Outline

Cyber-physical systems

Models for CPS

Verification and control of SHS

Correct-by-design control synthesis of CPS

Correct-by-design control synthesis of MPL systems

Beyond model-based deduction
Control Systems meet Computing and Verification

Cyber

Physical
Control Systems meet Computing and Verification

\[ \dot{\xi} = f(\xi, \nu) \]
\[
\dot{\xi} = f(\xi, \nu)
\]
CPS are Ubiquitous

Digital Communication/Computation + Control Plant = Cyber-Physical System (CPS)

“Internet of things”
CPS Example: Control with shared resource

The processor is a shared resource.

Other tasks
CPS Example: Control with shared resource

- The processor is a shared resource.

- scheduler reserves at most one slot every nine for the jet engine

- some of the possible microprocessor schedules:

  |aaaaaaaaaa|aaaaaaaaaa|aaaaaaaaaa|aaaaaaaaaa|aaaaaaaaaa|...
  |aaaaaaaaaa|aaaaaaaaaa|aaaaaaaaaa|aaaaaaaaaa|aaaaaaaaaa|...
  |aaaaaaaaaa|aaaaaaaaaa|aaaaaaaaaa|aaaaaaaaaa|aaaaaaaaaa|...
CPS Example: Control with shared resource

- schedule can be described using an automaton

\[
\begin{bmatrix}
\frac{d\phi}{dt} \\
\frac{d\psi}{dt}
\end{bmatrix} = \begin{bmatrix}
-\psi - \frac{3}{2} \phi^2 - \frac{1}{2} \phi^3 \\
\phi - \nu
\end{bmatrix} dt + \begin{bmatrix}
0.1 \delta W_1^t \\
0.1 \psi d W_2^t
\end{bmatrix}
\]
Outline

Cyber-physical systems

Models for CPS

Verification and control of SHS

Correct-by-design control synthesis of CPS

Correct-by-design control synthesis of MPL systems

Beyond model-based deduction
Stochastic hybrid (discrete/continuous) systems

- countable set of *discrete* modes
- uncountable *continuous* domains,
  with mode-dependent dynamics, transitions, resets
- *stochasticity* everywhere
Stochastic hybrid (discrete/continuous) systems

- discrete-time models

**finite-space Markov chain**

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

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

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

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

**uncountable-space Markov process**

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

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

\( \mathcal{T}_s(x|s) = \frac{\exp\left(-\frac{1}{2}(x-m(s))^T \Sigma^{-1}(s)(x-m(s))\right)}{\sqrt{2\pi}|\Sigma(s)|^{1/2}} \)

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

- discrete-time models

**finite-space Markov chain**

$$(\mathcal{Z}, \mathcal{T})$$

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

$$\mathcal{T} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}$$

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

**uncountable-space Markov process**

$$(S, T_s)$$

$$S = \mathbb{R}^2$$

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

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

⇒ discrete-time, stochastic hybrid systems
Stochastic hybrid (discrete/continuous) systems

Definition
A discrete-time stochastic hybrid system is a pair \((S, T_s)\),
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, \(n : Q \to \mathbb{N}\).
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 \(s = (q, x)\):

\[ T_s(ds' | s) \]
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 \(\pi : S \to [0, 1]\)
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** \(\pi : S \rightarrow [0, 1]\)

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

\[
T_s(ds' | s, u) = \begin{cases} 
T_x(dx'|(q, x), u) T_q(q|(q, x), u), & \text{if } q' = q \\
T_r(dx'|(q, x), u, q') T_q(q'|(q, x), u), & \text{if } q' \neq q
\end{cases}
\]

- **policy** \(\mu\): “string” of controls

- equivalent dynamical representation: \(s_{k+1} = f(s_k, \xi_k, u_k)\)
Stochastic hybrid systems in risk analysis

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

where \(\zeta_n, \xi_n\) i.i.d. random variables; \(g, h\) measurable; no control

where \(\zeta_n, \xi_n\) i.i.d. random variables; \(g, h\) measurable; no control
Stochastic hybrid systems in risk analysis

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

where \(\zeta_n, \xi_n\) i.i.d. random variables; \(g, h\) measurable; no control

- **objective:** what is the probability that, starting from initial capital \(Z_0 = x\), high capitalisation \(y\) is reached, while company’s bankruptcy is avoided
Stochastic hybrid systems in robotics and ATM

- **modes**: fly straight, vs. veer left/right, vs. ascend/descend
- **dynamics** are mode-dependent
- **uncertainty**: wind, weather forecast, “human in the loop”
Stochastic hybrid systems in robotics and ATM

- **modes**: fly straight, vs. veer left/right, vs. ascend/descend
- **dynamics** are mode-dependent
- **uncertainty**: wind, weather forecast, “human in the loop”

- **objective**: maximize probability that, starting within region $S$, robot/plane reaches set $U$, conditional on having passed over $T$, while avoiding set $A$
Stochastic hybrid systems in power networks

