Automated Verification and Control Synthesis of CPS with SHS Models

Alessandro Abate

Department of Computer Science, University of Oxford

www.oxcav.org

Symposium on Stochastic Hybrid Systems - July 2021
Formal verification: successes and frontiers

- Industrial impact in checking correctness of

  protocols, hardware circuits, and software

- Model-based, automated, and sound guarantees (formal certificates)
Formal verification: successes and frontiers

- industrial impact in checking correctness of

  protocols, hardware circuits, and software

- model-based, automated, and sound guarantees (formal certificates)
Formal verification: successes and frontiers

- industrial impact in checking correctness of protocols, hardware circuits, and software

- model-based, automated, and sound guarantees (formal certificates)
Formal verification and control in the real world

- tech trends: advances in sensing, networking and embedded computation
Formal verification and control in the real world
1. integration of learning from data within model-based verification & control ("learning for verification and control")

2. certified reinforcement learning for policy synthesis ("certified learning")
Formal verification and control in the real world

- verification and control of **complex models**
  - hybrid models with uncertainty, noise
  - via **formal abstractions**
Building automation systems - a CPS exemplar

Building automation system setup in rooms 478/9 at Oxford CS

- advanced modelling for smart buildings
- applications: certifiable energy management
  1. control of temperature, humidity, CO₂
  2. model-based predictive maintenance of devices
  3. fault-tolerant certified control
  4. demand-response over smart grids
Building automation systems - a CPS exemplar

Building automation system setup in rooms 478/9 at Oxford CS

- advanced modelling for smart buildings
- applications: certifiable energy management
  - control of temperature, humidity, CO₂
  - model-based predictive maintenance of devices
  - fault-tolerant certified control
  - demand-response over smart grids
Building automation systems - a CPS exemplar

Building automation system setup in rooms 478/9 at Oxford CS

- advanced modelling for smart buildings
- applications: certifiable energy management
  1. control of temperature, humidity, CO₂
  2. model-based predictive maintenance of devices
  3. fault-tolerant certified control
  4. demand-response over smart grids
Building automation systems – a SHS

- model $\text{CO}_2$ dynamics, coupled with temperature evolution

\begin{align*}
  x_{k+1} &= x_k + \frac{\Delta}{V} \left( -\mathbb{1}_{\text{ON}} m x_k + \mu_{\{O,C\}} (C_{out} - x_k) \right) + \mathbb{1}_F C_{occ} + \sigma_x w_k \\
  y_{k+1} &= y_k + \frac{\Delta}{C} \left( \mathbb{1}_{\text{ON}} m (T_{set} - y_k) + \mu_{\{O,C\}} \frac{1}{R} (T_{out} - y_k) \right) + \mathbb{1}_F T_{occ,k} + \sigma_y w_k
\end{align*}

where $T_{occ,k} = \nu x_k + \zeta$

- $x$ - zone $\text{CO}_2$ level
- $y$ - zone temperature

- $T_{set}$ - set temperature (air circulation)
- $T_{out}$ - outside temperature (window)
- $T_{occ}$ - generated heat (occupants)
- $\sigma(\cdot)$ - variance of noise $w_k \sim \mathcal{N}(0,1)$
Building automation systems – a SHS

CO₂ levels

Zone Temperature

Occupancy (occupied, empty)

Windows (open, closed)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td>94.41 J/°C</td>
</tr>
<tr>
<td>$T_{set}$</td>
<td>20 °C</td>
</tr>
<tr>
<td>$T_{out}$</td>
<td>24 °C</td>
</tr>
<tr>
<td>$\nu$</td>
<td>$2.4 \cdot 10^{-4}$</td>
</tr>
<tr>
<td>$\zeta$</td>
<td>0.0107</td>
</tr>
</tbody>
</table>

- air circulation: ON
CPS models: both finite and uncountable

finite-space Markov chain

\((S, T)\)

\(S = (z_1, z_2, z_3, z_4)\)

\(T = \begin{bmatrix}
p_{11} & \cdots & p_{14} \\ 
\cdots & \cdots & \cdots \\ 
p_{41} & \cdots & \cdots 
\end{bmatrix}\)

