Formal verification of complex systems: model-based and data-driven methods

Alessandro Abate

Department of Computer Science, University of Oxford

MEMOCODE - Sept 30, 2017

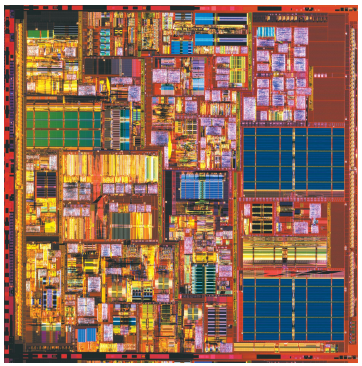
Automated formal verification: successes and frontiers

- automated, sound, formal

Automated formal verification: successes and frontiers

- automated, sound, formal
- industrial impact in verification of

protocols, hardware circuits, and software

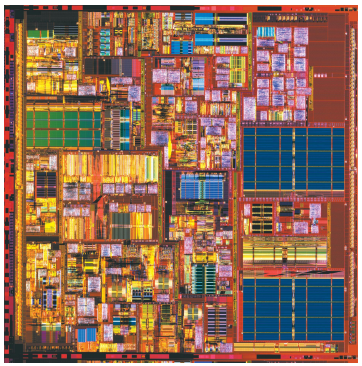


```c
/* This line basically imports the "stdio" header file, part of
 * the standard library. It provides input and output functionality
 * to the program.
 */
#include <stdio.h>

/*
 * Function (method) declaration. This outputs "Hello, world" to
 * standard output when invoked.
 */
void sayHello() {
    // printf() in C outputs the specified text (with optional
    // formatting options) when invoked.
    printf("Hello, world!");
}

/*
 * This is a "main function". The compiled program will run the code
 * defined here.
 */
void main() {
    // Invoke the sayHello() function.
    sayHello();
}
```

Automated formal verification: successes and frontiers

- automated, sound, formal
- industrial impact in verification of

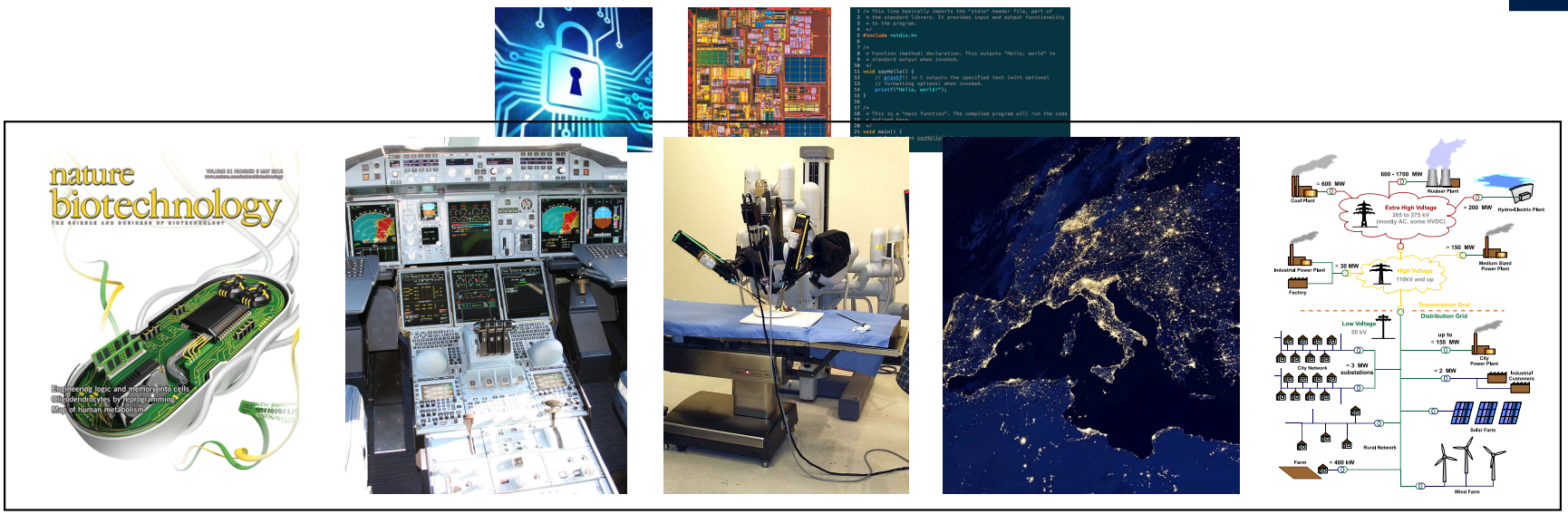
protocols, hardware circuits, and software



```c
/* This line basically imports the "stdio" header file, part of
 * the standard library. It provides input and output functionality
 * to the program.
 */
#include <stdio.h>

/*
 * Function (method) declaration. This outputs "Hello, world" to
 * standard output when invoked.
 */
void sayHello() {
    // printf() in C outputs the specified text (with optional
    // formatting options) when invoked.
    printf("Hello, world!");
}

/*
 * This is a "main function". The compiled program will run the code
 * defined here.
 */
void main() {
    // Invoke the sayHello() function.
    sayHello();
}
```

- asserts properties over given model of a system
- scalable and useful on unsophisticated models

Automated formal verification: pushing the envelope



1. verification of physical systems (cyber-physical systems)
   - bridging the gap between data and models
   - principled integration of learning and verification

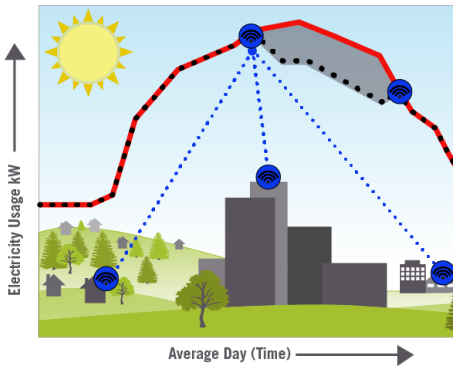
2. verification of complex models
   - dynamical models with uncertainty, noise (for CPS)
   - via formal abstractions

Building automation systems: an exemplar of CPS

- cyber-physical systems: integration of physical/analogue with cyber/digital
- building automation systems as a CPS exemplar

Building automation systems: an exemplar of CPS

- cyber-physical systems: integration of physical/analogue with cyber/digital
- building automation systems as a CPS exemplar



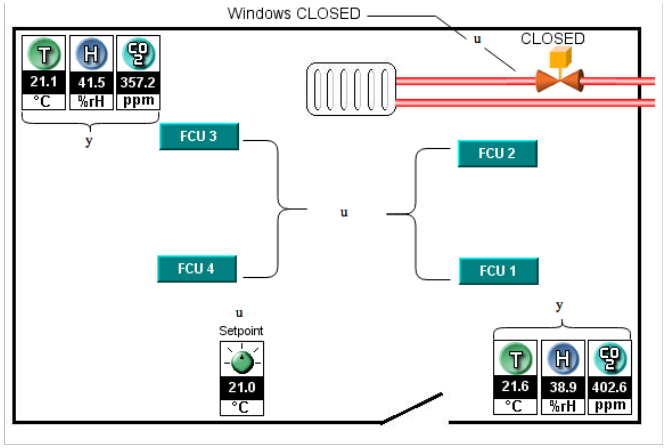
- *smart energy* initiatives at Oxford CS

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, CO_2
  2. model-based predictive maintenance of devices
  3. fault-tolerant certified control
  4. demand-response over smart grids

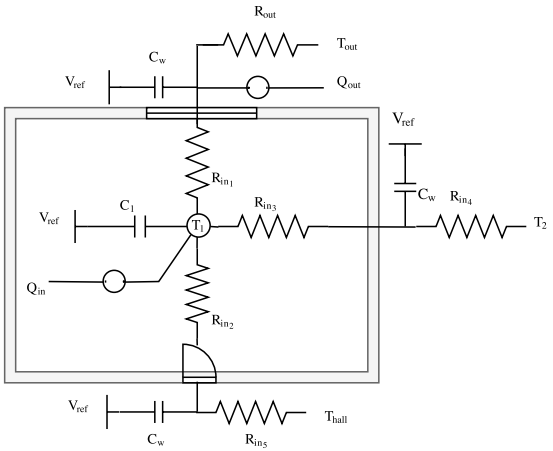
Building automation systems - a CPS exemplar



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

- advanced modelling for smart buildings
- applications: certifiable energy management
  1. control of temperature, humidity, CO_2
  2. model-based predictive maintenance of devices
  3. fault-tolerant certified control
  4. demand-response over smart grids

Building automation systems - a CPS exemplar



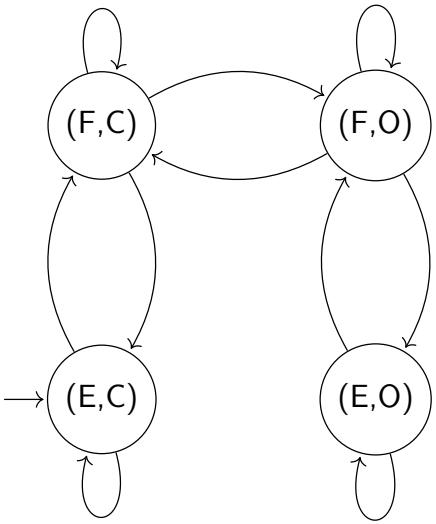
Building automation system setup in rooms 478/9 at Oxford CS

- advanced modelling for smart buildings
- applications: certifiable energy management
  1. control of temperature, humidity, CO_2
  2. model-based predictive maintenance of devices
  3. fault-tolerant certified control
  4. demand-response over smart grids

Building automation systems - problem setup

- model CO_2 dynamics, under the effect of
  1. occupants: room full (F)/empty (E)
  2. window: open (O)/closed (C)
  3. air circulation: ON/OFF

$$x_{k+1} = x_k + \frac{\Delta}{V}\left(-\mathbb{1}_{ON} m x_k + \mu_{\{O,C\}}(C_{out} - x_k)\right) + \mathbb{1}_F C_{occ}$$

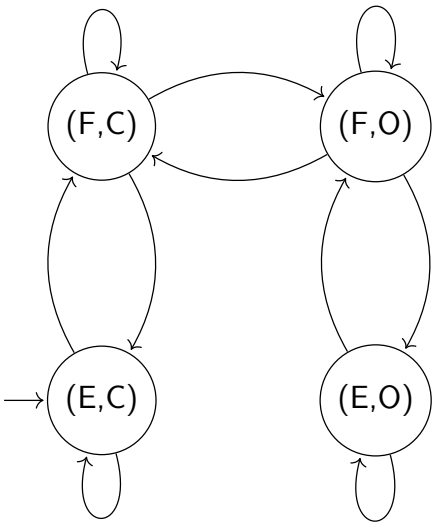


- x - zone CO_2 level
- Δ - sampling time
- V - zone volume
- m - air inflow (when ON)
- μ_O - air exchange with outside (when O)
- μ_C - air leakage with outside (when C)
- C_{out} - outside CO_2 level
- C_{occ} - CO_2 by occupants (when F)

Building automation systems - problem setup

- model CO_2 dynamics, under the effect of
  1. occupants: room full (F)/empty (E)
  2. window: open (O)/closed (C)
  3. air circulation: ON/OFF

$$x_{k+1} = x_k + \frac{\Delta}{V}\left(-\mathbb{1}_{ON} m x_k + \mu_{\{O,C\}}(C_{out} - x_k)\right) + \mathbb{1}_F C_{occ}$$

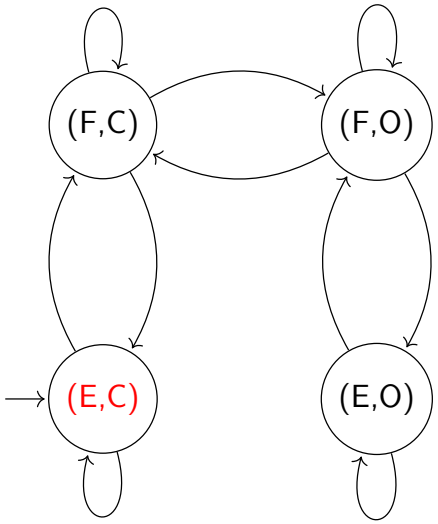


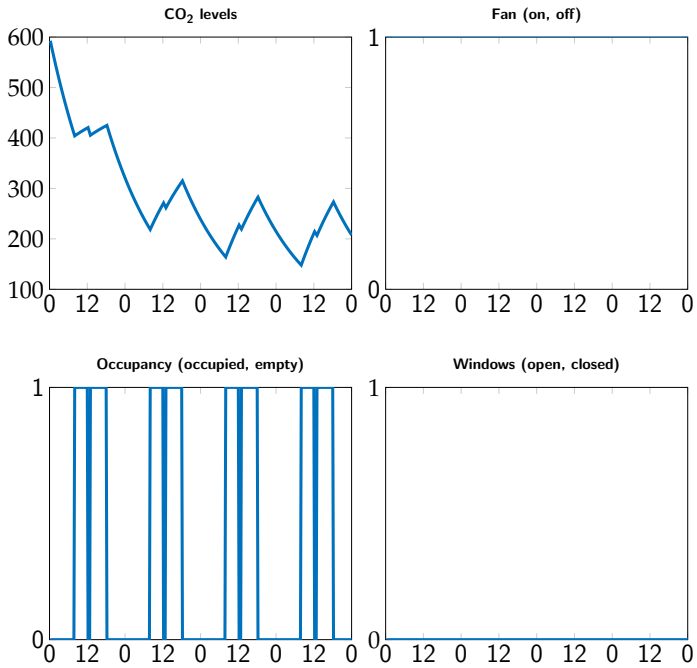
Parameter	Value
Δ	15 min
V	288 m^3
m	0.25 m^3/min
μ_O	0.1667 m^3/min
μ_C	0.01 m^3/min
C_{out}	375 ppm
C_{occ}	0.4 ppm/min

Building automation systems - problem setup

- model CO_2 dynamics, under the effect of
  1. occupants: room empty E
  2. window: closed C
  3. air circulation: ON

$$x_{k+1} = x_k + \frac{\Delta}{V}\left(-m x_k + \mu_C (C_{out} - x_k)\right) + 0 \cdot C_{occ}$$

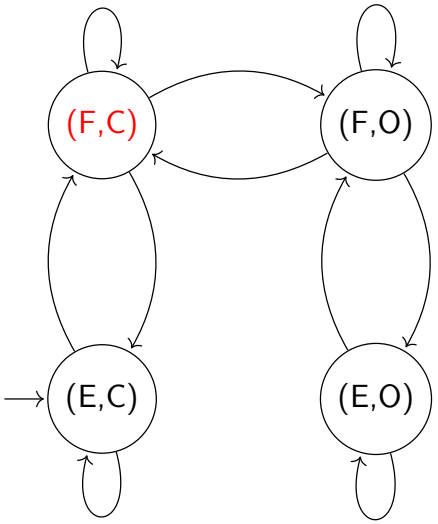


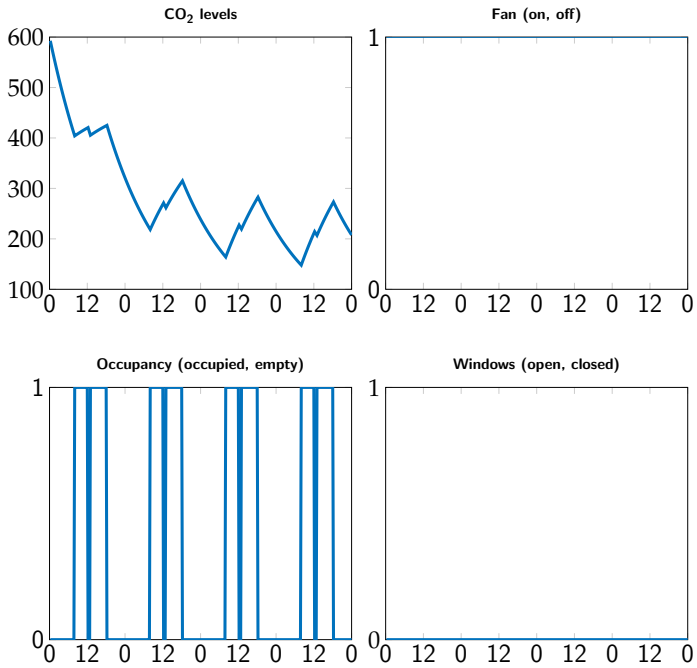


Building automation systems - problem setup

- model CO_2 dynamics, under the effect of
  1. occupants: room full F
  2. window: closed C
  3. air circulation: ON

$$x_{k+1} = x_k + \frac{\Delta}{V}\left(-m x_k + \mu_C (C_{out} - x_k)\right) + C_{occ}$$