- **modes**: ON vs OFF ($m$)
- **continuous dynamics** over temperature ($\theta$)
- **uncertainty**: noise, weather ($w$)

**single TCL:**

$$\theta(t + 1) = a \theta(t) + (1 - a)(\theta_a - m(t) \cdot RP_{rate}) + w(t)$$

$$m(t + 1) = \begin{cases} 
0, & \theta(t)\langle \theta_s - \delta/2 \\
1, & \theta(t)\langle \theta_s + \delta/2 \\
m(t), & \text{else} 
\end{cases}$$
Stochastic hybrid systems in power networks

- **modes**: ON vs OFF ($m$)
- **continuous dynamics** over temperature ($\theta$)
- **uncertainty**: noise, weather ($w$)

**single TCL:**

$$\theta(t + 1) = a \theta(t) + (1 - a)(\theta_a - m(t) R P_{rate}) + w(t)$$

$$m(t + 1) = \begin{cases} 
0, & \theta(t) \langle \theta_s - \delta/2 \\
1, & \theta(t) \rangle \theta_s + \delta/2 \\
m(t), & \text{else}
\end{cases}$$

**population of TCLs:**

**objective**: conditional on meeting demand, prevent aggregate load to go beyond desired max for total power consumption
Outline

Cyber-physical systems

Models for CPS

Verification and control of SHS

Correct-by-design control synthesis of CPS

Correct-by-design control synthesis of MPL systems

Beyond model-based deduction
Analysis and control synthesis problems

reachability
(safety/invariance)

reach-avoid
(constrained reachability)

sequential reachability
(trajecory planning)

∞-horizon objectives
(invariance, persistence)
Analysis and control synthesis problems

reachability (safety/invariance)

reach-avoid (constrained reachability)

sequential reachability (trajectory planning)

∞-horizon objectives (invariance, persistence)

▶ how to express these properties via cost/value functions?
▶ ...as specifications in PCTL, LTL logics, or as automata
Formal abstractions for verification of complex models

<table>
<thead>
<tr>
<th>concrete complex model</th>
<th>property, specification, cost or reward</th>
</tr>
</thead>
</table>

if not, tune ϵ-spec holds yes/no policy µ → spec (correct by design)
Formal abstractions for verification of complex models

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

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

If not, tune policy \( \mu \rightarrow \epsilon \)-spec holds yes/no (correct by design)
Formal abstractions for verification of complex models

abstract simple model \(\epsilon\)-specification

\(\epsilon\)-quantitative abstraction

concrete complex model

property, specification, cost or reward

\(\epsilon\)-spec holds yes/no policy \(\mu\) \(\epsilon\rightarrow\epsilon\)-spec holds correct by design
Formal abstractions for verification of complex models

abstract simple model

$\epsilon$-specification

$\epsilon$-quantitative abstraction

concrete complex model

property, specification, cost or reward

automatic verification

control synthesis

if not, refine back

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

(correct by design)
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

concrete complex model

property, specification, cost or reward

model checking

automatic verification

control synthesis

$\epsilon$-spec holds yes/no policy $\mu_\epsilon \rightarrow \epsilon$-spec
Formal abstractions for verification of complex models

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

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

\( \epsilon \)-quantitative abstraction

refine back

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 $\mu_\epsilon \rightarrow \epsilon$-spec

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

concrete complex model

property, specification, cost or reward

refine back
Formal abstractions for verification of complex models

abstract simple model

abstract simple model

concrete complex model

property, specification, cost or reward

ε-quantitative abstraction

ε-specification

model checking

automatic verification

control synthesis

ε-spec holds yes/no policy

policy \( \mu_\epsilon \rightarrow \epsilon\text{-spec} \)

refine back

spec holds yes/no policy

policy \( \mu \rightarrow \text{spec} \) (correct by design)

if not, tune \( \epsilon \)
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 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

\[
S(\epsilon) = \{s \in S : \mathcal{P}_\pi(S) \geq \epsilon\} \quad \text{(here } \pi = \delta_s\text{)}
\]
Probabilistic safety/invariance: computation

- computation of $P_s(S)$ (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

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

$$V_0(s) = P_s(S) \Rightarrow S(\epsilon)$$
Probabilistic safety/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

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

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

$$V_k(s, u) : [0, N] \times S \times \mathcal{U} \to \mathbb{R}^+$$
Dynamical properties as temporal specifications in PCTL

- temporal specifications of finite-state systems expressed via modal logics
- e.g., PCTL used to express dynamical properties of finite-state dt-MC

Each PCTL formula over a SHS can be characterised via dynamic programming.
Dynamical properties as temporal specifications in PCTL

- temporal specifications of finite-state systems expressed via modal logics
- e.g., PCTL used to express dynamical properties of finite-state dt-MC