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

uncountable-space Markov process

\((S, \mathcal{T})\)

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

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

\(\mathbb{P}(s, A) = \int_A \mathcal{T}(dx|s), \quad A \subseteq S\)
Stochastic hybrid (discrete/continuous) systems

- discrete-time, **stochastic hybrid system (SHS)**
  
  \((S, T_s)\)

  \(S = \bigcup_{q \in Q} (\{q\} \times \mathcal{X}), Q\) a discrete set of modes, \(\mathcal{X} = \mathbb{R}^n\)

  \(T_s : S \times S \rightarrow [0,1]\) specifies the dynamics of process at any hybrid point \((q, x)\)

- **model semantics**: initial state \(\pi : S \rightarrow [0,1];\)
  at any point \(s = (q, x),\)
    1. sample discrete kernel \(T_q \rightarrow \) select location \(q'\)
    2. conditional on \(q',\) sample continuous kernel \(T_x \rightarrow \) select point \(x'\)
Stochastic hybrid (discrete/continuous) systems

- \( T_s : S \times S \rightarrow [0, 1] \) specifies the dynamics of process at point \( s = (q, x) \):

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

- equivalent dynamical representation

  e.g., SDE with NL drift and Gaussian noise

\[
s(k + 1) = f(s(k)) + g(s(k))\eta(k), \quad \eta(\cdot) \sim \mathcal{N}(0, 1)
\]

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

- \( T_s : S \times S \rightarrow [0,1] \) specifies the dynamics of process at point \( s = (q, x) \):

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

- equivalent dynamical representation
  e.g., SDE with NL drift and Gaussian noise

\[
s(k + 1) = f(s(k)) + g(s(k))\eta(k), \quad \eta(\cdot) \sim \mathcal{N}(0, 1)
\]

- can be control/action dependent \((u \in U)\):

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

\[
T_s : S \times U \times S \rightarrow [0, 1]
\]

[AA et al - Automatica 08]
Probabilistic model checking of complex models

- general specifications expressed as PCTL formulae, e.g.

- simplest instance: **probabilistic safety** is the probability that the execution, started at \( s \), stays in safe set \( A \) during the time horizon \([0, N]\)

\[
P_s(A) = \mathbb{P}_s(s_k \in A, \forall k \in [0, N])
\]

- select \( p \in [0, 1] \); probabilistic safe set with safety level \( p \) is

\[
S(p) = \{s \in S : P_s(A) \geq p\}
\]

- PCTL formula: \( \mathbb{P}^{\leq 1-p} (\text{true} U^{\leq N} \neg A) \)
Probabilistic model checking of complex models

- general specifications expressed as PCTL formulae, e.g.

- simplest instance: probabilistic safety is the probability that the execution, started at \( s \), stays in safe set \( A \) during the time horizon \([0, N]\)

\[
\mathbb{P}_s(A) = \mathbb{P}_s(s_k \in A, \forall k \in [0, N])
\]

- select \( p \in [0, 1] \); probabilistic safe set with safety level \( p \) is

\[
S(p) = \{s \in S : \mathbb{P}_s(A) \geq p\}
\]

- PCTL formula: \( \mathbb{P} \leq 1-p (\text{true} \lor \leq N \neg A) \)

- \( \mathbb{P}_s(A) \) can be fully characterised (and optimised)

- issues with computation of \( \mathbb{P}_s(A) \) and of \( S(p) \)
Formal abstractions

- Complex model
- Specification
Formal abstractions

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

complex model \hspace{0.5cm} \text{specification}
Formal abstractions

abstract model

\( \xi \)-specification

\( \xi \)-quantitative abstraction

complex model

specification
Formal abstractions

abstract model

\( \zeta \)-specification

\( \zeta \)-quantitative abstraction

complex model

specification

automatic verification

if not, tune \( \zeta \)-spec holds, or \( \zeta \)-spec does not hold
Formal abstractions