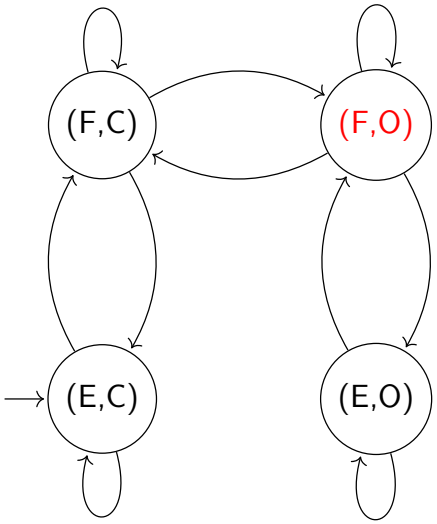


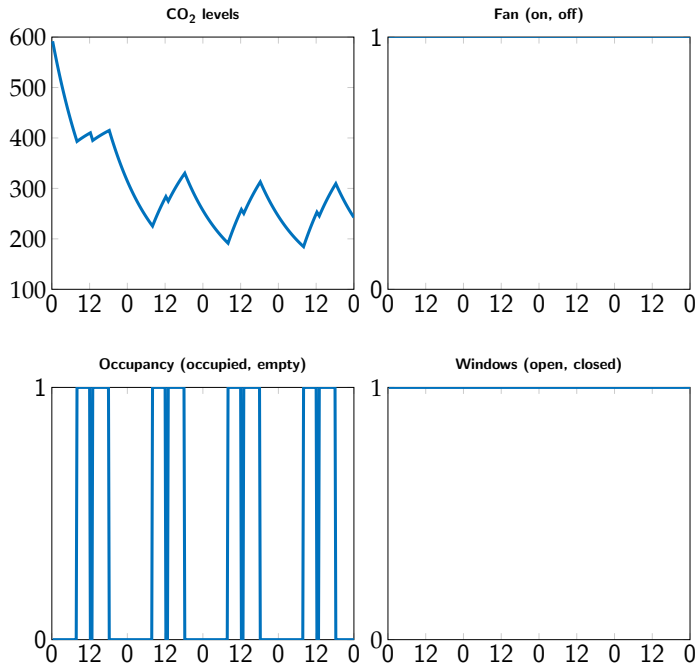


Building automation systems - problem setup

- model CO_2 dynamics, under the effect of
  1. occupants: room full F
  2. window: open O
  3. air circulation: ON

$$x_{k+1} = x_k + \frac{\Delta}{V}\left(-m x_k + \mu_O (C_{out} - x_k)\right) + C_{occ}$$

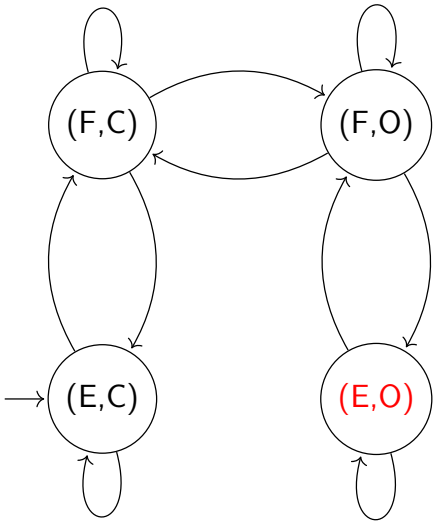


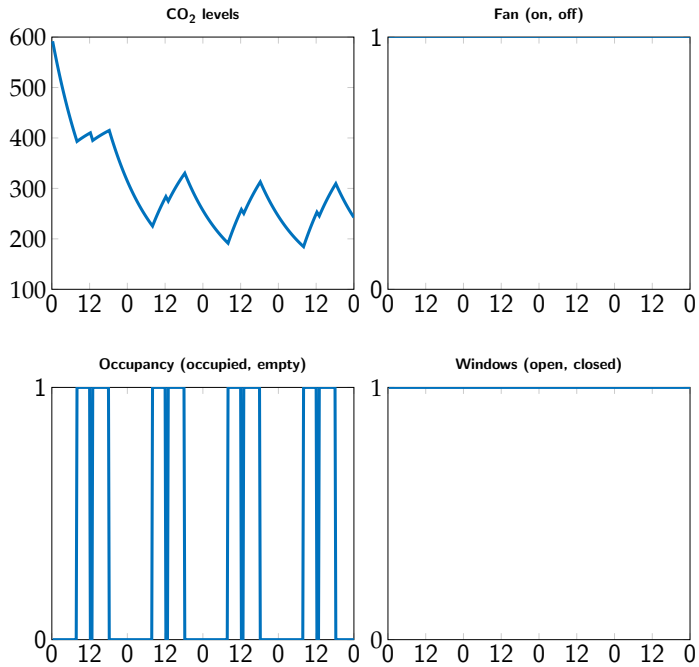


Building automation systems - problem setup

- model CO_2 dynamics, under the effect of
  1. occupants: room empty E
  2. window: closed C
  3. air circulation: ON

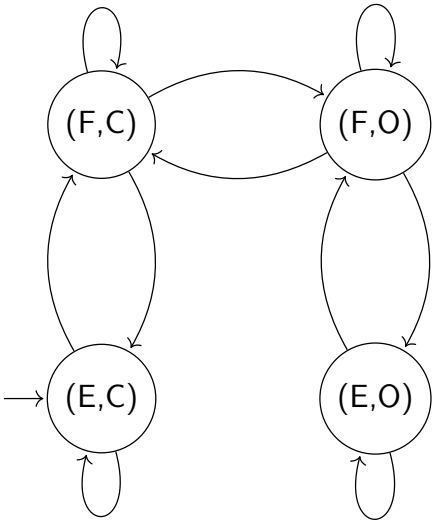
$$x_{k+1} = x_k + \frac{\Delta}{V}\left(-m x_k + \mu_O (C_{out} - x_k)\right)$$



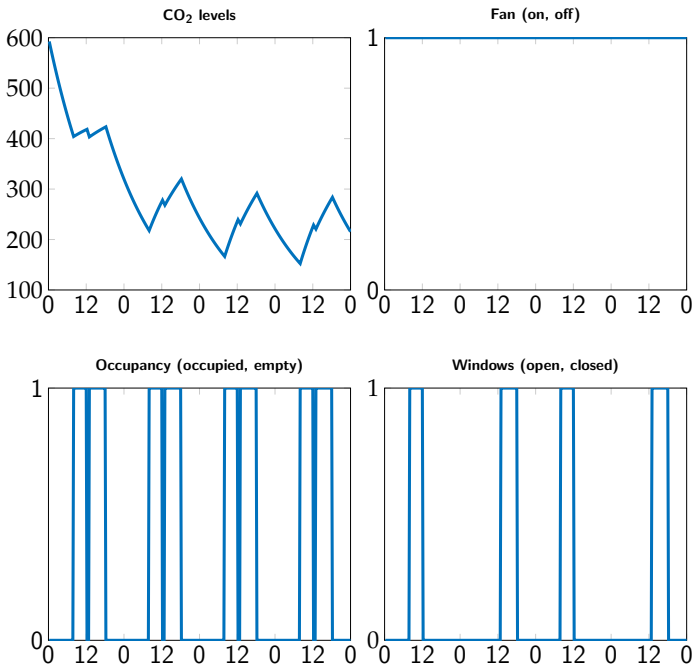


Building automation systems - problem setup

- model CO_2 dynamics, under the effect of
  1. occupants: room full (F)/empty (E)
  2. window: open (O)/closed (C)
  3. air circulation: ON

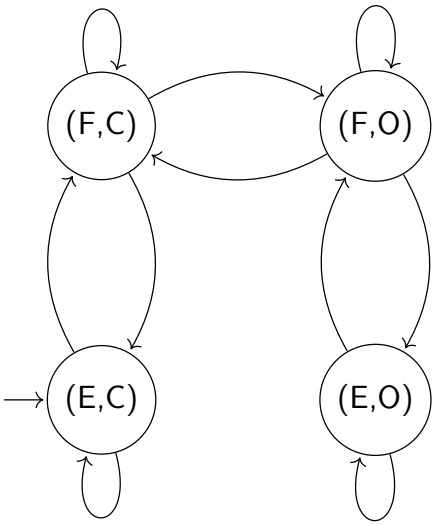


model with *hybrid* dynamics

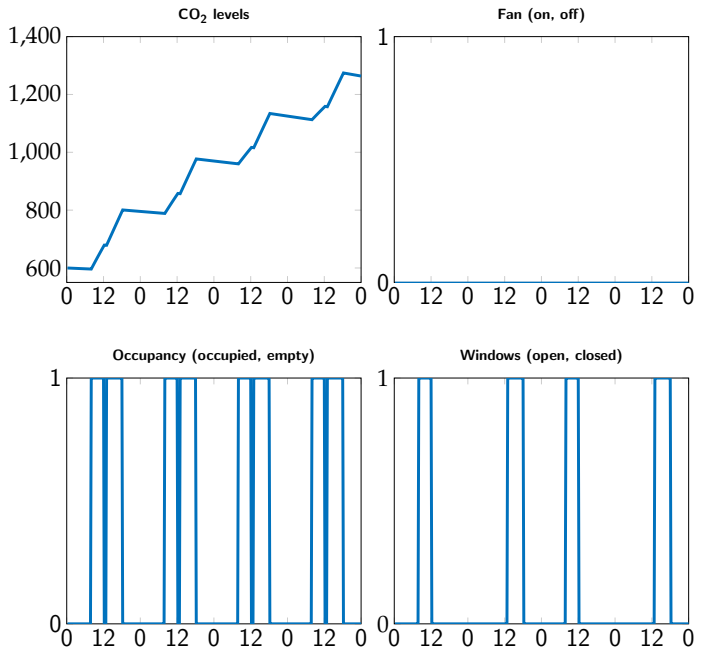


Building automation systems - problem setup

- model CO_2 dynamics, under the effect of
  1. occupants: room full (F)/empty (E)
  2. window: open (O)/closed (C)
  3. air circulation: OFF



model with *hybrid* dynamics



Outline

1. Verification of physical systems: Integration of learning and verification

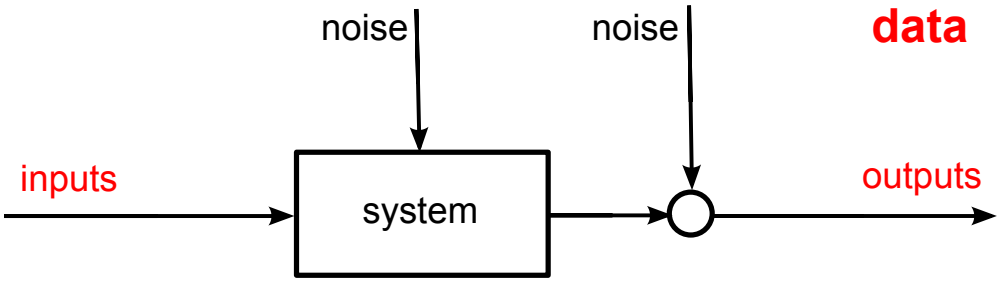
2. Verification of complex models with formal abstractions

Outline

1. Verification of physical systems: Integration of learning and verification

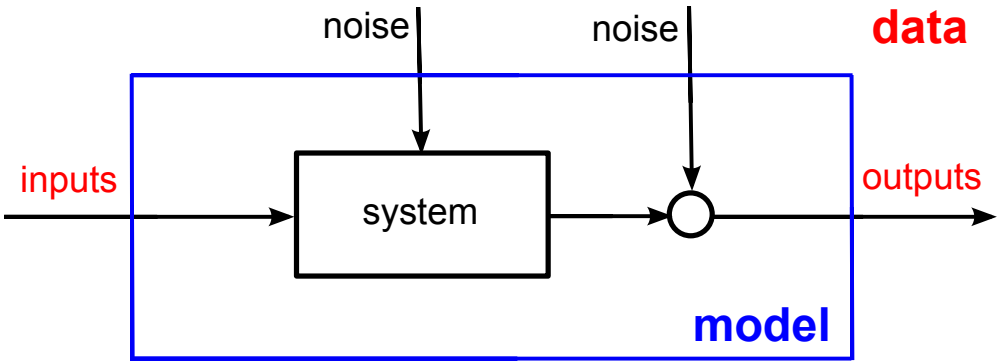
2. Verification of complex models with formal abstractions

Learning and verification: state of art and objective



data-driven analysis

Learning and verification: state of art and objective

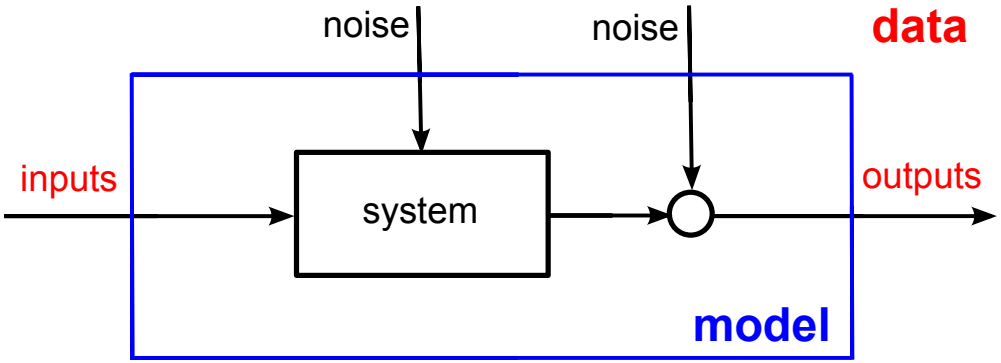


data-driven analysis

model learning (with data), and

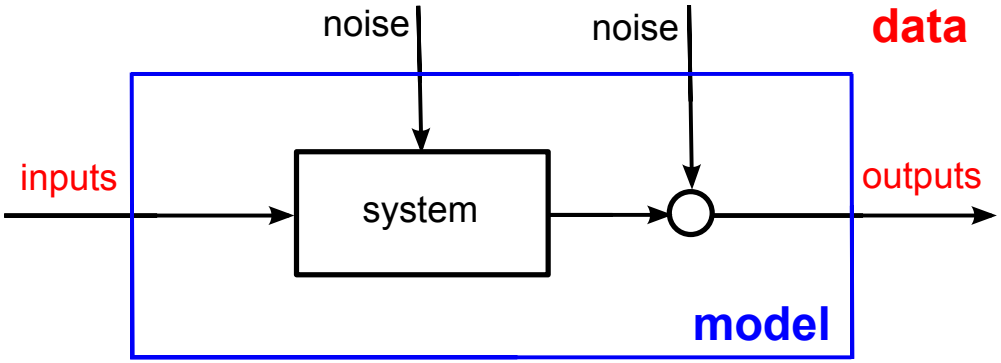
model-based verification

Learning and verification: state of art and objective



disconnect between data-driven learning and model-based verification

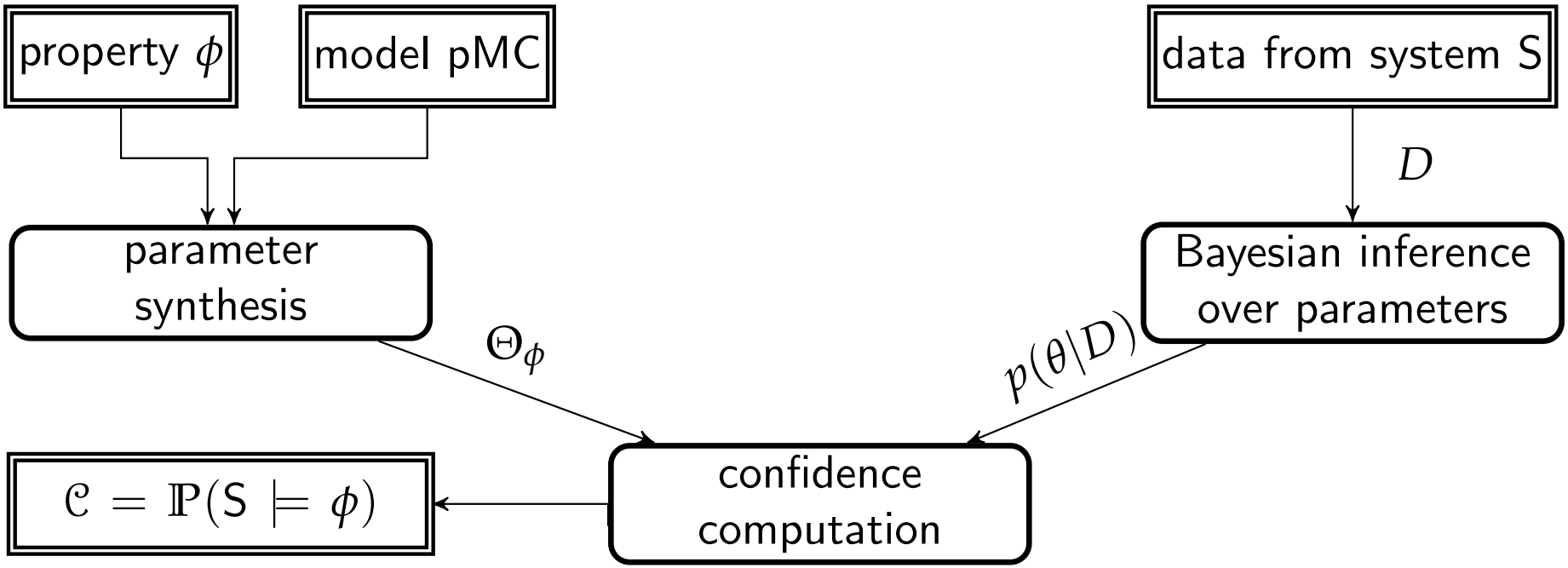
Learning and verification: state of art and objective