- PCTL extends CTL
- PCTL formulae are defined over states (via labels) or over paths
- if $\varphi$ is a path formula and $p \in [0, 1]$, then $\Psi = P_{\sim p}[\varphi]$ is a state formula
- example: $P_{\geq 0.5}[\Phi \cup_{\leq 10} \Psi]$
temporal specifications of finite-state systems expressed via modal logics
- e.g., PCTL used to express dynamical properties of finite-state dt-MC

- PCTL extends CTL
- PCTL formulae are defined over states (via labels) or over paths
  - if \( \varphi \) is a path formula and \( p \in [0, 1] \), then \( \Psi = P_{\sim p}[\varphi] \) is a state formula
  - example: \( P_{\geq 0.5}[\Phi \cup \leq 10 \Psi] \)

- each PCTL formula over a SHS can be characterised via dynamic programming
Dynamical properties as linear-time specifications (pLTL)

- generalisation to “richer” set of linear-time properties over dtSHS; specifications expressed as a DFAs or a Büchi automata
- characterisation via product automaton
- computation via probabilistic reachability over product
Dynamical properties as linear-time specifications (pLTL)

- generalisation to “richer” set of linear-time properties over dtSHS; specifications expressed as a DFA or a Büchi automata
- characterisation via product automaton
- computation via probabilistic reachability over product

- recall formal verification via model checking: automatic, model-based suite of algorithms, used with modal logics for PCTL:
  - input: dt-MC $(\mathcal{Z}, \mathcal{T})$, PCTL formula $\Psi$
  - output: $\text{Sat}(\Psi) = \{z \in \mathcal{Z} : z \models \Psi\}$
Model checking probabilistic invariance by abstractions

- model \((S, T_s)\), invariance set \(S \in S\), time horizon \(N\), safety level \(\epsilon\)
Model checking probabilistic invariance by abstractions

- model \((S, T_s)\), invariance set \(S \in S\), time horizon \(N\), safety level \(\epsilon\)

\(\delta\)-approximate \((S, T_s)\) with finite-state dt-MC \((Z, T)\)
compute related approximation error \(f(\delta, N)\)

- set \(S \rightarrow S_\delta\): label \(S_\delta\)-states in \(Z\) with formula \(\Phi_{S_\delta}\)
Model checking probabilistic invariance by abstractions

- model \((S, T_s)\), invariance set \(S \in S\), time horizon \(N\), safety level \(\epsilon\)

\[\delta\text{-approximate } (S, T_s) \text{ with finite-state dt-MC } (Z, T)\]
compute related approximation error \(f(\delta, N)\)

- set \(S \rightarrow S_\delta\): label \(S_\delta\)-states in \(Z\) with formula \(\Phi_{S_\delta}\)

\[\Rightarrow \text{probabilistic safe set}\]

\[S(\epsilon) = \{s \in S : \mathcal{P}_s(S) \geq \epsilon\}\]
\[= \{s \in S : (1 - \mathcal{P}_s(S)) \leq 1 - \epsilon\}\]
Model checking probabilistic invariance by abstractions

- model \((S, T_s)\), invariance set \(S \in S\), time horizon \(N\), safety level \(\epsilon\)

\(\delta\)-approximate \((S, T_s)\) with finite-state dt-MC \((Z, T)\)
compute related approximation error \(f(\delta, N)\)

- set \(S \rightarrow S_\delta\): label \(S_\delta\)-states in \(Z\) with formula \(\Phi_{S_\delta}\)

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

can be related to

\[
\begin{align*}
Z_\delta(\epsilon + f(\delta, N)) &= \operatorname{Sat}\left(\mathbb{P}_{\leq 1-\epsilon-f(\delta, N)}\left(\text{true } U^{\leq N} \neg \Phi_{S_\delta}\right)\right) \\
&= \{z \in Z : z \models \mathbb{P}_{\leq 1-\epsilon-f(\delta, N)}\left(\text{true } U^{\leq N} \neg \Phi_{S_\delta}\right)\}
\end{align*}
\]
Abstraction algorithm

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

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

- algorithm:

  \begin{itemize}
  \item \textbf{input:} dt-MP \((S, T_s)\)
  \item 1 select finite partition \(S = \bigcup_{i=1}^{p} S_i\)
  \item 2 and representative points \(z_i \in S_i\)
  \item 3 define finite state space \(\mathcal{Z} := \{z_i, i = 1, \ldots, p\}\)
  \item 4 and transition probability matrix: \(T(z_i, z_j) = T_s(S_j | z_i)\)
  \end{itemize}

  \textbf{output:} dt-MC \((\mathcal{Z}, T)\)
Abstraction algorithm

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

- algorithm:

  \begin{itemize}
  \item \textbf{input:} dt-MP \((S, T_s)\)
  \item 1 select finite partition \(S = \bigcup_{i=1}^{p} S_i\)
  \item 2 and representative points \(z_i \in S_i\)
  \item 3 define finite state space \(\mathcal{Z} := \{z_i, i = 1, \ldots, p\}\)
  \item 4 and transition probability matrix: \(T(z_i, z_j) = T_s(S_j \mid z_i)\)
  \item \textbf{output:} dt-MC \((\mathcal{Z}, T)\)
  \end{itemize}
Abstraction algorithm