- abstract model
  - $\zeta$-specification

  $\zeta$-quantitative abstraction

- complex model
  - specification

  model checking
  automatic verification

Alessandro Abate, CS, Oxford
Automated Verification and Control of SHS
Formal abstractions

abstract model

ξ-specification

model checking

automatic verification

ξ-spec holds, or ξ-spec does not hold

ξ-quantitative abstraction

complex model

specification

Alessandro Abate, CS, Oxford Automated Verification and Control of SHS slide 8 /15
Formal abstractions

abstract model

$\xi$-specification

$\xi$-quantitative abstraction

complex model

specification

model checking

automatic verification

$\xi$-spec holds, or $\xi$-spec does not hold

refine back
Formal abstractions

Abstract model

$\zeta$-specification

$\zeta$-quantitative abstraction

Model checking

Automatic verification

$\zeta$-spec holds, or
$\zeta$-spec does not hold

Complex model

Specification

Refine back

Spec holds, or
Spec does not hold
Formal abstractions

Abstract model

$\zeta$-specification

$\zeta$-quantitative abstraction

Model checking

Automatic verification

$\zeta$-spec holds, or $\zeta$-spec does not hold

Refine back

Complex model

Specification

If not, tune $\zeta$

Spec holds, or spec does not hold
Formal abstractions: algorithm

- approximate stochastic process \((S, T)\) as MC \((S, T)\), where
  - \(S = \{z_1, z_2, \ldots, z_p\}\) – finite set of abstract states
  - \(T : S \times S \rightarrow [0, 1]\) – transition probability matrix
Formal abstractions: algorithm

- approximate stochastic process \((S, \mathcal{T})\) as MC \((S, \mathcal{T})\), where
  - \(S = \{z_1, z_2, \ldots, z_p\}\) – finite set of abstract states
  - \(\mathcal{T} : S \times S \rightarrow [0, 1]\) – transition probability matrix

- algorithm:

  **input:** stochastic process \((S, \mathcal{T})\)

  **output:** MC \((S, \mathcal{T})\)
Formal abstractions: algorithm

- approximate stochastic process \((\mathcal{S}, \mathcal{T})\) as MC \((\mathcal{S}, \mathcal{T})\), where
  - \(\mathcal{S} = \{z_1, z_2, \ldots, z_p\}\) – finite set of abstract states
  - \(\mathcal{T} : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]\) – transition probability matrix

- algorithm:

  ```
  input: stochastic process \((\mathcal{S}, \mathcal{T})\)
  1. select finite partition \(\mathcal{S} = \bigcup_{i=1}^{p} S_i\)
  
  output: MC \((\mathcal{S}, \mathcal{T})\)
  ```
Formal abstractions: algorithm

- approximate stochastic process \((S, \mathcal{T})\) as MC \((S, \mathcal{T})\), where
  - \(S = \{z_1, z_2, \ldots, z_p\}\) – finite set of abstract states
  - \(\mathcal{T} : S \times S \rightarrow [0, 1]\) – transition probability matrix

- algorithm:
  
  \[
  \begin{align*}
  \textbf{input:} & \text{ stochastic process } (S, \mathcal{T}) \\
  1 & \text{ select finite partition } S = \bigcup_{i=1}^{p} S_i \\
  2 & \text{ select representative points } z_i \in S_i \\
  3 & \text{ define finite state space } S := \{z_i, i = 1, \ldots, p\}
  \end{align*}
  \]

  \[
  \textbf{output: } \text{MC } (S, \mathcal{T})
  \]
Formal abstractions: algorithm

- approximate stochastic process \((S, \mathcal{T})\) as MC \((S, \mathcal{T})\), where
  - \(S = \{z_1, z_2, \ldots, z_p\}\) – finite set of abstract states
  - \(\mathcal{T}: S \times S \rightarrow [0, 1]\) – transition probability matrix

- algorithm:

  input: stochastic process \((S, \mathcal{T})\)
  1. select finite partition \(S = \bigcup_{i=1}^{p} S_i\)
  2. select representative points \(z_i \in S_i\)
  3. define finite state space \(S := \{z_i, i = 1, \ldots, p\}\)
  4. compute transition probability matrix: \(\mathcal{T}(z_i, z_j) = \mathcal{T}(S_j \mid z_i)\)

  output: MC \((S, \mathcal{T})\)
Formal abstractions: algorithm

- approximate stochastic process \((S, \mathcal{T})\) as MC \((S, \mathcal{T})\), where
  - \(S = \{z_1, z_2, \ldots, z_p\}\) – finite set of abstract states
  - \(\mathcal{T} : S \times S \rightarrow [0, 1]\) – transition probability matrix