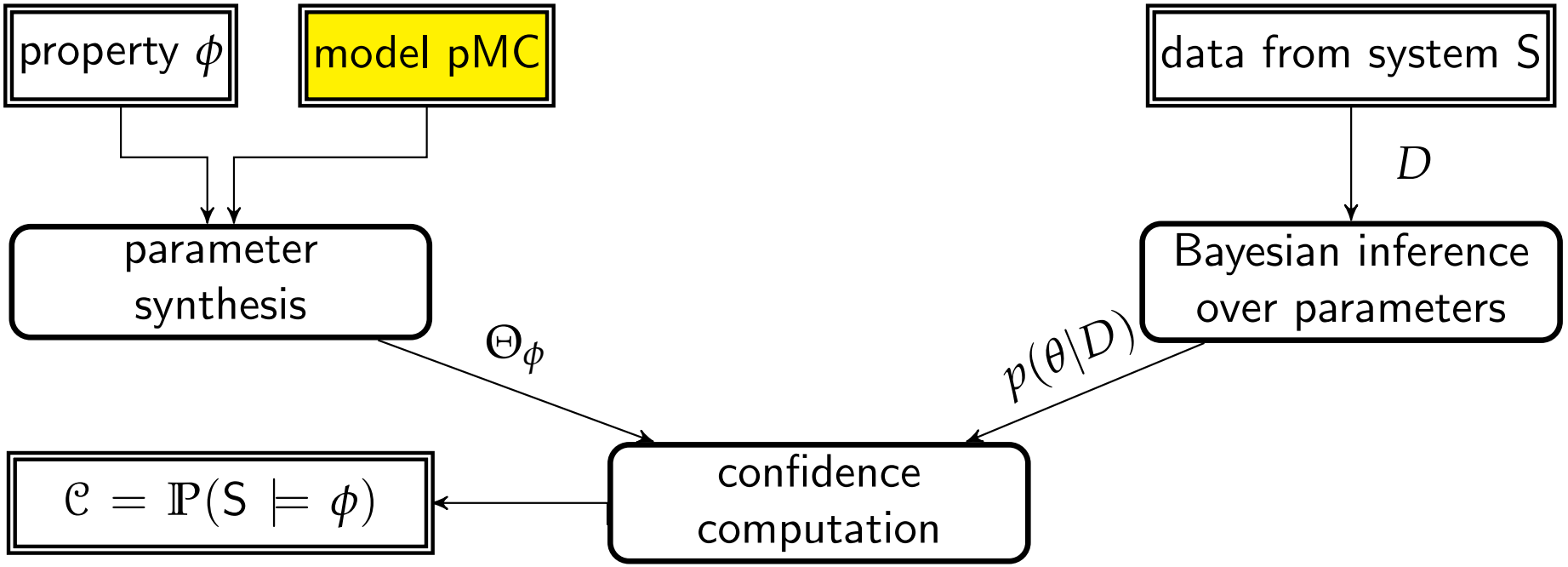
disconnect between data-driven learning and model-based verification

principled integration of learning and verification

Overview of method



Parametric Markov chains



Parametric Markov chains

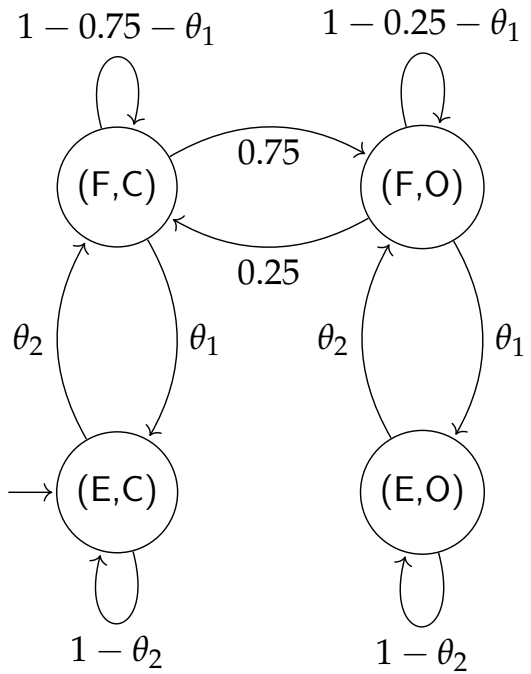
$$\mathcal{G} = (\Theta, S, \mathbb{T}_\theta, \rightarrow, \mathrm{AP}, L)$$

S – set of states

$\mathbb{T}_\theta$ – mapping $S \times S \rightarrow [0,1]$ expressed in terms of $\theta \in \Theta$

Θ – set of all possible valuations of θ, vector of parameters

$\rightarrow$ – starting states



$$\Theta = [0, 0.25] \times [0, 1]$$

Parametric Markov chains

$$\mathcal{G} = (\Theta, S, \mathbb{T}_\theta, \rightarrow, \mathrm{AP}, L)$$

S – set of states

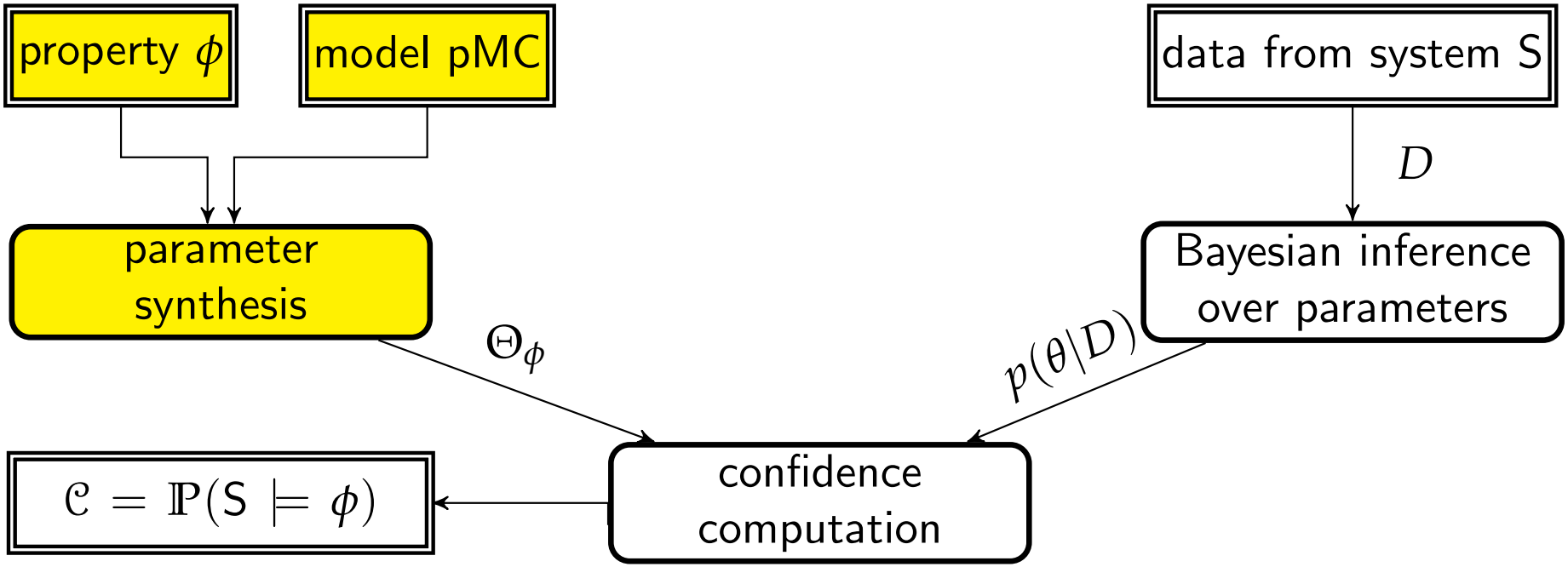
$\mathbb{T}_\theta$ – mapping $S \times S \rightarrow [0,1]$ expressed in terms of $\theta \in \Theta$

Θ – set of all possible valuations of θ, vector of parameters

$\rightarrow$ – starting states

L – labelling function, mapping states into 2^{AP}, AP alphabet

- denote by $M(\theta) \in \mathcal{G}$ a model parameterised by $\theta \in \Theta$


property ϕ
model pMC
data from system S
D
parameter synthesis
Bayesian inference over parameters
Θ_ϕ
$p(\theta|D)$
$\mathcal{C} = \mathbb{P}(S \models \phi)$
confidence computation

- property ϕ specified in PCTL, e.g.

  $\phi = \mathbb{P}_{\geq 0.99}(\Box^{\leq 20} \text{ safe}), \qquad \phi = \mathbb{P}_{>0.5}(\text{safe U reach}), \qquad \text{safe}, \text{reach} \in \text{AP}$

- *probabilistic model checking* PCTL properties over Markov chains
  - `input:` Markov chain $(S, \mathbb{T})$, PCTL formula ϕ
  - `output:` $\text{Sat}(\phi) = \{z \in S : z \models \phi\}$

- tools: PRISM, STORM, ...

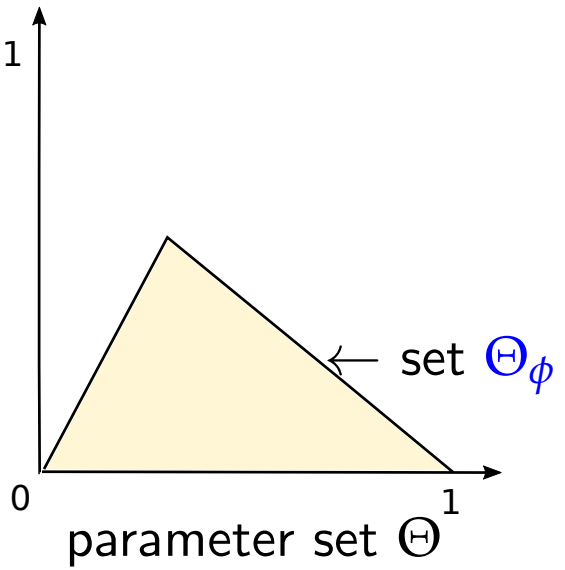
Parameter synthesis

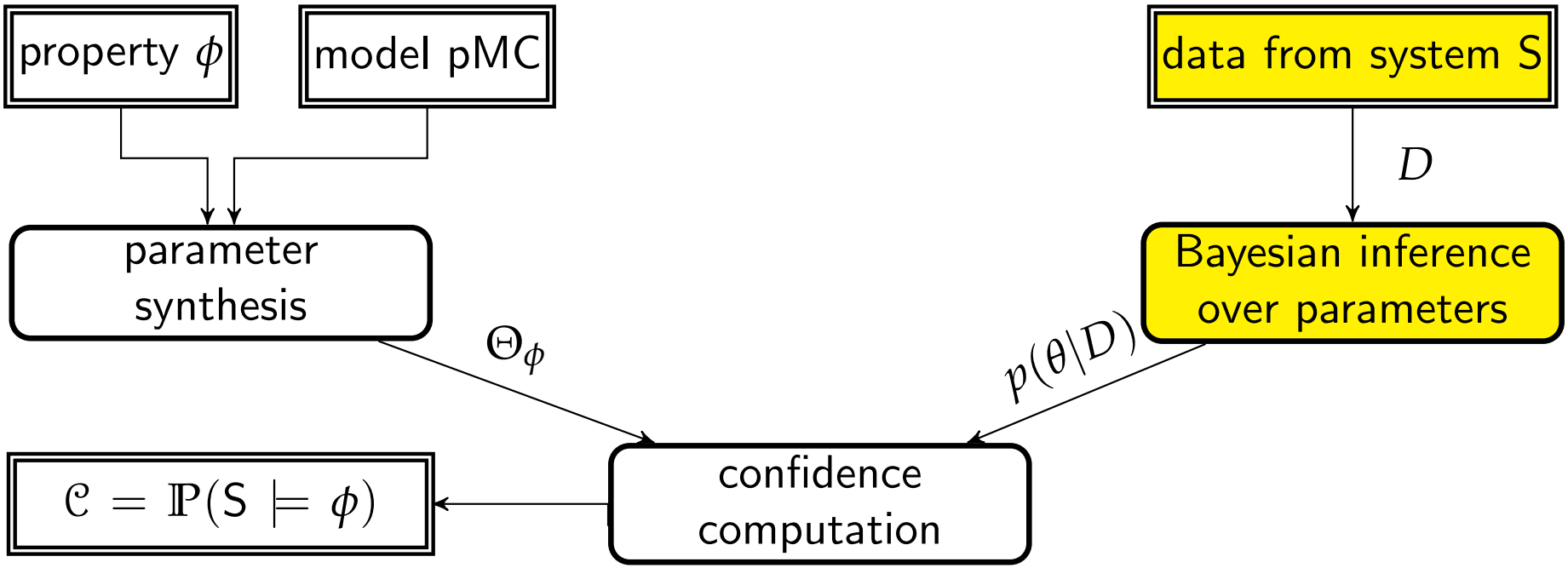
- property ϕ specified in PCTL, e.g.

$$\phi = \mathbb{P}_{\geq 0.99}(\Box^{\leq 20} \text{ safe}), \qquad \phi = \mathbb{P}_{>0.5}(\text{safe U reach}), \qquad \text{safe}, \text{reach} \in \text{AP}$$

- classify models in Θ according to property of interest ϕ, that is
- synthesise parameters $\theta \in \Theta$ s.t. $M(\theta)$ satisfies ϕ:

$$\Theta_\phi = \{\theta \in \Theta : M(\theta) \models \phi\} \subseteq \Theta$$




property ϕ
model pMC
data from system S
D
parameter synthesis
Bayesian inference over parameters
Θ_ϕ
$p(\theta|D)$
$\mathcal{C} = \mathbb{P}(S \models \phi)$
confidence computation

Bayesian inference

$$p(\theta_j \mid D) = \frac{\mathbb{P}(D \mid \theta_j) p(\theta_j)}{\mathbb{P}(D)}$$

$$= \frac{\prod_{s' \in S} \mathbb{T}_\theta(s_j, s')^{D^{s'}_{s_j}} \; p(\theta_j)}{\mathbb{P}(D_{s_j})}$$

- D – overall data gathered (traces)
  D_{s_j} – traces crossing state s_j, where $\theta_j = \theta_{s_j}$
- $p(\theta_j)$ – prior distribution
- $\prod_{s' \in S} \mathbb{T}_\theta(s_j, s')^{D^{s'}_{s_j}}$ – likelihood, multinomial distribution at state s_j

Bayesian inference

$$p(\theta_j \mid D) = \frac{\mathbb{P}(D \mid \theta_j) p(\theta_j)}{\mathbb{P}(D)}$$

$$= \frac{\prod_{s' \in S} \mathbb{T}_\theta(s_j, s')^{D^{s'}_{s_j}} \; p(\theta_j)}{\mathbb{P}(D_{s_j})}$$

- D – overall data gathered (traces)
  D_{s_j} – traces crossing state s_j, where $\theta_j = \theta_{s_j}$
- select as conjugate prior the Dirichlet distribution

$$p(\theta_j) = \mathrm{Dir}(\theta_j \mid \alpha) \propto \theta_j^{\alpha_1 - 1}(1 - \theta_j)^{\alpha_2 - 1}$$

for pair $(\theta_j, 1 - \theta_j)$, with $\alpha = (\alpha_1, \alpha_2)$ hyperparameters

Bayesian inference

$$p(\theta_j \mid D) = \frac{\mathbb{P}(D \mid \theta_j) p(\theta_j)}{\mathbb{P}(D)}$$