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

- algorithm:

  ```
  input: dt-MP \((S, T_s)\)
  1 select finite partition \(S = \bigcup_{i=1}^{p} S_i\)
  2 and representative points \(z_i \in S_i\)
  3 define finite state space \(\mathcal{Z} := \{z_i, i = 1, \ldots, p\}\)
  4 and transition probability matrix: \(T(z_i, z_j) = T_s(S_j | z_i)\)
  output: dt-MC \((\mathcal{Z}, T)\)
  ```
Abstraction algorithm

- approximate dt-MP \((S, T_s)\) as dt-MC \((Z, T)\), where
  - \(Z = \{z_1, z_2, \ldots, z_p\}\) – finite set of abstract states
  - \(T : Z \times Z \to [0, 1]\) – transition probability matrix

algorithm:

<table>
<thead>
<tr>
<th>input:</th>
<th>dt-MP ((S, T_s))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>select finite partition (S = \bigcup_{i=1}^{p} S_i)</td>
</tr>
<tr>
<td>2</td>
<td>and representative points (z_i \in S_i)</td>
</tr>
<tr>
<td>3</td>
<td>define finite state space (Z := {z_i, i = 1, \ldots, p})</td>
</tr>
<tr>
<td>4</td>
<td>and transition probability matrix: (T(z_i, z_j) = T_s(S_j</td>
</tr>
</tbody>
</table>

output: dt-MC \((Z, T)\)
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 $z^i \in S_\delta$, $\forall s : s \land z^i \in S^i$, error is

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

$$\tilde{\delta}_i = \text{diam} (S^i), \quad \delta = \max_{i=1,\ldots,m} \delta_i$$
On the approximation error \( f(\delta, N) \)

- approximation via \( \delta \)-partitioning: \( S = \bigcup_{i=1,...,m} S^i \)

- under Lip-continuity assumptions on density of kernel \( T_s \),
  \[
  h(i,j), \quad i,j = 1,...,m
  \]

- for any \( z^i \in S_\delta \), \( \forall s : s \land z^i \in S^i \), error is
  \[
  f(\delta, N) \doteq |P_s(S) - P_{z^i}(S_\delta)| \leq \max_{i=1,...,m} N\delta_i \sum_{j=1,...,m} h(i,j),
  \]
  \[
  \delta_i = \text{diam } (S^i), \quad \delta = \max_{i=1,...,m} \delta_i
  \]

error is linear in \( N, \delta_i \) and depends on local constants \( h(i,j) \rightarrow \text{local tuning} \)
On the approximation error $f(\delta, N)$

- error generalization
  - discontinuous and partially degenerate kernels
  - ill-conditioned kernels (different time scales, as in biology)
  - structured dynamics (sparse variable coupling, linear pathways)

- error refinement by higher-order approximations
  - $\delta$: faster convergence upon tuning
  - $N$: possibly bounded in time (allows considering $\infty$-horizon properties)
On the approximation error $f(\delta, N)$

- error generalization
  - discontinuous and partially degenerate kernels
  - ill-conditioned kernels (different time scales, as in biology)
  - structured dynamics (sparse variable coupling, linear pathways)

- error refinement by higher-order approximations
  - $\delta$: faster convergence upon tuning
  - $N$: possibly bounded in time (allows considering $\infty$-horizon properties)

- practical implementation:
On the approximation error $f(\delta, N)$

- error generalization
  - discontinuous and partially degenerate kernels
  - ill-conditioned kernels (different time scales, as in biology)
  - structured dynamics (sparse variable coupling, linear pathways)

- error refinement by higher-order approximations
  - $\delta$: faster convergence upon tuning
  - $N$: possibly bounded in time (allows considering $\infty$-horizon properties)

- practical implementation:
On the approximation error $f(\delta, N)$

- error generalization
  - discontinuous and partially degenerate kernels
  - ill-conditioned kernels (different time scales, as in biology)
  - structured dynamics (sparse variable coupling, linear pathways)

- error refinement by higher-order approximations
  - $\delta$: faster convergence upon tuning
  - $N$: possibly bounded in time (allows considering $\infty$-horizon properties)

- practical implementation:
On the approximation error $f(\delta, N)$

- error generalization
  - discontinuous and partially degenerate kernels
  - ill-conditioned kernels (different time scales, as in biology)
  - structured dynamics (sparse variable coupling, linear pathways)

- error refinement by higher-order approximations
  - $\delta$: faster convergence upon tuning
  - $N$: possibly bounded in time (allows considering $\infty$-horizon properties)