- algorithm:

  **input:** stochastic process \((S, \mathcal{T})\)
  
  1. select finite partition \(S = \bigcup_{i=1}^{p} S_i\)
  2. select representative points \(z_i \in S_i\)
  3. define finite state space \(S := \{z_i, i = 1, \ldots, p\}\)
  4. compute transition probability matrix: \(\mathcal{T}(z_i, z_j) = \mathcal{T}(S_j \mid z_i)\)

  **output:** MC \((S, \mathcal{T})\)
Model checking probabilistic safety via formal abstractions

- safety set $A \subset S$, time horizon $N$, safety level $p$
Model checking probabilistic safety via formal abstractions

- safety set $A \subset S$, time horizon $N$, safety level $p$
- $\delta$-abstract $(\mathcal{S}, \mathcal{T})$ as MC $(\mathcal{S}, \mathcal{T})$, so that $A \rightarrow A_\delta$,

 quantify error $\xi(\delta, N)$

$\Rightarrow$ probabilistic safe set

$$S(p) = \{s \in S : P_s(A) \geq p\}$$
$$= \{s \in S : (1 - P_s(A)) \leq 1 - p\}$$
Model checking probabilistic safety via formal abstractions

- safety set $A \subset S$, time horizon $N$, safety level $p$
- $\delta$-abstract $(S, \mathcal{T})$ as MC $(S, \mathcal{T})$, so that $A \rightarrow A_\delta$,
- quantify error $\xi(\delta, N)$

⇒ probabilistic safe set

$$S(p) = \{s \in S : \mathbb{P}_s(A) \geq p\}$$
$$= \{s \in S : (1 - \mathbb{P}_s(A)) \leq 1 - p\}$$

can be computed via

$$Z_\delta(p + \xi) \doteq \text{Sat} \left( \mathbb{P}_{\leq 1-p-\xi} \left( \text{true} \ U^{\leq N} \neg A_\delta \right) \right)$$
$$= \{z \in S : z \models \mathbb{P}_{\leq 1-p-\xi} \left( \text{true} \ U^{\leq N} \neg A_\delta \right)\}$$
Formal abstractions: error $\zeta$

- consider $\mathcal{T}(d\bar{s}|s) = t(\bar{s}|s)d\bar{s}$; assume $t$ is Lipschitz continuous, namely

$$\exists 0 \leq h_s < \infty : \quad |t(\bar{s}|s) - t(\bar{s}|s')| \leq h_s \|s - s'\|, \quad \forall s, s', \bar{s} \in S$$
Formal abstractions: error $\xi$

- consider $\mathcal{T}(d\bar{s}|s) = t(\bar{s}|s)d\bar{s}$; assume $t$ is Lipschitz continuous, namely

$$\exists 0 \leq h_s < \infty : \quad |t(\bar{s}|s) - t(\bar{s}|s')| \leq h_s \|s - s'\|, \quad \forall s, s', \bar{s} \in S$$

- **one-step error** \((related to approximate probabilistic bisimulation)\)

$$\epsilon = h_s \delta \mathcal{L}(A)$$

  - $\delta$ – max diameter of partition sets
  - $\mathcal{L}(A)$ – volume of set of interest

- **$N$-step error** \((tunable via $\delta$)\)

$$\xi(\delta, N) = \epsilon N$$
Formal abstractions: error  $\tilde{\xi}$

- consider $\mathcal{T}(d\bar{s}|s) = t(\bar{s}|s)d\bar{s}$; assume $t$ is Lipschitz continuous, namely
  \[ \exists 0 \leq h_s < \infty : \quad |t(\bar{s}|s) - t(\bar{s}|s')| \leq h_s \|s - s'\| , \quad \forall s, s', \bar{s} \in S \]