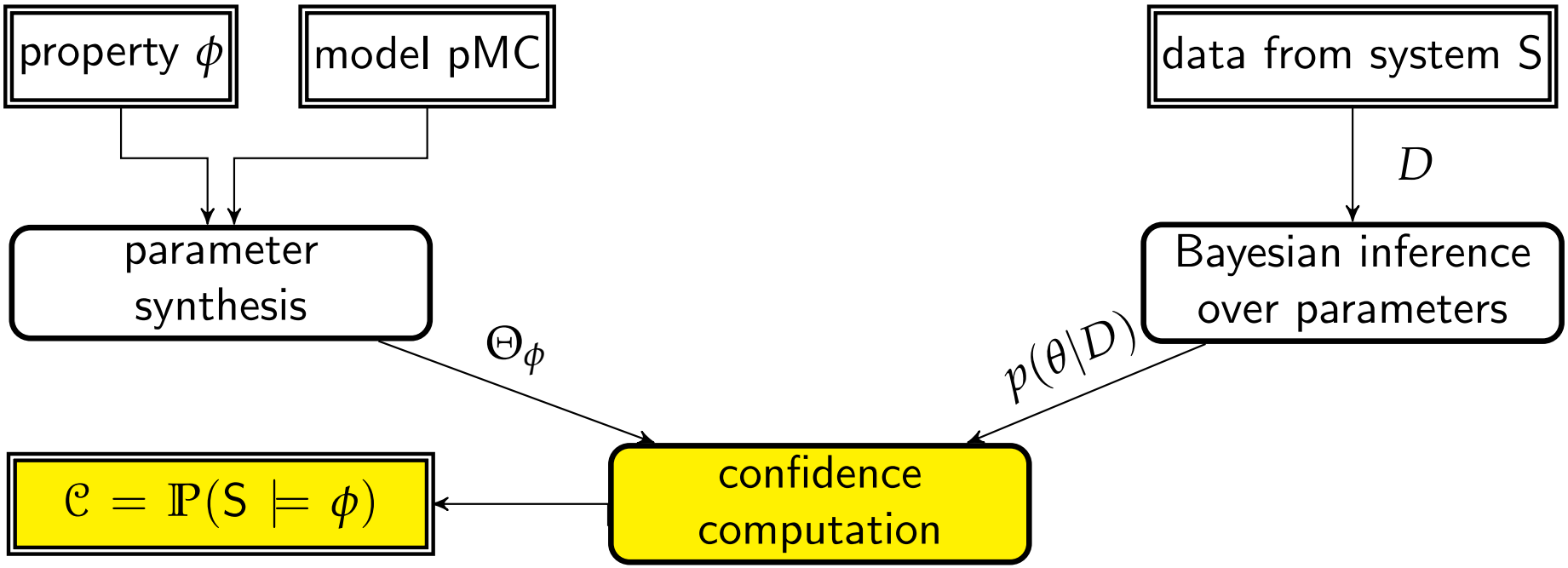
$$= \frac{\prod_{s' \in S} \mathbb{T}_\theta(s_j, s')^{D^{s'}_{s_j}} \; p(\theta_j)}{\mathbb{P}(D_{s_j})}$$

- D – overall data gathered (traces)
  D_{s_j} – traces crossing state s_j, where $\theta_j = \theta_{s_j}$
- under Dirichlet prior, posterior update is analytic

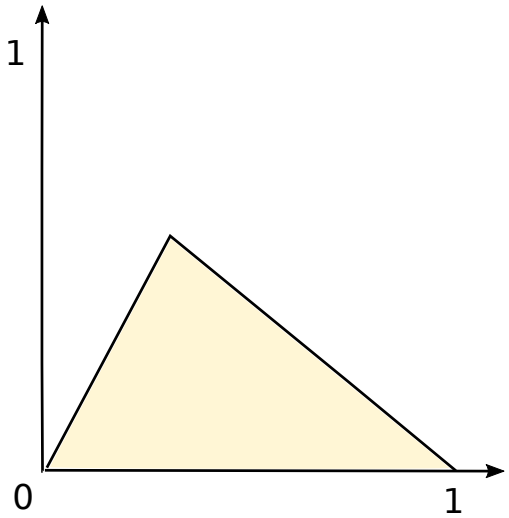
$$p(\theta_j \mid D) \propto \theta_j^{D^{s'_1}_{s_j}} (1-\theta_j)^{D^{s'_2}_{s_j}} \; \theta_j^{\alpha_1 - 1} (1-\theta_j)^{\alpha_2 - 1}$$

and obtained updating hyperparameters of Dirichlet distribution, as

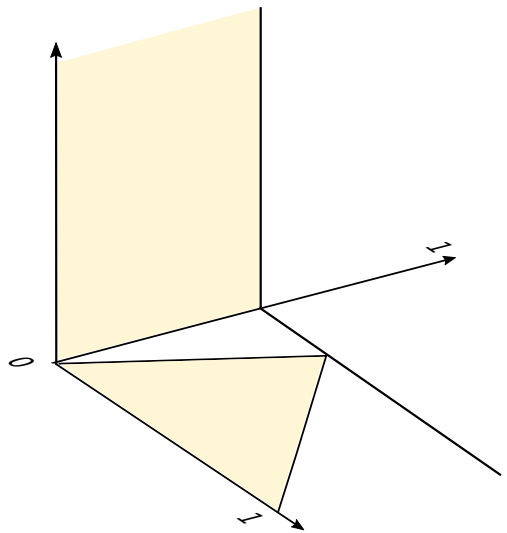
$$p(\theta_j | D) = \mathrm{Dir}(\theta_j \mid D_{s_j} + \alpha)$$


property ϕ
model pMC
data from system S
D
parameter synthesis
Bayesian inference over parameters
Θ_ϕ
$p(\theta|D)$
$\mathcal{C} = \mathbb{P}(S \models \phi)$
confidence computation

Confidence computation



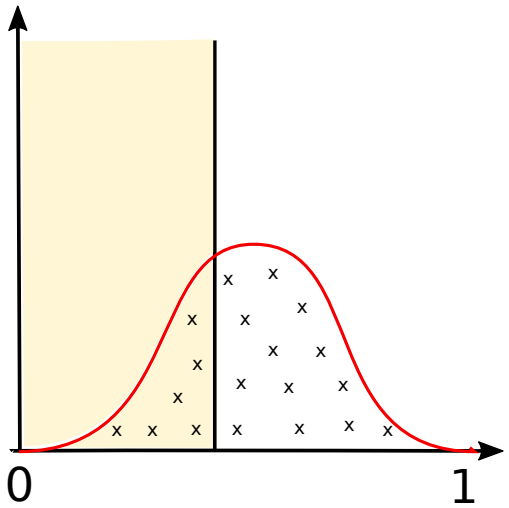
Confidence computation



Confidence computation

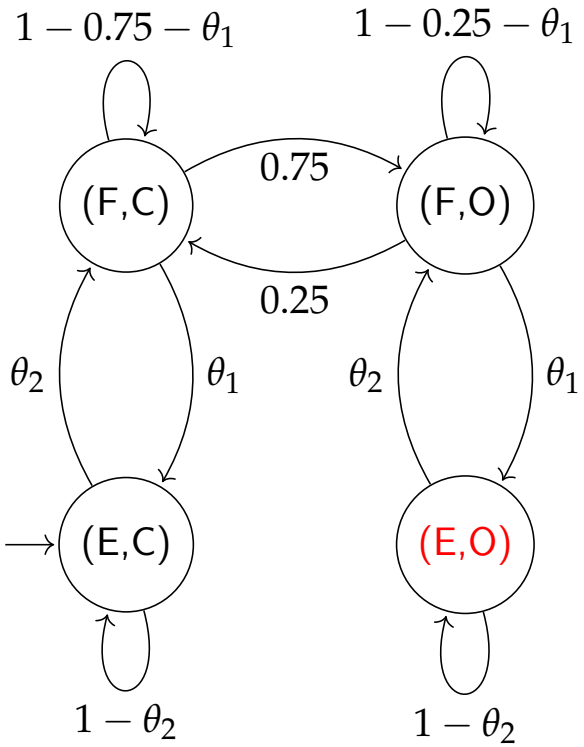
- compute confidence $\mathcal{C}$ on whether system S satisfies property ϕ as

$$\mathcal{C} = \mathbb{P}(\mathsf{S} \models \phi \mid D) = \int_{\Theta_\phi} p(\theta \mid D) d\theta$$



Case study: setup

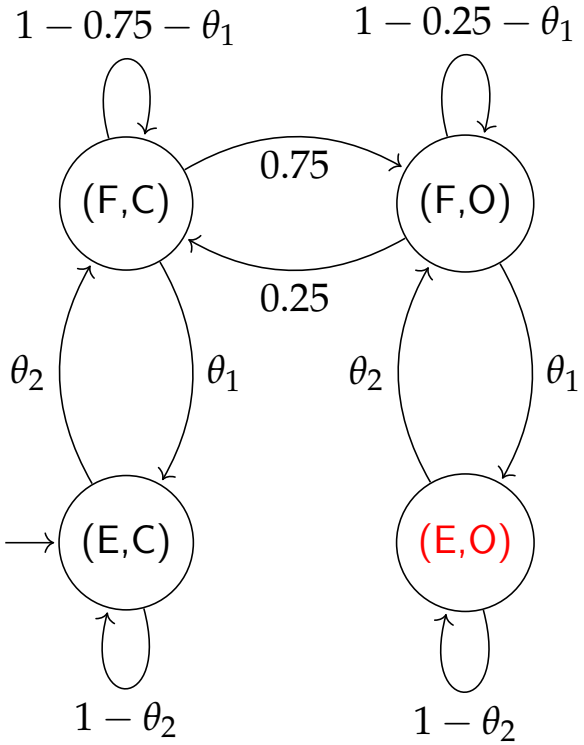
- goal: benchmark against statistical model checking (SMC)
- pMC model:

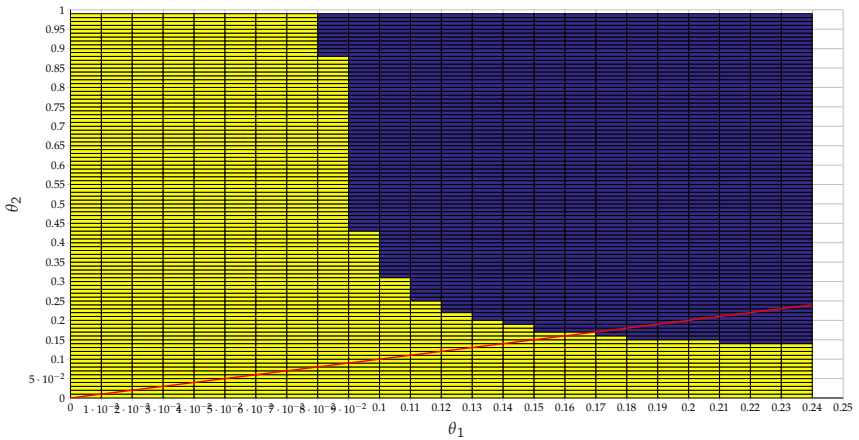


- specification: $\phi = \mathbb{P}_{>0.3}(\Box^{\leq 20} \neg(E,O))$

Case study: setup

- goal: benchmark against statistical model checking (SMC)
- pMC model:



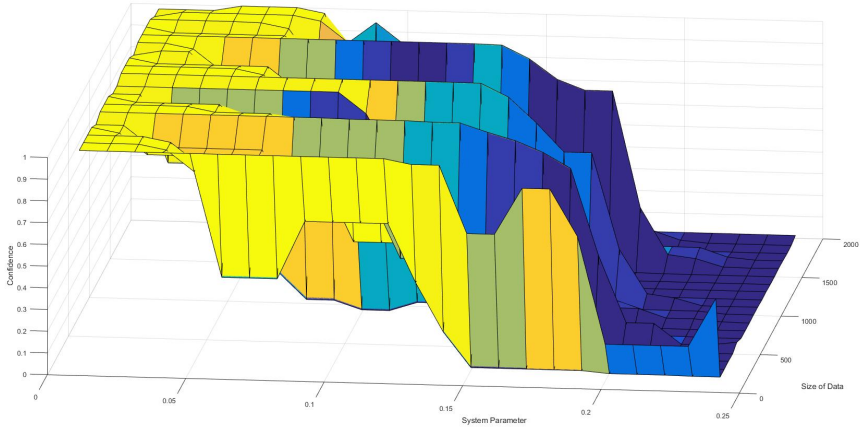


- specification: $\phi = \mathbb{P}_{>0.3}(\Box^{\leq 20} \neg(E,O))$
- for selected pMC and property, synthesis yields Θ_ϕ (yellow set)

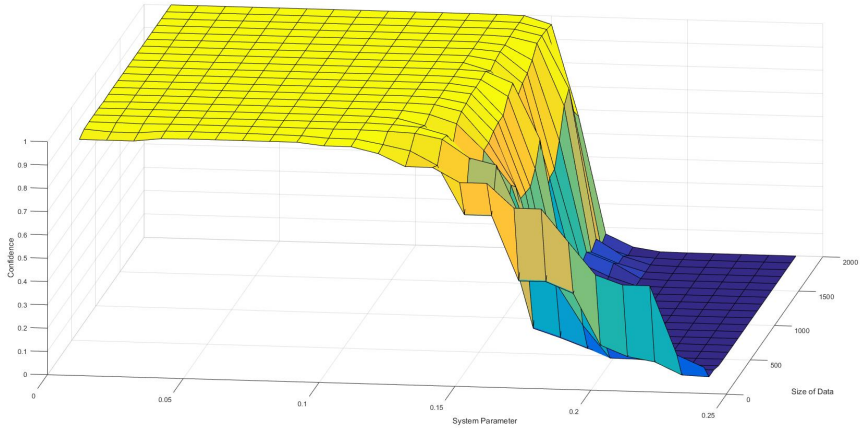
Case study: experiments

- data: state trajectories of different length

SMC



this work



- attains confidence closer to "true" value than SMC
- extracts information from data more efficiently
- is more robust with limited data

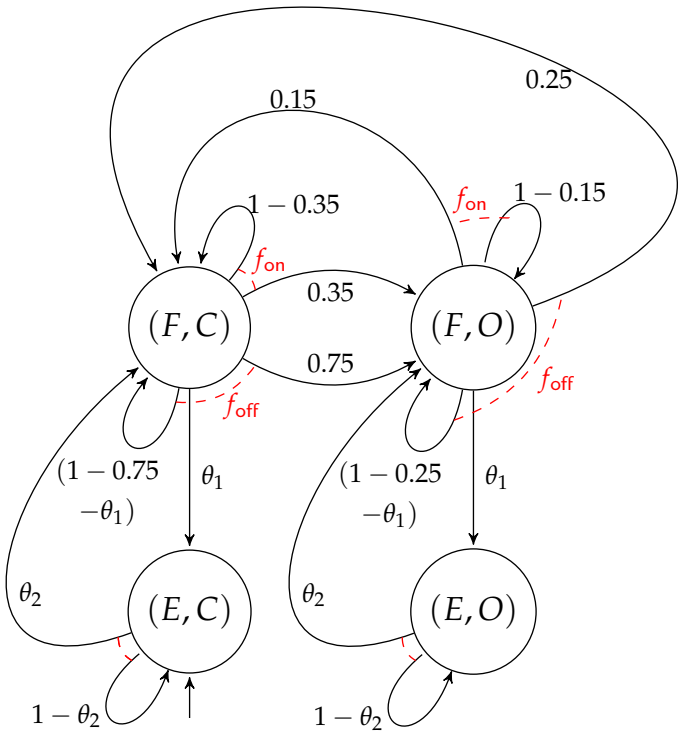
Parametric Markov decision processes

$$\mathcal{G} = (\Theta, S, A, \mathbb{T}_\theta, \to, \mathrm{AP}, L)$$

$\Theta, S, \to, L$ – as before

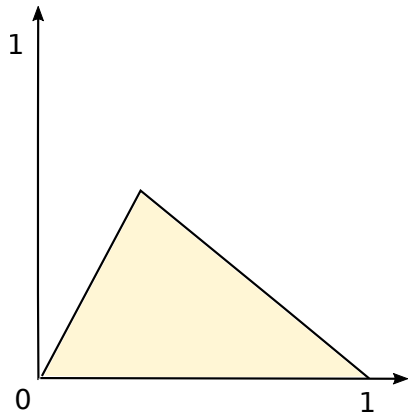
A – set of actions

$\mathbb{T}_\theta$ – mapping $S \times A \times S \to [0,1]$ expressed in terms of $\theta \in \Theta$