- practical implementation:
On the approximation error $f(\delta, N)$

- error generalization
  - discontinuous and partially degenerate kernels
  - ill-conditioned kernels (different time scales, as in biology)
  - structured dynamics (sparse variable coupling, linear pathways)

- error refinement by higher-order approximations
  - $\delta$: faster convergence upon tuning
  - $N$: possibly bounded in time (allows considering $\infty$-horizon properties)

- practical implementation:
On the approximation error \( f(\delta, N) \)

- error generalization
  - discontinuous and partially degenerate kernels
  - ill-conditioned kernels (different time scales, as in biology)
  - structured dynamics (sparse variable coupling, linear pathways)

- error refinement by higher-order approximations
  - \( \delta \): faster convergence upon tuning
  - \( N \): possibly bounded in time (allows considering \( \infty \)-horizon properties)

- practical implementation:
Software for formal abstractions

- FAUST$^2$: software for **sequential**, **adaptive** quantitative abstractions
- from MATLAB/Simulink model to MRMC/PRISM input
- scales to larger models than alternative grid-based techniques

http://sourceforge.net/projects/faust2
Approximate probabilistic bisimulations

\[
T_z = \begin{bmatrix}
0.6 & 0.3 & 0.1 \\
0.4 & 0.1 & 0.5 \\
0.6 & 0.3 & 0.1 \\
\end{bmatrix}
\]

- \((\tilde{S}, \tilde{T}_z)\) is a “lumped” version of \((S, T_z)\)
- exact probabilistic bisimulation as model abstraction

\[
\tilde{T}_z = \begin{bmatrix}
0.7 & 0.3 \\
0.9 & 0.1 \\
\end{bmatrix}
\]

- now consider \(\tilde{T}_z = \begin{bmatrix}
0.6 + \delta_1 & 0.3 - \delta_1 & 0.1 \\
0.4 & 0.1 & 0.5 \\
0.6 + \delta_2 & 0.3 - \delta_2 & 0.1 \\
\end{bmatrix}\)

\(\Rightarrow\) approximate probabilistic bisimulation
Demand response in power networks

- consider dynamical behaviour of temperature of single TCL, controlled via temperature setpoint

⇒ provide aggregate model for large-dimensional population of $n_p$ TCL
Demand response in power networks

▶ consider dynamical behaviour of temperature of single TCL, controlled via temperature setpoint

⇒ provide aggregate model for large-dimensional population of $n_p$ TCL

▶ control total power consumption (TCL in ON mode) via temperature setpoint:

$$y_{total}(t) = \sum_{i=1}^{n_p} \frac{1}{\eta} P_i m_i(t)$$

⇒ attain demand response while users are oblivious to setpoint change
Aggregate TCL population model in the literature

- input/output aggregate model
- goal: power regulation via (small) modifications in the set-point, \( u = f(\theta_s) \) \( \Rightarrow \) changes in the percentage of TCL in ON mode (and in the total power consumption)
- noisy LTI model (input: set-point, output: power):
  \[
  A(q)y(t) = B(q)u(t) + C(q)e(t)
  \]
- \( A(\cdot), B(\cdot), C(\cdot) \) are polynomials, with coefficients estimated from data
- works under homogeneity assumption
- there is no precise relationship btw model and population parameters
Aggregate TCL population model in the literature

- aggregation based on state-space partition
  1. divide TCL dead band into bins
  2. introduce population state $X$ (TCL portion with temperature in specific bin)
  3. build dynamics $X(t + 1) = AX(t) + Bu(t)$
     (A depends on TCL temperature dynamics over bins)
     ($u$ affects portion of TCL in each bin via $B$)
Aggregate TCL population model in the literature

- aggregation based on state-space partition
  1. divide TCL dead band into bins
  2. introduce population state $X$ (TCL portion with temperature in specific bin)
  3. build dynamics $X(t + 1) = AX(t) + Bu(t)$
     (A depends on TCL temperature dynamics over bins)
     ($u$ affects portion of TCL in each bin via $B$)

- aggregation procedure leading to $X$
  1. works under assumptions on TCL dynamics (e.g., is deterministic)
  2. dependence of model precision on number of bins not easily controllable
Aggregate TCL population model via abstractions

1. approximate dynamics of single TCL (dtSHS) by Markov chain (dtMC)
   ▶ induce approximate bisimulation by state-space binning
2. consider population of TCL by taking cross product of Markov chains
3. define labels of product Markov chain to be the number of TCL within an abstract state (bin)
4. construct exact bisimulation of product Markov chain based on labels, obtain dynamics of aggregated model
Aggregate TCL population model via abstractions

1. approximate dynamics of single TCL (dtSHS) by Markov chain (dtMC)
   ▶ induce approximate bisimulation by state-space binning
2. consider population of TCL by taking cross product of Markov chains
3. define labels of product Markov chain to be the number of TCL within an abstract state (bin)
4. construct exact bisimulation of product Markov chain based on labels, obtain dynamics of aggregated model