- **one-step error**  
  \[ \epsilon = h_s \delta \mathcal{L}(A) \]
  - $\delta$ – max diameter of partition sets
  - $\mathcal{L}(A)$ – volume of set of interest

- **$N$-step error**  
  \[ \xi(\delta, N) = \epsilon N \]
  
  → improved and generalised error

(related to approximate probabilistic bisimulation)
FAUST$^2$: software for formal abstractions

http://sourceforge.net/projects/faust2
FAUST\textsuperscript{2}: software for formal abstractions

http://sourceforge.net/projects/faust2

- sequential, adaptive, anytime
**FAUST²:** software for formal abstractions

http://sourceforge.net/projects/faust2

- sequential, adaptive, anytime
FAUST²: software for formal abstractions

http://sourceforge.net/projects/faust2

- sequential, adaptive, anytime
FAUST\textsuperscript{2}: software for formal abstractions

http://sourceforge.net/projects/faust2

- sequential, adaptive, anytime
FAUST$^2$: software for formal abstractions

http://sourceforge.net/projects/faust2

- sequential, adaptive, anytime
StocHy: software for formal abstractions

verification
- abstraction based
- novel algorithm with tighter bounds and more scalability

gitlab.com/natchi92/StocHy
StocHy: software for formal abstractions

**verification**
- abstraction based
- novel algorithm with tighter bounds and more scalability

**synthesis**
- abstraction based
- optimisation via sparse matrices

gitlab.com/natchi92/StocHy
StocHy: software for formal abstractions

**verification**
- abstraction based
- novel algorithm with tighter bounds and more scalability

**simulation**
- automatically generates statistics
- visualisation via time varying histograms

**synthesis**
- abstraction based
- optimisation via sparse matrices

**features**
- modular
- C++ implementation
- extendable
- multiple options

[gitlab.com/natchi92/StocHy](gitlab.com/natchi92/StocHy)
StocHy: software for formal abstractions

**verification**
- abstraction based
- novel algorithm with tighter bounds and more scalability

**simulation**
- automatically generates statistics
- visualisation via time varying histograms

**synthesis**
- abstraction based
- optimisation via sparse matrices

**features**
- modular
- C++ implementation
- extendable
- multiple options

gitlab.com/natchi92/StocHy
Building automation systems – case study

\[
\begin{align*}
  x_{k+1} &= x_k + \frac{\Delta}{V} \left( -\mathbbm{1}_{ON} m x_k + \mu \{ O, C \} \left( C_{out} - x_k \right) \right) + \mathbbm{1}_F C_{occ} + \sigma_x w_k \\
  y_{k+1} &= y_k + \frac{\Delta}{C} \left( \mathbbm{1}_{ON} m (T_{set} - y_k) + \mu \{ O, C \} \frac{1}{R} \left( T_{out} - y_k \right) \right) + \mathbbm{1}_F T_{occ,k} + \sigma_y w_k \\
\end{align*}
\]

where \( T_{occ,k} = \nu x_k + \zeta \)
Building automation systems – case study

- Safe set $A = [300\ 700] ppm \times [19\ 21]^\circ C$
- Air circulation: closed-loop control policy at $k+1$
  \[
  \begin{cases}
    \text{OFF} & \text{if } (x_k, y_k) \leq A \\
    \text{ON} & \text{if } (x_k, y_k) \geq A \\
    \text{stay put} & \text{else}
  \end{cases}
  \]
- Specification:
  \[
  \mathbb{P} = ? \left[ \square \leq 20 (x,y) \in A \right]
  \]

- 5 hours, 8:00-13:00 ($\Delta = 15$ min, N=20), divided into
  - 8:00-8:30 (N=2) - (E,C)
  - 8:30-11:30 (N=12) - (F,C)
  - 11:30-13:00 (N=6) - (F,O)
Building automation systems – case study

\[ \text{CO}_2 \text{ level (ppm)} \]

\[ \text{Probability} \]

\[ N=20, \sigma = 15\sqrt{\Delta} \]
Selected journal references


Thank you for your attention

For more info: aabate@cs.ox.ac.uk