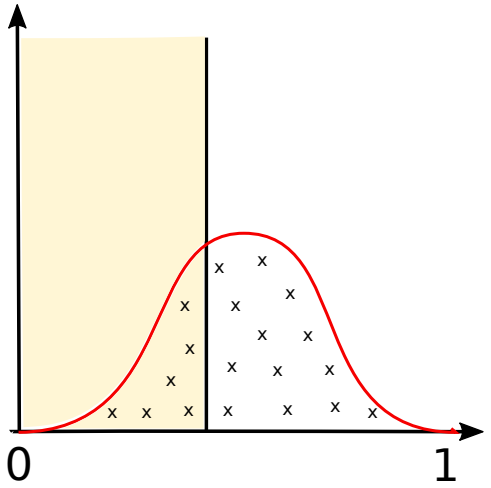
Dual role of actions in pMDP

- actions can be employed to shape set Θ_ϕ



shape set Θ_ϕ

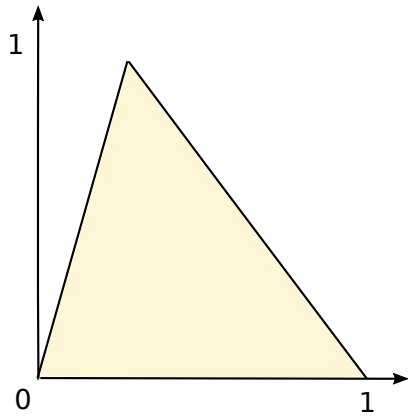
- actions can be chosen to affect confidence level $\mathcal{C}$



integral = confidence level

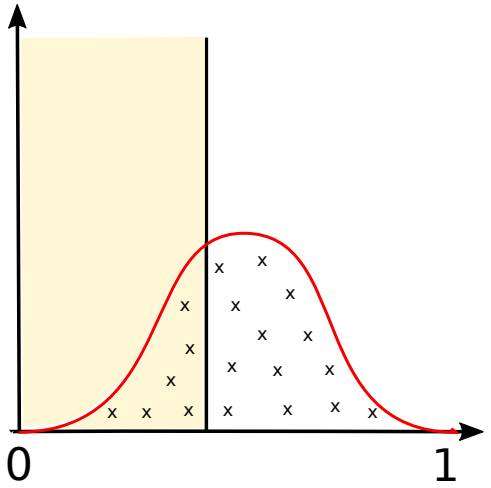
Dual role of actions in pMDP

- actions can be employed to shape set Θ_ϕ



shape set Θ_ϕ

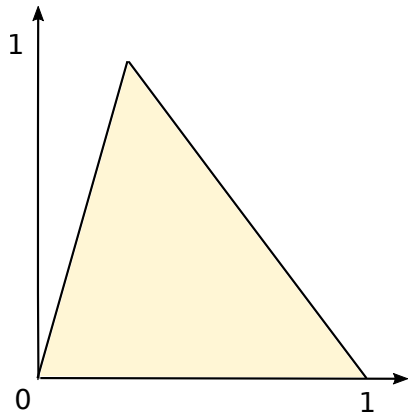
- actions can be chosen to affect confidence level $\mathcal{C}$



integral = confidence level

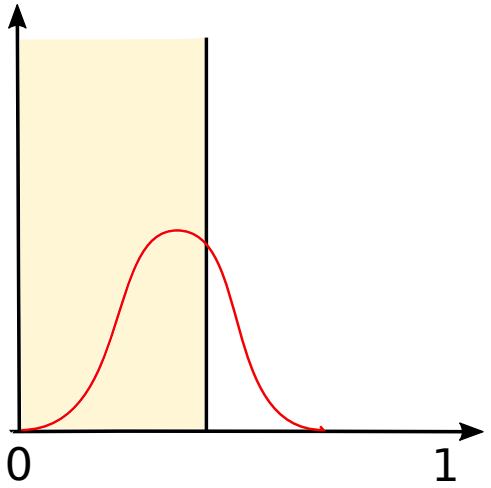
Dual role of actions in pMDP

- actions can be employed to shape set Θ_ϕ



shape set Θ_ϕ

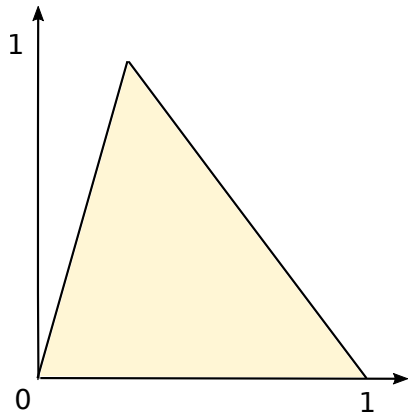
- actions can be chosen to affect confidence level $\mathcal{C}$



integral $\rightarrow$ confidence level

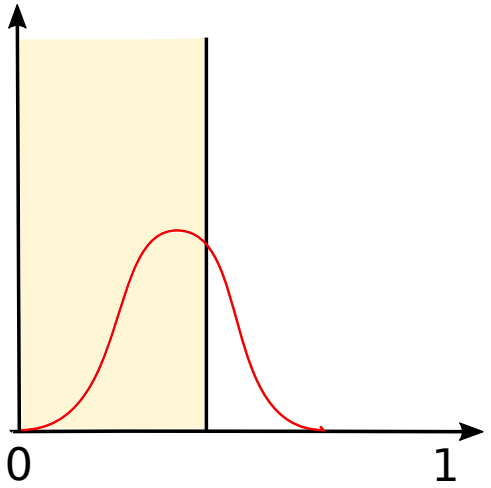
Dual role of actions in pMDP

- actions can be employed to shape set Θ_ϕ



shape set Θ_ϕ

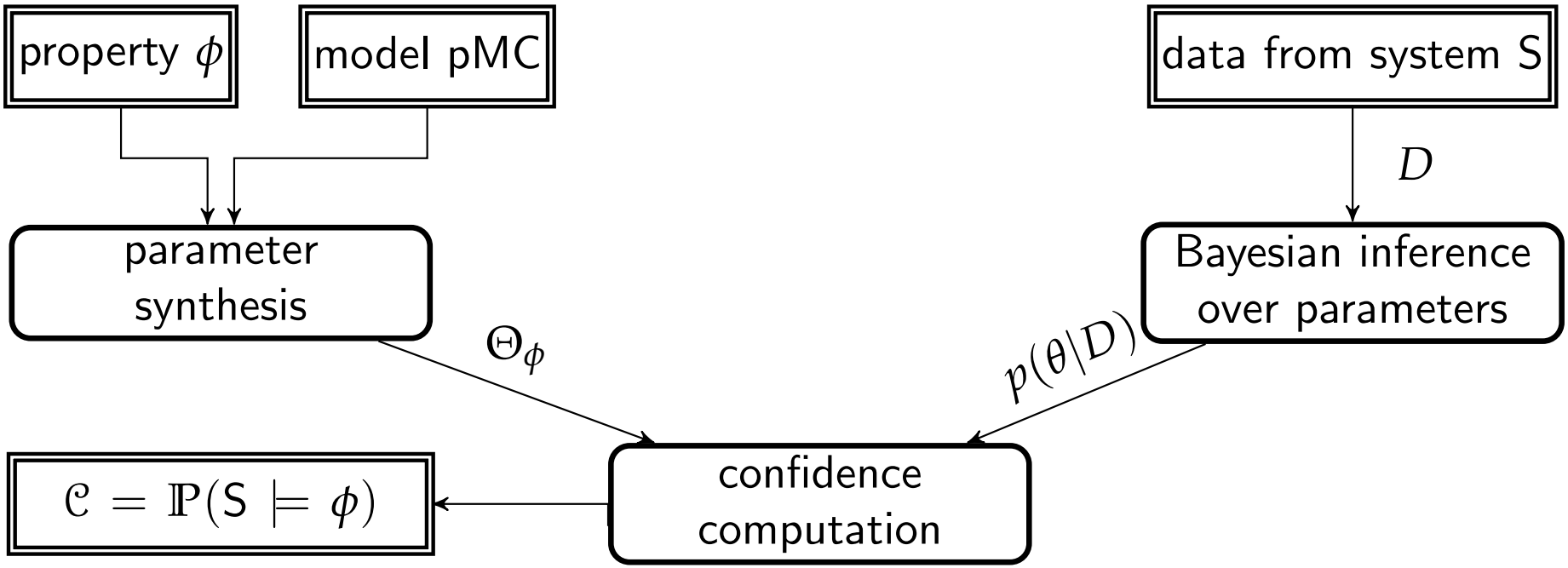
- actions can be chosen to affect confidence level $\mathcal{C}$



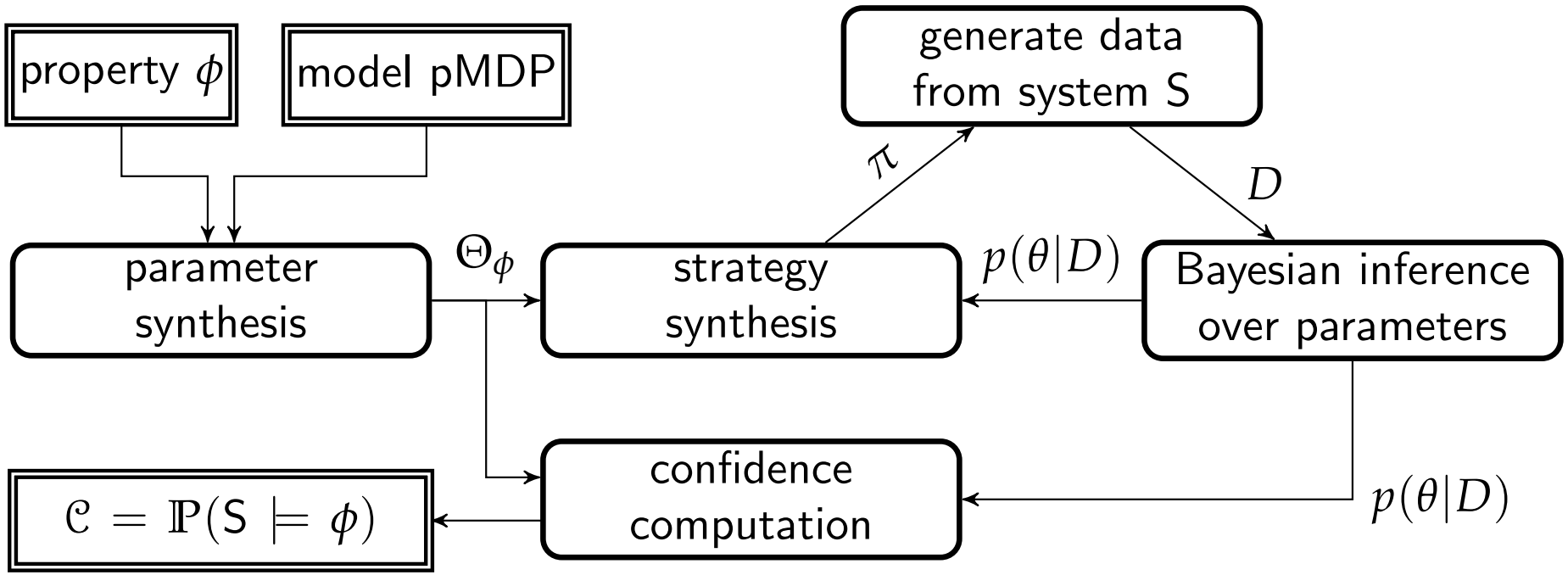
integral $\rightarrow$ confidence level

reminiscent of exploration/exploitation tradeoff in RL

Overview of method



Overview of method



Strategy synthesis for experiment design

- design experiments to affect confidence calculation

$$\max \{\mathbb{P}(\mathsf{S} \models \phi \mid D), \mathbb{P}(\mathsf{S} \not\models \phi \mid D)\}$$

Strategy synthesis for experiment design

- design experiments to affect confidence calculation

$$\max\{\mathbb{P}(\mathsf{S} \models \phi \mid D), \mathbb{P}(\mathsf{S} \not\models \phi \mid D)\}$$

- expected confidence gain at state-action (s, α) (and corresp. parameter)

$$\mathcal{C}_{s,\alpha} = \int_{\Theta_\phi} \prod_{\theta_i \in \theta} p(\theta_i \mid \mathbb{E}_{s,\alpha}(D_i)) d\theta$$

Strategy synthesis for experiment design

- design experiments to affect confidence calculation

$$\max\{\mathbb{P}(\mathsf{S} \models \phi \mid D), \mathbb{P}(\mathsf{S} \not\models \phi \mid D)\}$$

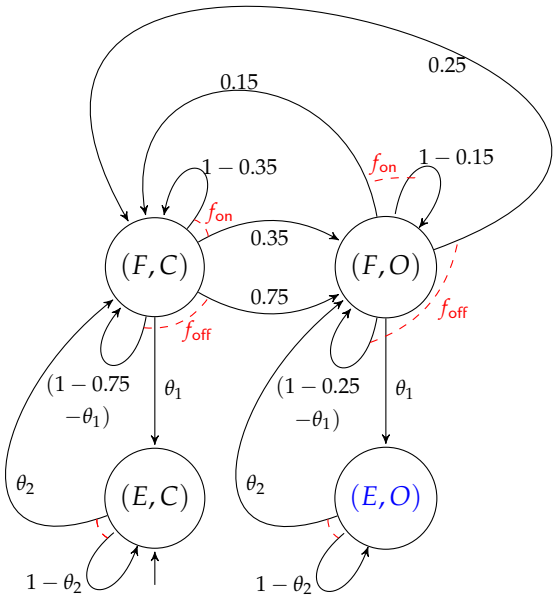
- expected confidence gain at state-action (s, α) (and corresp. parameter)

$$\mathcal{C}_{s,\alpha} = \int_{\Theta_\phi} \prod_{\theta_i \in \theta} p(\theta_i \mid \mathbb{E}_{s,\alpha}(D_i)) d\theta$$

- use $\mathcal{C}_{s,\alpha}$ as a reward for (s, α)
- synthesise optimal strategy π for experiment design

Case study: setup

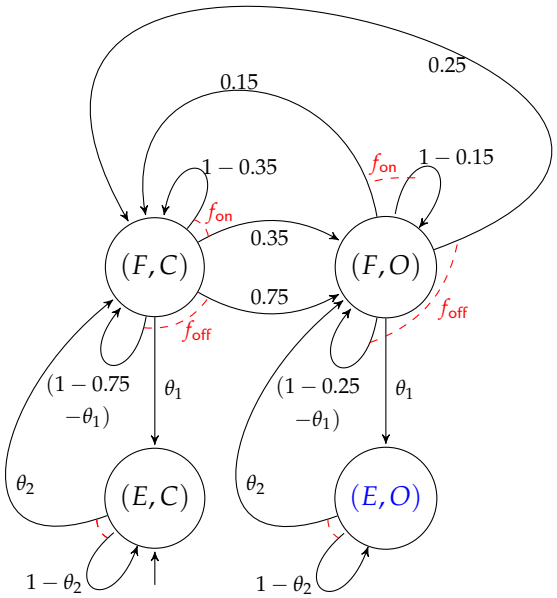
- goal: compare optimally synthesised policies vs. random/deterministic ones
- pMDP model:

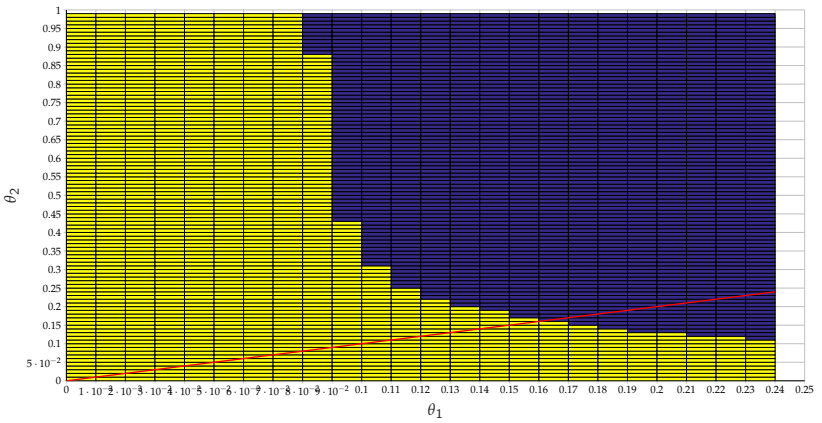


- specification: $\phi = \mathbb{P}_{>0.3}(\Box^{\leq 20} \neg(E,O))$

Case study: setup

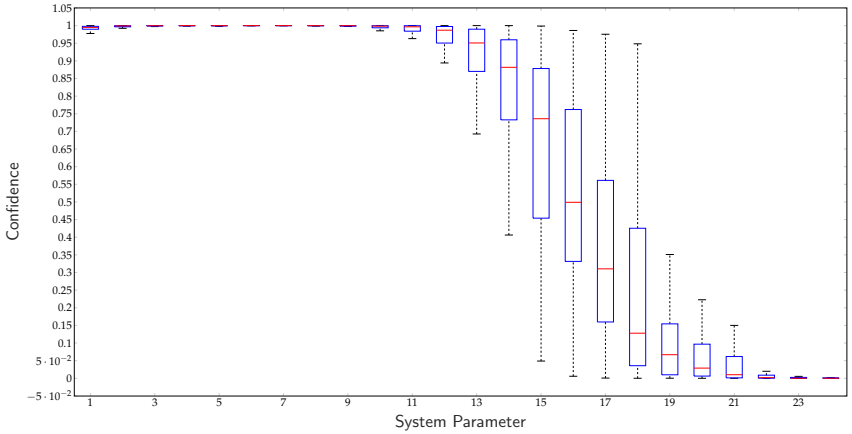
- goal: compare optimally synthesised policies vs. random/deterministic ones
- pMDP model:

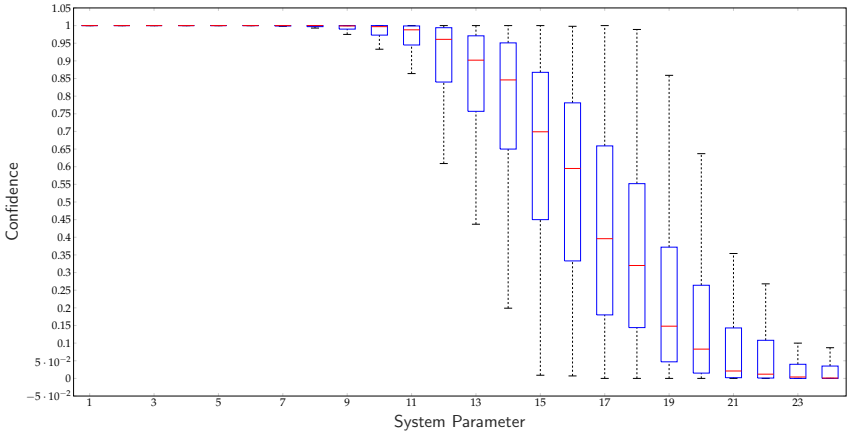




- specification: $\phi = \mathbb{P}_{>0.3}(\Box^{\leq 20} \neg(E,O))$
- for selected pMDP and given ϕ, Θ_ϕ is shown in yellow

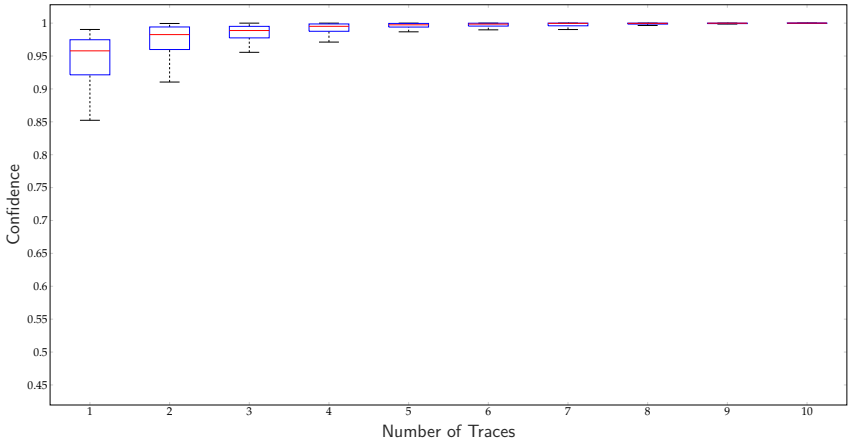
Case study: experiments

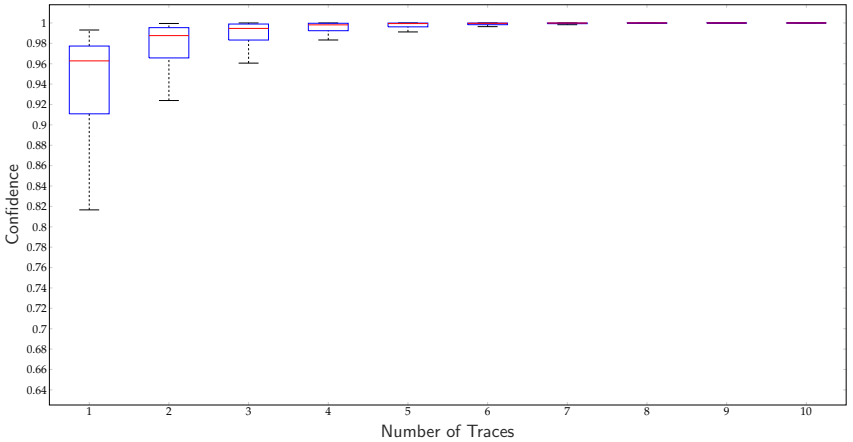




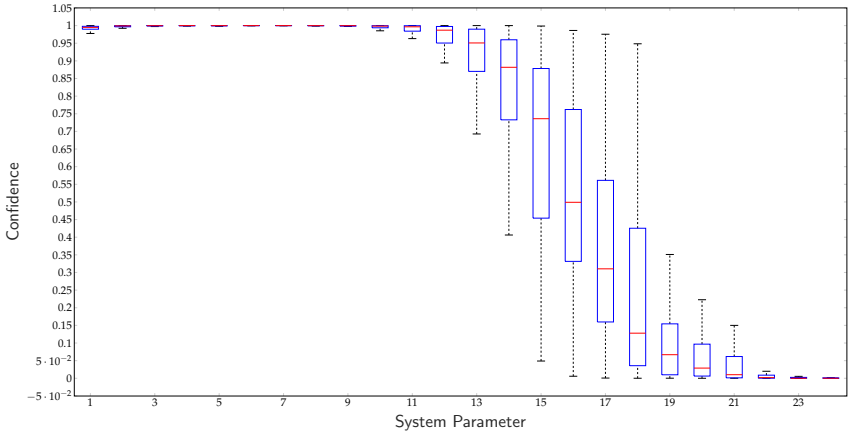
Synthesised strategy π

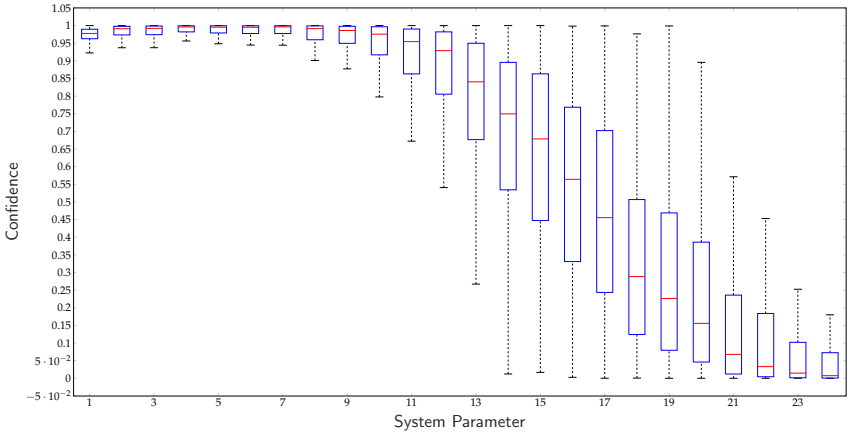
Fully random strategy





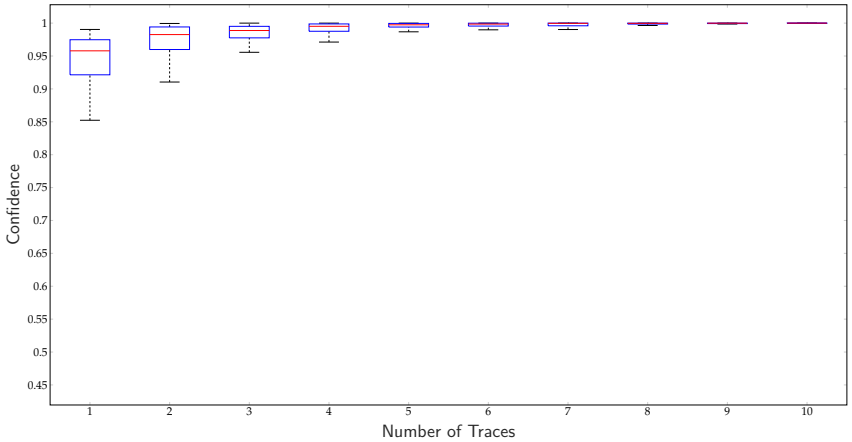
Case study: experiments

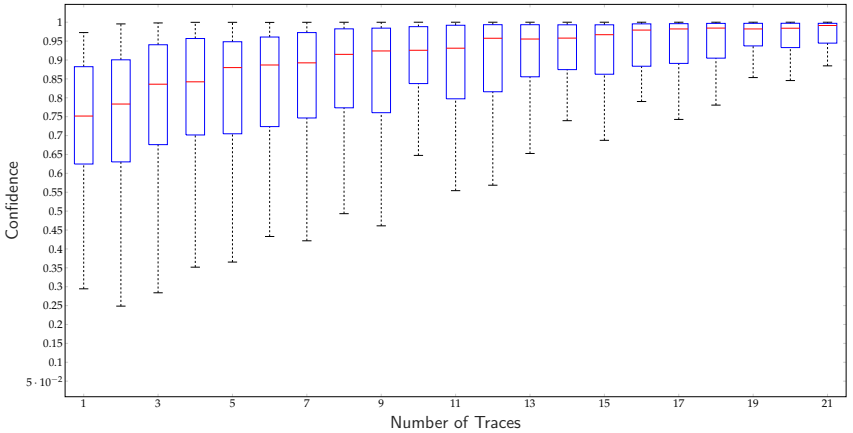


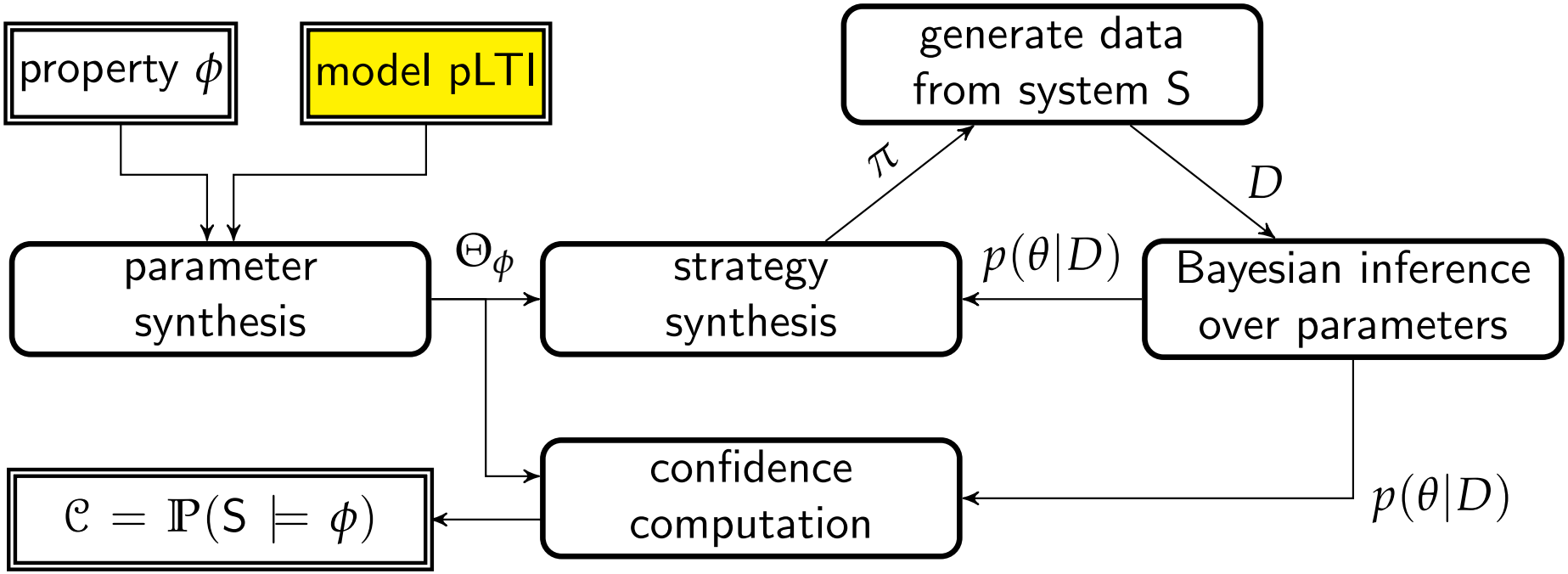


Synthesised strategy π

Deterministic strategy

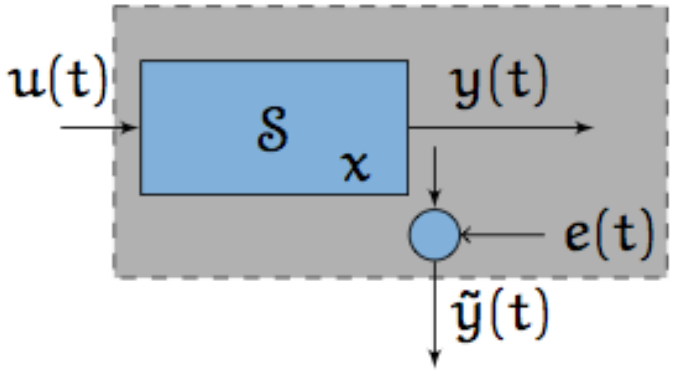





property ϕ
model pLTI
generate data from system S
π
D
parameter synthesis
Θ_ϕ
strategy synthesis
$p(\theta|D)$
Bayesian inference over parameters
confidence computation
$p(\theta|D)$
$\mathcal{C} = \mathbb{P}(\mathsf{S} \models \phi)$

Extensions to alternative model classes

- parametrised LTI model



$u(t)$ – input

$y(t)$ – system output

$\tilde{y}(t)$ – measured output

$e(t)$ – measurement noise, $e(t) \sim \mathcal{N}(0, \sigma_e^2)$

- model set $\mathcal{G} = \{M(\theta) \mid \theta \in \Theta\}$, where

$$M(\theta): \quad \begin{cases} x(t+1) & = Ax(t) + Bu(t) \\ y(t) & = \theta^T x(t) \end{cases}$$

Outline

1. Verification of physical systems: Integration of learning and verification

2. Verification of complex models with formal abstractions

Outline

1. Verification of physical systems: Integration of learning and verification

2. Verification of complex models with formal abstractions

Building automation systems – a complex model

- model CO_2 dynamics, coupled with temperature evolution

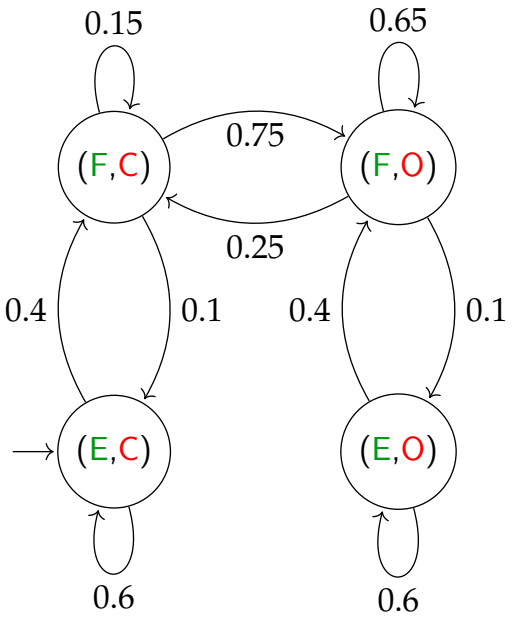
$$x_{k+1} = x_k + \frac{\Delta}{V}\left(-\mathbb{1}_{ON} m x_k + \mu_{\{O,C\}}(C_{out} - x_k)\right) + \mathbb{1}_F C_{occ} + \sigma_x w_k$$

$$y_{k+1} = y_k + \frac{\Delta}{C}\left(\mathbb{1}_{ON} m (T_{set} - y_k) + \mu_{\{O,C\}} \frac{1}{R}(T_{out} - y_k)\right) + \mathbb{1}_F T_{occ,k} + \sigma_y w_k$$