★ quantify error (on total output power) introduced by the abstraction
  ▶ approximate bisimulation
  ▶ (exact bisimulation → no error)
Aggregate TCL population model via abstractions

- given the parameters of the TCL population, the aggregate model is

\[ X(t + 1) = FX(t) + W(t), \]

where \( F = P^T \), and \( W \) is explicitly characterised from TCL dynamics

- for a homogeneous population

  \( \bar{Y}(t) \) – total power consumption of the population
  \( Y(t) \) – power consumption of the aggregated model

- for any distribution of the initial state of the population,

\[
\left| \mathbb{E}[Y(t) - \bar{Y}(t)] \right| \leq n_p Pt \left[ t\epsilon + \frac{a}{\sigma \sqrt{2\pi}} \delta_p \right],
\]

where \( \epsilon = \frac{e^{-\alpha^2/2}}{\alpha \sqrt{2\pi}} \), and \( \alpha \) is an affine function of the length of the partition
Aggregate TCL population model via abstractions

- average of 50 Monte Carlo simulations for a population of $n_p = 10^3$ TCL
- all TCL initialised in OFF mode and at set point $\theta_s$
- comparison: “deterministic” vs formal abstractions
- (top plot: $\sigma = 0.003$; bottom plot: $\sigma = 0.03$)
Aggregate TCL population model via abstractions

- average of 50 Monte Carlo simulations for a population of \( n_p = 10^3 \) TCL
- with \( \sigma = 0.03 \), heterogeneous population
- comparison: “deterministic” vs formal abstractions
- (top plot: \( C \in [8, 12] \); bottom plot: \( C \in [2, 18] \) )
Aggregate TCL population model via abstractions

- centralised control of population of TCL model via setpoint changes
  1. perform state estimation from total power consumption
  2. devise closed-loop control scheme for signal tracking
- again error of procedure can be tuned

\[
\begin{align*}
\text{Population: } & \quad \text{TCL}_1(\theta_1, m_1) \\
& \quad \text{TCL}_2(\theta_2, m_2) \\
& \quad \vdots \\
& \quad \text{TCL}_n_p(\theta_{n_p}, m_{n_p}) \\
\end{align*}
\]

\[
\begin{align*}
\theta_s(t) & \quad \rightarrow \quad y_{\text{meas}}(t + 1) \\
\text{Population} & \quad \rightarrow \quad \hat{X}(0) \\
\text{parameters} & \quad \rightarrow \quad \hat{X}(t + 1) \\
\end{align*}
\]

**Conditional Kalman filter with state-dependent process noise**

\[
\begin{align*}
\dot{X}^-(t + 1) & = F_{\sigma(t)} \hat{X}(t) \\
P^-(t + 1) & = F_{\sigma(t)} P(t) F_{\sigma(t)}^T + \Sigma(\hat{X}(t)) \\
K_{t+1} & = P^-(t + 1) H^T [H P^-(t + 1) H^T + R_e]^{-1} \\
\hat{P}(t+1) & = [I - K_{t+1} H] P^-(t + 1) \\
\hat{X}(t + 1) & = \hat{X}^-(t + 1) + K_{t+1} [y_{\text{meas}}(t + 1) - H \hat{X}^-(t + 1)] \\
\end{align*}
\]

**One-step regulation**

\[
\begin{align*}
\min_{\sigma(t+1)\in\mathbb{Z}_n} & \quad |y_{\text{ext}}(t + 2) - y_{\text{des}}(t + 2)| \\
\text{subject to:} & \quad \dot{X}(t + 2) = F_{\sigma(t+1)} \dot{X}(t + 1) \\
& \quad y_{\text{ext}}(t + 2) = H \hat{X}(t + 2) \\
\end{align*}
\]

\[
\begin{align*}
\theta_s(t + 1) & \quad \rightarrow \quad z^{-1} \\
\end{align*}
\]

\[
\begin{align*}
\theta_s(t + 1) & \quad \rightarrow \quad y_{\text{des}}(t + 2) \\
\end{align*}
\]
Aggregate TCL population model via abstractions

- centralised control of population of TCL model via setpoint changes
  1. perform state estimation from total power consumption
  2. devise closed-loop control scheme for signal tracking
- again error of procedure can be tuned

- tracking of a piece-wise constant reference signal via set-point, by one-step regulation
Aggregate TCL population model via abstractions

- centralised control of population of TCL model via setpoint changes
  1. perform state estimation from total power consumption
  2. devise closed-loop control scheme for signal tracking
- again error of procedure can be tuned

- tracking of a piece-wise constant reference signal via set-point, by one-step regulation
Aggregate TCL population model via abstractions

- centralised control of population of TCL model via setpoint changes
  1. perform state estimation from total power consumption
  2. devise closed-loop control scheme for signal tracking

