39.4 [m]</td>
<td>7.4·10³</td>
<td>2.0·10⁶</td>
</tr>
</tbody>
</table>

- bottleneck: generation of transitions
Computational benchmark for reachability analysis

- *A randomly generated* with elements taking values between 1 and 100
- set of *initial conditions* is selected as the unit hypercube
- *10 independent experiments* per dimension – mean values are displayed:

<table>
<thead>
<tr>
<th>size of MPL model</th>
<th>time for generation of abstract TS</th>
<th>number of regions of abstract TS</th>
<th>time for generation of reach tube</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.09 [s]</td>
<td>5</td>
<td>0.09 [s]</td>
</tr>
<tr>
<td>10</td>
<td>4.73 [s]</td>
<td>700</td>
<td>8.23 [s]</td>
</tr>
<tr>
<td>19</td>
<td>67.07 [m]</td>
<td>$3.48 \cdot 10^5$</td>
<td>7.13 [h]</td>
</tr>
</tbody>
</table>

- *generation time for reach tube* of 10-dimensional MPL model, different time horizons
- *comparison VeriSiMPL vs MPT* (multi-parametric tool, ETH Zürich):

<table>
<thead>
<tr>
<th>time horizon</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>VeriSiMPL</td>
<td>11.02 [s]</td>
<td>17.94 [s]</td>
<td>37.40 [s]</td>
<td>51.21 [s]</td>
<td>64.59 [s]</td>
</tr>
<tr>
<td>MPT</td>
<td>47.61 [m]</td>
<td>1.19 [h]</td>
<td>2.32 [h]</td>
<td>3.03 [h]</td>
<td>3.73 [h]</td>
</tr>
</tbody>
</table>
Stochastic Max-plus-linear models

Definition (Deterministic MPL model)

\[ x(k + 1) = A \otimes x(k), \]

where \( A \in \mathbb{R}^{n \times n} \) and \( k \in \mathbb{N} \cup \{0\} \)

Definition (Stochastic MPL model)

\[ x(k + 1) = A \otimes x(k), \]

where \( A(k) = [a_{ij}(k)]_{i,j} \in \mathbb{R}^{n \times n} \), \( \{a_{ij}(k)\}_k \) are i.i.d. random processes with pdf \( t_{ij}(\cdot) \), and \( k \in \mathbb{N} \cup \{0\} \)
Stochastic Max-plus-linear models

**Definition (Deterministic MPL model)**

\[ x(k + 1) = A \otimes x(k), \]

where \( A \in \mathbb{R}_{\geq}^{n \times n} \) and \( k \in \mathbb{N} \cup \{0\} \)

**Definition (Stochastic MPL model)**

\[ x(k + 1) = A \otimes x(k), \]

where \( A(k) = [a_{ij}(k)]_{i,j} \in \mathbb{R}_{\geq}^{n \times n} \), \( \{a_{ij}(k)\}_k \) are i.i.d. random processes with pdf \( t_{ij}(\cdot) \), and \( k \in \mathbb{N} \cup \{0\} \)

- abstraction of SMPL models as Markov chains
- can be obtained in two possible ways:
  1. leveraging theory above, under continuity assumptions on kernels \( t_{ij}(\cdot) \)
  2. by symbolic approach over distributions that are closed under max-plus algebra operations
- error quantification
Simulations over 2D SMPL model

- exponential distributions (rates btw 1/3 and 1) for the entries of 2D matrix $A$
- pick time horizon $N = 5$, safe set $\mathcal{A} = [-5, 5]^2$
- select $(3700, 2900)$ bins per dimension, partition uniformly
- abstraction error results in $E = 32.5\delta < 0.1$
Outline

1. Formal abstractions for verification of complex models

2. Formal verification of stochastic hybrid systems
   - Analysis and control synthesis problems
   - Computable analysis and control synthesis via formal abstractions

3. Formal verification of max-plus linear models
   - Analysis and control synthesis problems
   - Computable analysis and control synthesis via formal abstractions

4. Concluding remarks
Formal abstractions for verification of complex models

abstract simple model

$\epsilon$-specification

model checking

automatic verification

control synthesis

$\epsilon$-spec holds yes/no policy $\mu \rightarrow \epsilon$-spec

$\epsilon$-quantitative abstraction

refine back

concrete complex model

property, specification, cost or reward

spec holds yes/no policy $\mu \rightarrow \text{spec}$ (correct by design)

if no, tune $\epsilon$
Acknowledgments

- **students:** D. Adzkiya, S. Haesaert, S.E.Z. Soudjani, I. Tkachev, M. Zamani

- **main collaborators:** J. Lygeros, M. Prandini, J.-P. Katoen, C. Tomlin, B. De Schutter

- **topics:** stochastic hybrid systems, max-plus linear models
Thanks for your attention!

For more info:

www.dcsc.tudelft.nl/~aabate
a.abate@tudelft.nl
Selected key references