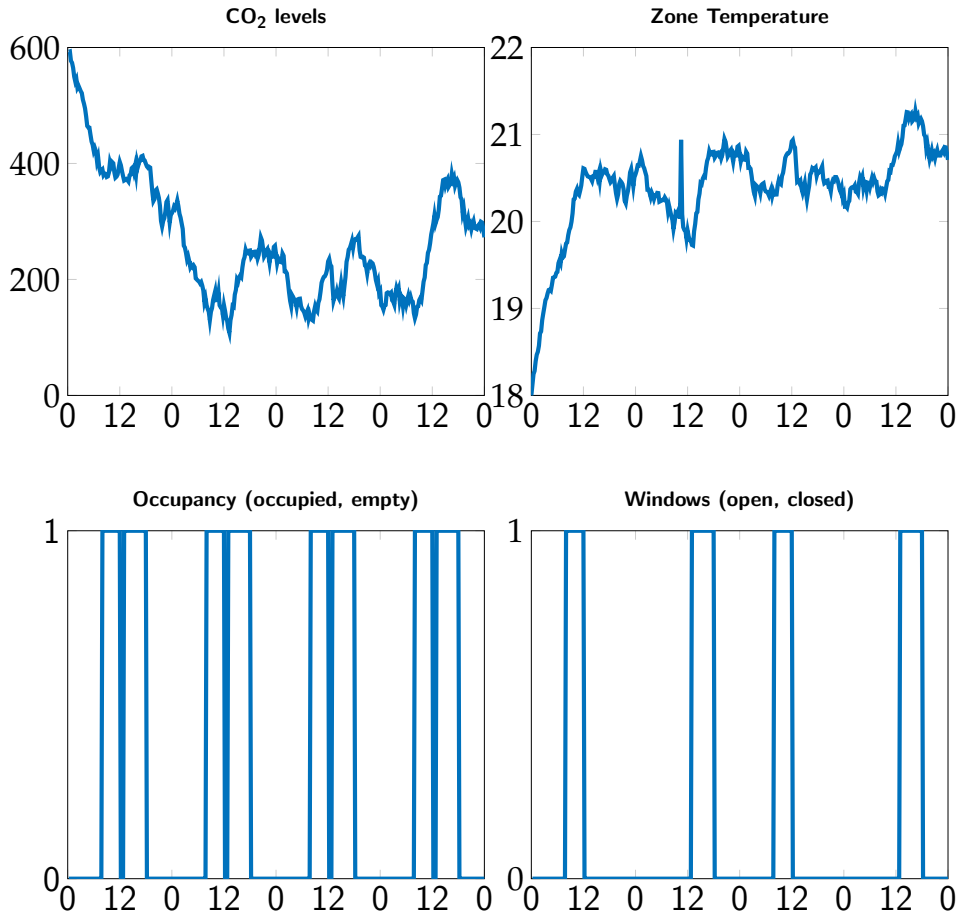
where $T_{occ,k} = \nu x_k + \zeta$

- x - zone 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 complex model



Parameter	Value
C	94.41 J/^{o}C
T_{set}	20 ^{o}C
T_{out}	24 ^{o}C
ν	$2.4 \cdot 10^{-4}$
ζ	0.0107

- air circulation: ON

Complex models: from finite to uncountable

finite-space Markov chain

$(S, \mathbb{T})$

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

$$\mathbb{T} = \begin{bmatrix} p_{11} & \cdots & p_{14} \\ \cdots & \cdots & \cdots \\ p_{41} & \cdots & \cdots \end{bmatrix}$$

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

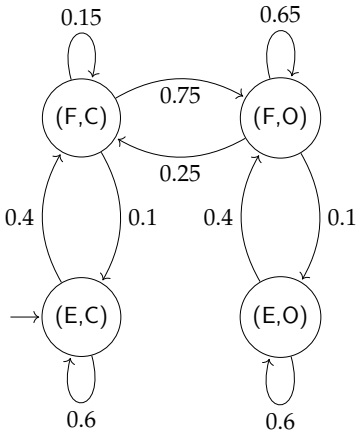
uncountable-space Markov process

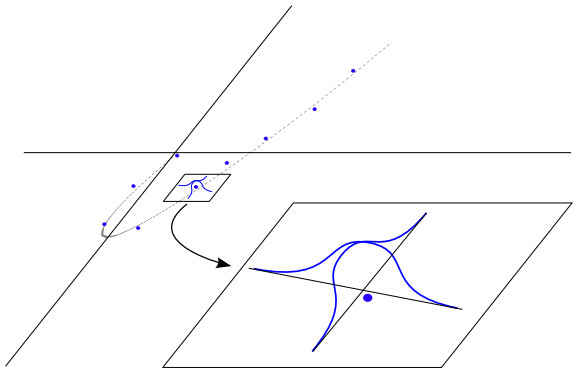
$(\mathcal{S}, \mathcal{T})$

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

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

$\mathbb{P}(s, A) = \int_A \mathcal{T}(dx|s), \quad A \subseteq \mathcal{S}$





Probabilistic model checking of complex models

- recall PCTL formulae

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

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

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

$$S(p) = \{s \in \mathcal{S} : \mathcal{P}_s(A) \geq p\}$$

Probabilistic model checking of complex models

- recall PCTL formulae

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

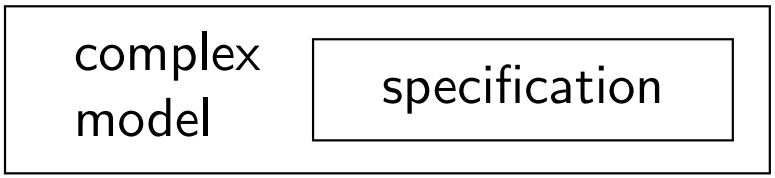
$$\mathcal{P}_s(A) = \mathbb{P}_s(s_k \in A, \forall k \in [0, N])$$

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

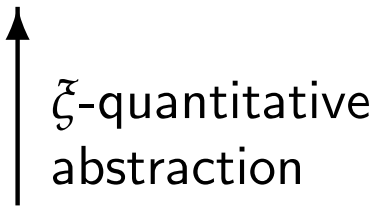
$$S(p) = \{s \in \mathcal{S} : \mathcal{P}_s(A) \geq p\}$$

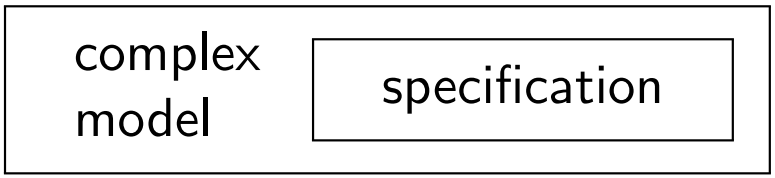
- issues with computation of $\mathcal{P}_s(A)$ and of $S(p)$