- again error of procedure can be tuned

- tracking of a piece-wise constant reference signal via set-point, by SMPC scheme (with constraints)
In collaboration with Honeywell labs Prague (CZ), Trend Control (UK), Google Nest Labs (US)

- development of innovative models for smart buildings
- use of verification and control synthesis techniques for certifiable energy behaviour
  1. temperature, humidity, $CO_2$ control
  2. predictive device maintenance
Outline

Cyber-physical systems

Models for CPS

Verification and control of SHS

Correct-by-design control synthesis of CPS

Correct-by-design control synthesis of MPL systems

Beyond model-based deduction
Verification of CPS

discrete abstraction

property

Proving

Model Checking

Abstraction

$\frac{d\xi}{dt} = f(\xi, v)$

continuous dynamics

hardware + software

Hybrid dynamics
Correct-by-design synthesis of CPS

\[ q(j + 1) = g(q(j), \xi(t)) \]
\[ v(t) = k(\xi(t), q(j)) \]

Abstraction

Refinement

Hybrid dynamics

Continuous dynamics

Discrete abstraction

Discrete controller

Hardware + software
Correct-by-design synthesis of CPS (alternative view)

\[
d\xi/dt = f(\xi, v)
\]

**continuous dynamics**

**hybrid controller**

\[
q(j + 1) = g(q(j), \xi(t))\\v(t) = k(\xi(t), q(j))
\]

**discrete controller**

**Abstraction**

**Refinement**
Outline

Cyber-physical systems

Models for CPS

Verification and control of SHS

Correct-by-design control synthesis of CPS

Correct-by-design control synthesis of MPL systems

Beyond model-based deduction
Introduction to MPL systems

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

\[ x(k) \] is the time of \( k \)-th event, \( k \in \mathbb{N} \cup \{0\} \)
timing updates: maximization (⊕) and addition (⊗) operations
→ max-plus algebra
\[ \epsilon = -\infty, \quad \mathbb{R}_\epsilon = \mathbb{R} \cup \{\epsilon\}, \quad \alpha, \beta \in \mathbb{R}_\epsilon \]
\[ \alpha \oplus \beta := \max(\alpha, \beta), \quad \alpha \otimes \beta := \alpha + \beta, \quad \text{and matrix operations} \]
Max-plus-linear models

Definition (Autonomous MPL model)

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

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

Example

A simple railway model

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

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

Definition (Non-autonomous MPL model)

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

where \( B \in \mathbb{R}^{n \times m} \) and \( u \in \mathbb{R}^m \) (synthesis = scheduling)
Classical analysis of MPL models

- study of transient and periodic regimes, of asymptotics
- classical analysis based on algebraic or geometric properties
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 $\lambda = 4$):
   
   $$
   \begin{bmatrix}
   1 \\
   0
   \end{bmatrix},
   \begin{bmatrix}
   5 \\
   4
   \end{bmatrix},
   \begin{bmatrix}
   9 \\
   8
   \end{bmatrix},
   \begin{bmatrix}
   13 \\
   12
   \end{bmatrix},
   \begin{bmatrix}
   17 \\
   16
   \end{bmatrix},
   \begin{bmatrix}
   21 \\
   20
   \end{bmatrix},
   \begin{bmatrix}
   25 \\
   24
   \end{bmatrix},
   \begin{bmatrix}
   29 \\
   28
   \end{bmatrix},
   \begin{bmatrix}
   33 \\
   32
   \end{bmatrix},
   \begin{bmatrix}
   37 \\
   36
   \end{bmatrix},
   \begin{bmatrix}
   41 \\
   40
   \end{bmatrix},
   \begin{bmatrix}
   45 \\
   44
   \end{bmatrix},
   $$

2. periodic regime with period $c = 2$ (transient $k_0 = 3$):
   
   $$
   \begin{bmatrix}
   4 \\
   0
   \end{bmatrix},
   \begin{bmatrix}
   6 \\
   7
   \end{bmatrix},
   \begin{bmatrix}
   12 \\
   10
   \end{bmatrix},
   \begin{bmatrix}
   15 \\
   15
   \end{bmatrix},
   \begin{bmatrix}
   20 \\
   18
   \end{bmatrix},
   \begin{bmatrix}
   23 \\
   23
   \end{bmatrix},
   \begin{bmatrix}
   28 \\
   26
   \end{bmatrix},
   \begin{bmatrix}
   31 \\
   31
   \end{bmatrix},
   \begin{bmatrix}
   36 \\
   34
   \end{bmatrix},
   \begin{bmatrix}
   39 \\
   39
   \end{bmatrix},
   \begin{bmatrix}
   41 \\
   42
   \end{bmatrix},
   \begin{bmatrix}
   47 \\
   47
   \end{bmatrix},
   $$
Labeled transition system (LTS)

- set of states \( S = \{1, 2, 3, 4\} \)
- set of inputs \( \text{Act} = \{\alpha, \beta\} \)
- transitions \( \rightarrow = \{(1, \alpha, 4), (4, \alpha, 3), \ldots\} \)
- set of outputs \( \text{AP} = \{a, b\} \) 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

- procedure: need to compute states, transitions, labels
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 $\simeq_{i,j} \in \{\langle, \leq\}$, $\alpha_{i,j} \in \mathbb{R} \cup \{+\infty\}$, 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
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

Example

- determinism vs non-determinism of obtained TS
LTS transitions, an example

Example

- determinism vs non-determinism of obtained TS
Relationship between LTS and MPL

Theorem

- *TS simulates the original MPL model*
- *TS bisimulates the MPL model if and only if it is deterministic*
- *every irreducible MPL model admits finite deterministic TS abstraction*

- non-deterministic TS can be “determinized” by refining partitioning regions
- termination of refinement procedure does not hold in general
LTS labels

Definition

- **state labels:**
  all possible values of $x_i(k) - x_j(k)$, for $1 \leq i \not\equiv j \leq n$
  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

- 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

http://sourceforge.net/projects/verisimpl
Example

- automatically find MPL eigenspace: \( \bigvee_{\phi \in L=AP} (\square \phi \land |\phi| = 0) \)
Example

- automatically find MPL periodic regime: \[ \bigvee_{\varphi \in L} AP \square (\varphi \land \square^c \varphi) \]
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 \times 10^3</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 \times 10^4</td>
</tr>
<tr>
<td>13</td>
<td>3.5 [m]</td>
<td>5.5 [m]</td>
<td>2.8 [m]</td>
<td>1.9 \times 10^3</td>
<td>1.9 \times 10^5</td>
</tr>
<tr>
<td>15</td>
<td>53.6 [m]</td>
<td>2.0 [h]</td>
<td>39.4 [m]</td>
<td>7.4 \times 10^3</td>
<td>2.0 \times 10^6</td>
</tr>
</tbody>
</table>

- bottleneck: generation of transitions
Computational benchmark for abstraction

- 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 · 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>
Outline

Cyber-physical systems

Models for CPS

Verification and control of SHS

Correct-by-design control synthesis of CPS

Correct-by-design control synthesis of MPL systems

Beyond model-based deduction
Integrating data-driven and model-based techniques

- throughout this week, we have looked at model-based deduction techniques
- such techniques are oblivious to the underlying system data
- induction should be brought back into play

- an AIMS project is being proposed on this topic
Integrating data-driven modelling and verification

- consider measurement setup above
- verify whether \( S \models \psi \) with confidence \( \delta \)
- assume \( S \in \{ M(\theta) : \theta \in \Theta \} \)
- **verification step**: determine property- and model-based feasible set \( \Theta_\psi \), so that \( \forall \theta \in \Theta_\psi, M(\theta) \models \psi \)
- step above can be performed under assumptions on \( M(\theta), \psi \)
- **identification step**: leverage classical data-based Bayesian inference
  1. parametric prior \( p(\theta) \)
  2. posterior \( p(\theta | \tilde{y}) \) obtained via inference step as \( p(\theta | \tilde{y}) = \frac{p(\tilde{y} | \theta)}{\int_\Theta p(\tilde{y} | \theta) p(\theta) d\theta} \)
- **confidence quantification**: \( \delta = P(\Theta_\psi | \tilde{y}) = \int_{\Theta_\psi} p(\theta | \tilde{y}) d\theta \)
Integrating data-driven modelling and verification

- start from confidence quantification: \( \delta = P(\Theta_\psi | \tilde{y}) = \int_{\Theta_\psi} p(\theta | \tilde{y}) d\theta \)

- recall we want to establish whether \( S \models \psi \),
  two decisions:
  
<table>
<thead>
<tr>
<th>decision</th>
<th>confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>( P(\Theta_\psi</td>
</tr>
<tr>
<td>no</td>
<td>( 1 - P(\Theta_\psi</td>
</tr>
</tbody>
</table>

- devise optimal experiment via

  \[
  \sup_u \int_{\tilde{y}} \max\{ P(\Theta_\psi | \tilde{y}, u), 1 - P(\Theta_\psi | \tilde{y}, u) \} p(\tilde{y} | u) d\tilde{y}
  \]

- on resulting \( \arg \sup_u \), iterate within identification step above

- obtain confidence \( \delta' \geq \delta \)
Summary

- stochastic transitions
- general Markov processes
- uncountable state spaces
  - abstraction based verification & controller synthesis
  - partition uncountable state space
  - adaptive methods for scalability

- model uncertainty
- integrated modelling and verification
  - design “smart” experiments
  - data-driven inductive step
  - model-based deductive step
Outline

Cyber-physical systems

Models for CPS

Verification and control of SHS

Correct-by-design control synthesis of CPS

Correct-by-design control synthesis of MPL systems

Beyond model-based deduction