Formal abstractions

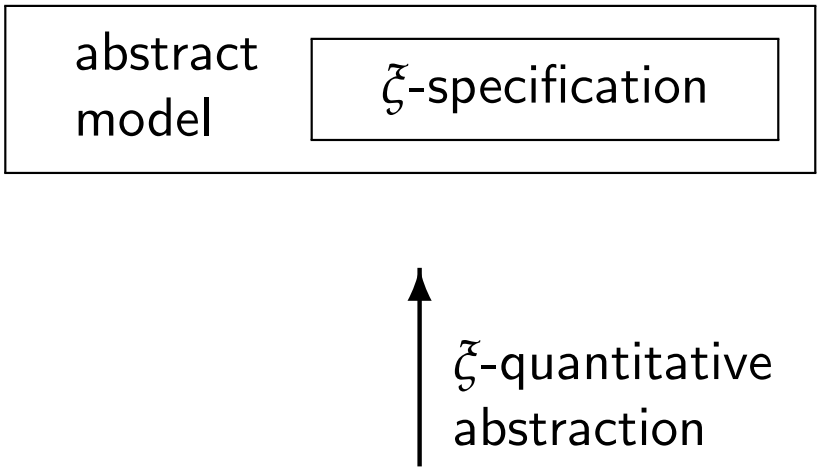


Formal abstractions

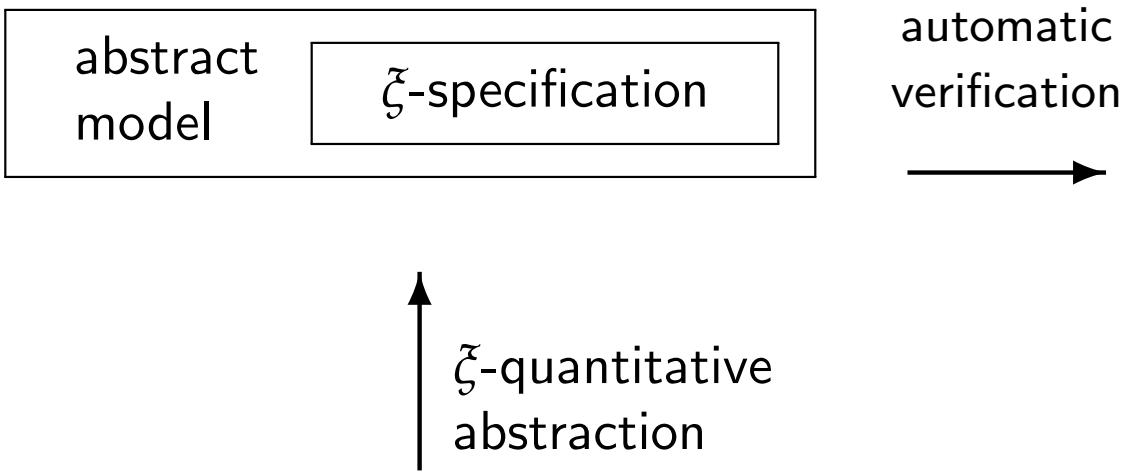




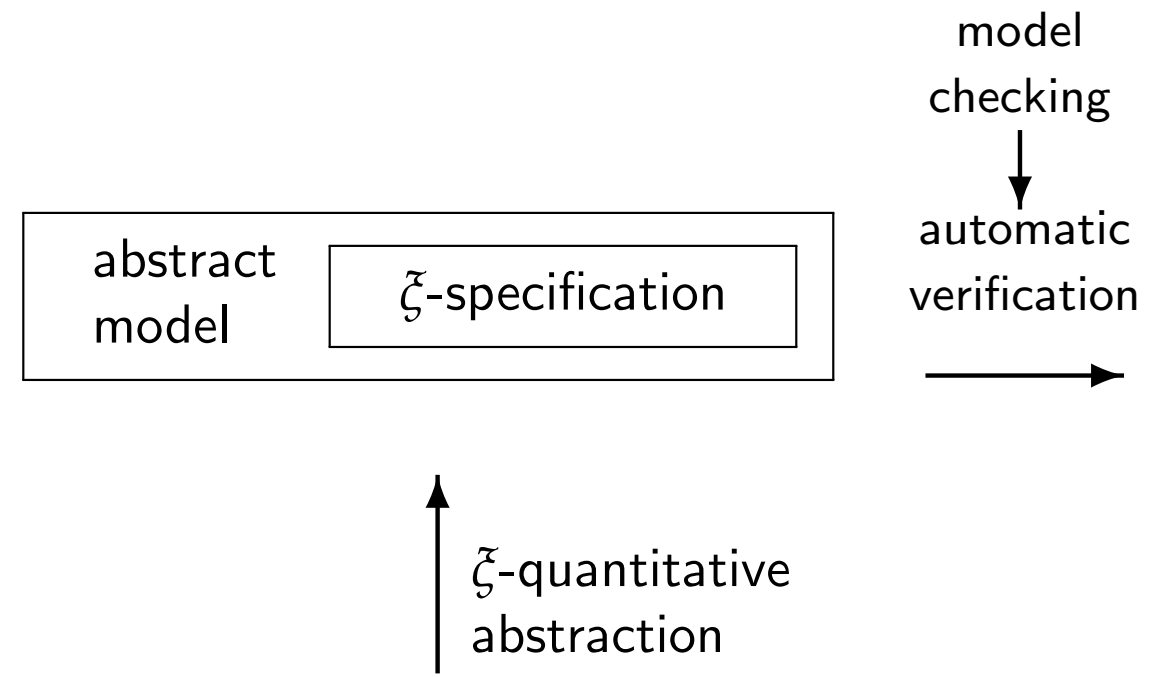
Formal abstractions



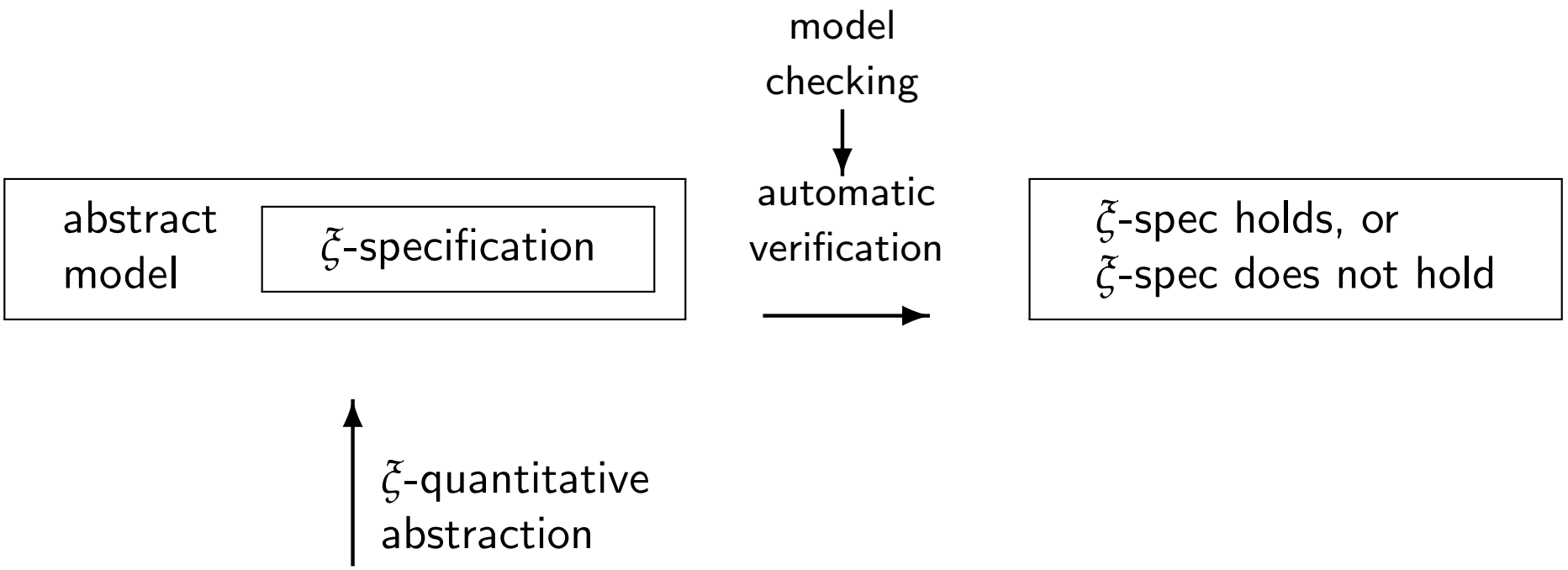
Formal abstractions



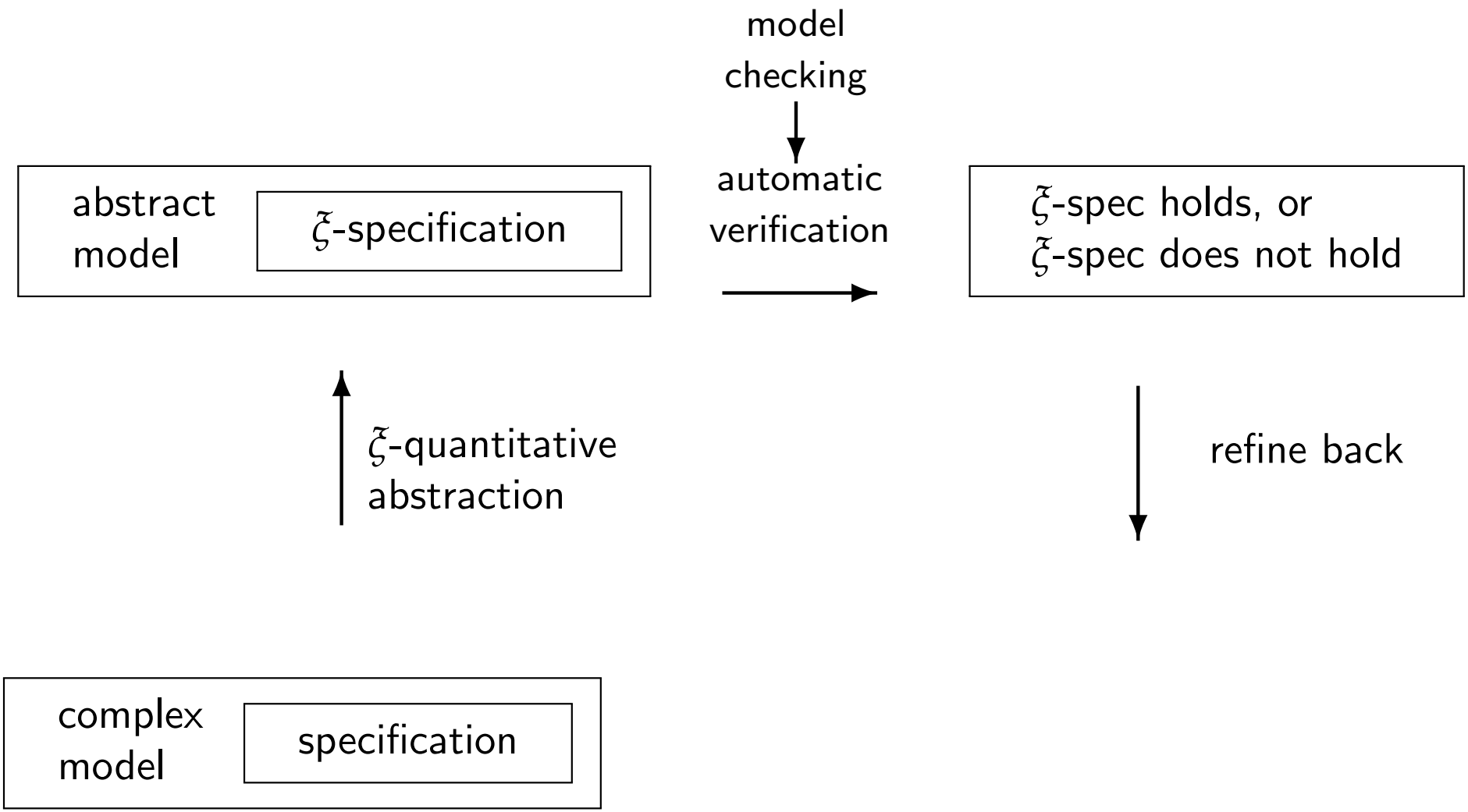
Formal abstractions



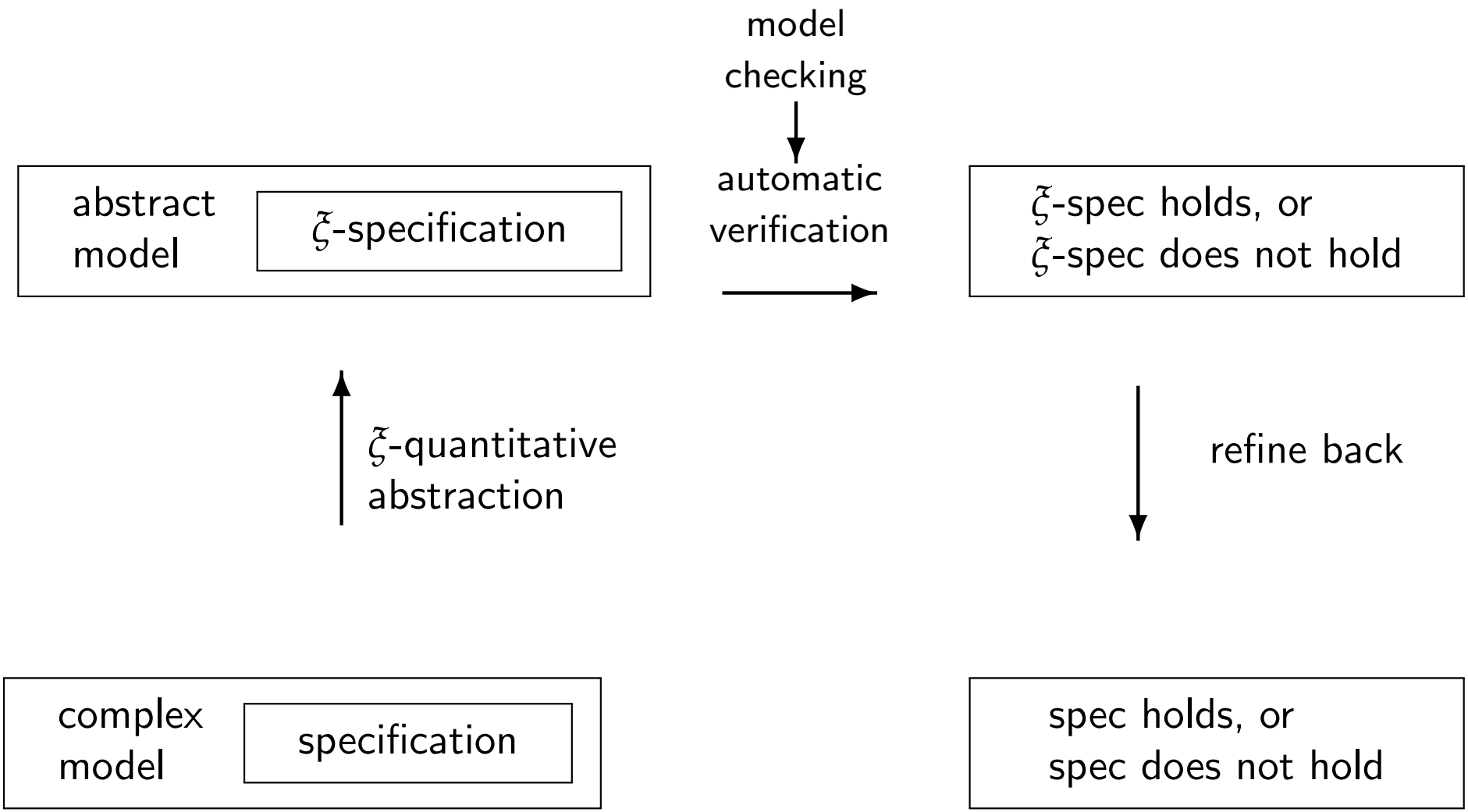
Formal abstractions



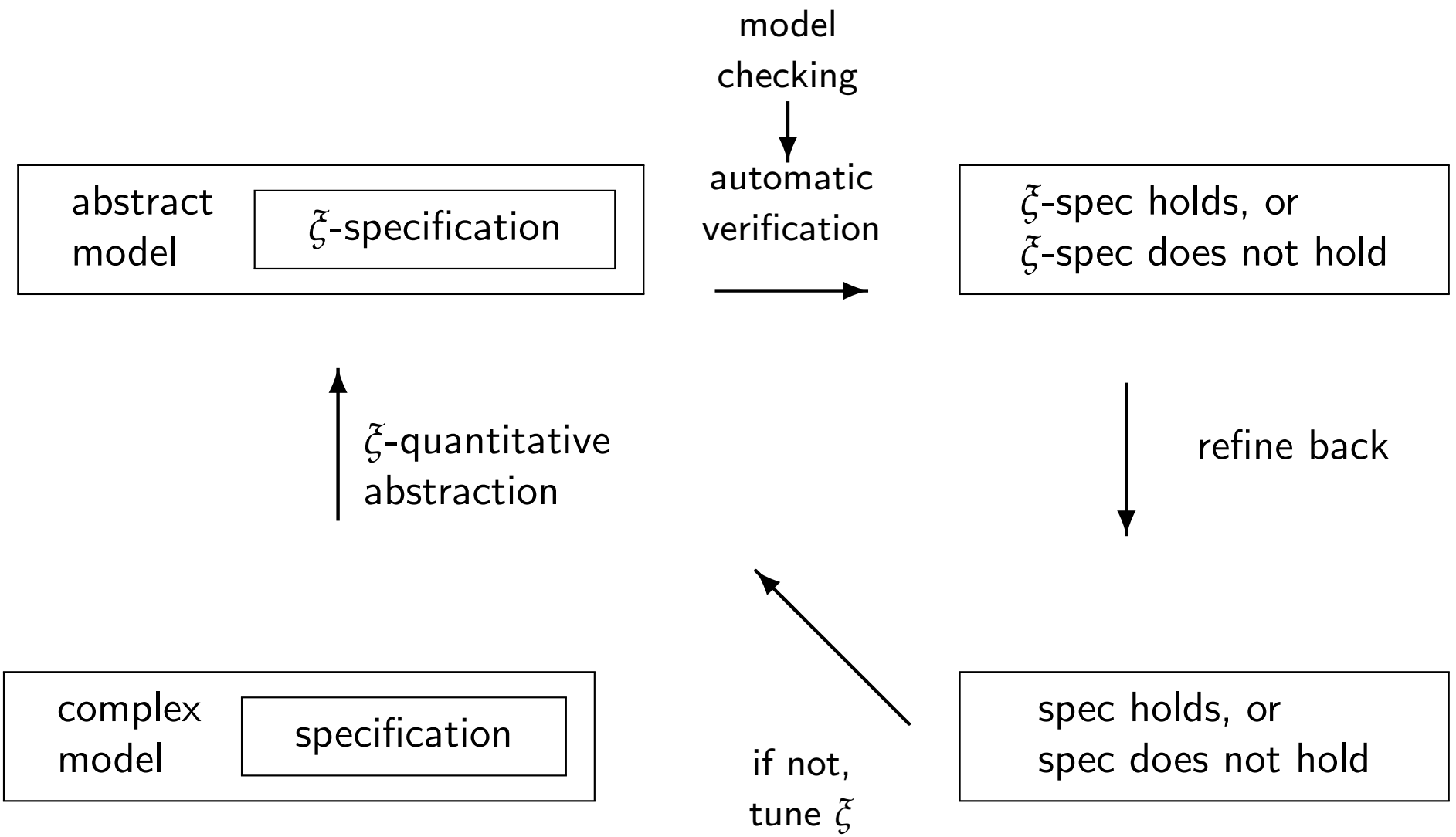
Formal abstractions



Formal abstractions



Formal abstractions



Formal abstractions: algorithm

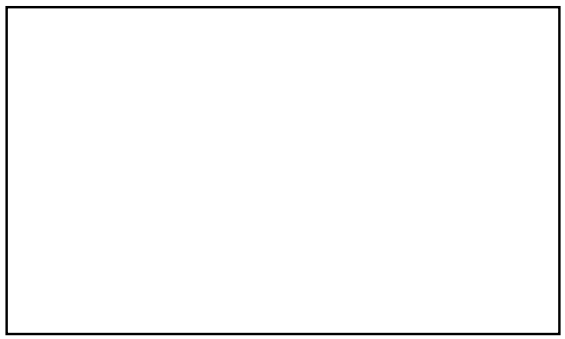
- approximate stochastic process $(\mathcal{S}, \mathcal{T})$ as MC $(S, \mathbb{T})$, where
  - $S = \{z_1, z_2, \ldots, z_p\}$ – finite set of abstract states
  - $\mathbb{T} : S \times S \to [0, 1]$ – transition probability matrix

Formal abstractions: algorithm

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

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

output: MC $(S, \mathbb{T})$



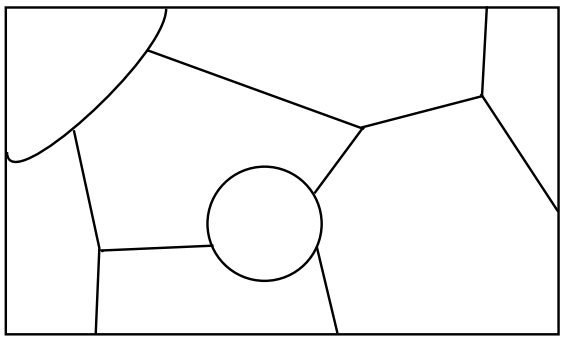
Formal abstractions: algorithm

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

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

1 select finite partition $\mathcal{S} = \cup_{i=1}^{p} S_i$

output: MC $(S, \mathbb{T})$



Formal abstractions: algorithm

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

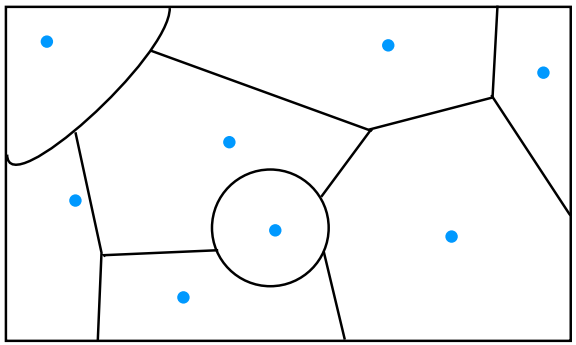
input: stochastic process $(\mathcal{S}, \mathcal{T})$

1 select finite partition $\mathcal{S} = \cup_{i=1}^{p} S_i$

2 select representative points $z_i \in S_i$

3 define finite state space $S := \{z_i, i = 1, ..., p\}$

output: MC $(S, \mathbb{T})$



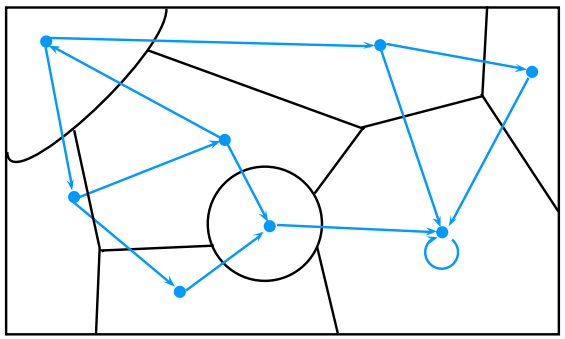
Formal abstractions: algorithm

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

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

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

output: MC $(S, \mathbb{T})$



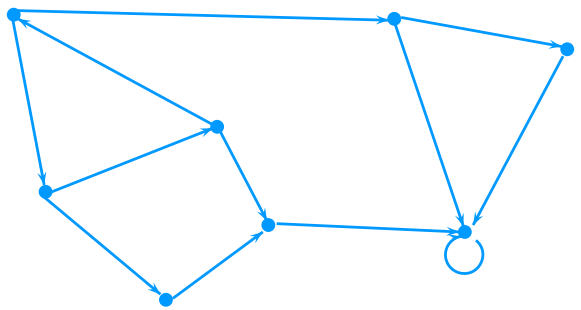
Formal abstractions: algorithm

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

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

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

output: MC $(S, \mathbb{T})$



Model checking probabilistic safety via formal abstractions

- safety set $A \subset \mathcal{S}$, time horizon N, safety level p

Model checking probabilistic safety via formal abstractions

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

$\Rightarrow$ probabilistic safe set

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

Model checking probabilistic safety via formal abstractions

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

$\Rightarrow$ probabilistic safe set

$$\begin{aligned} S(p) &= \{s \in \mathcal{S} : \mathcal{P}_s(A) \geq p\} \\ &= \{s \in \mathcal{S} : (1 - \mathcal{P}_s(A)) \leq 1 - p\} \end{aligned}$$

can be computed via

$$\begin{aligned} Z_\delta(p+\xi) &\doteq \mathrm{Sat}\left(\mathbb{P}_{\leq 1-p-\xi}\left(\mathtt{true}\ \mathsf{U}^{\leq N}\ \neg A_\delta\right)\right) \\ &= \left\{z \in S : z \models \mathbb{P}_{\leq 1-p-\xi}\left(\mathtt{true}\ \mathsf{U}^{\leq N}\ \neg A_\delta\right)\right\} \end{aligned}$$

Formal abstractions: error ξ

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

$$\exists\, 0 \leq h_S < \infty : \quad \left|\mathrm{t}(\bar{s}|s) - \mathrm{t}(\bar{s}|s')\right| \leq h_S \left\|s - s'\right\|, \quad \forall s, s', \bar{s} \in \mathcal{S}$$

Formal abstractions: error ξ

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

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

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

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

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

- N-step error *(tuneable via δ)*

  $\xi(\delta, N) = \epsilon N$

Formal abstractions: error ξ

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

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

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

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

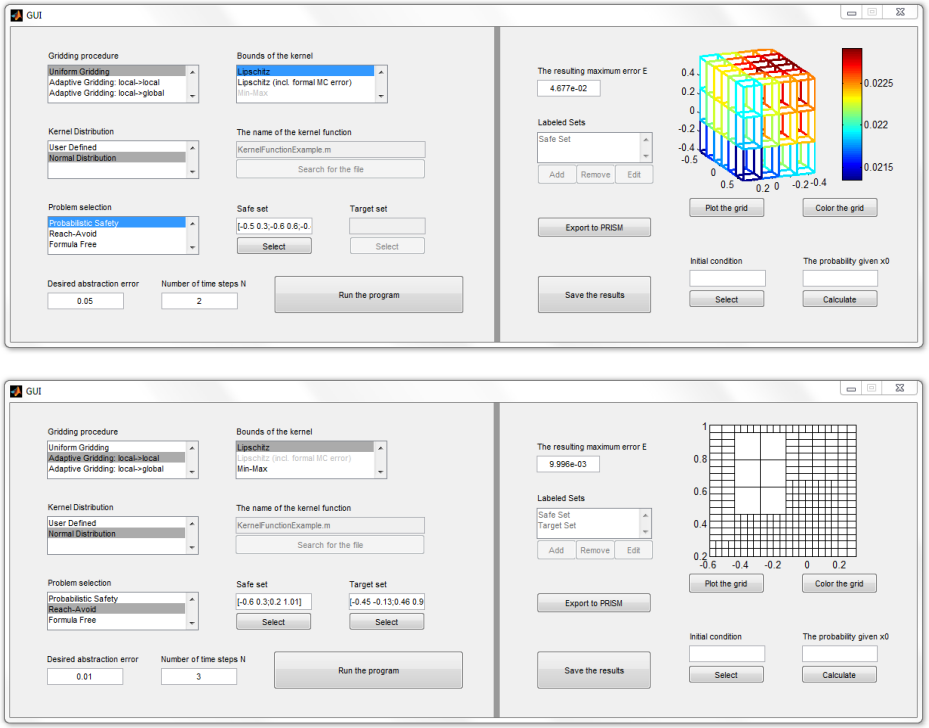
  - δ – max diameter of partition sets
  - $\mathscr{L}(A)$ – volume of set of interest

- N-step error *(tuneable via δ)*

  $\xi(\delta, N) = \epsilon N$

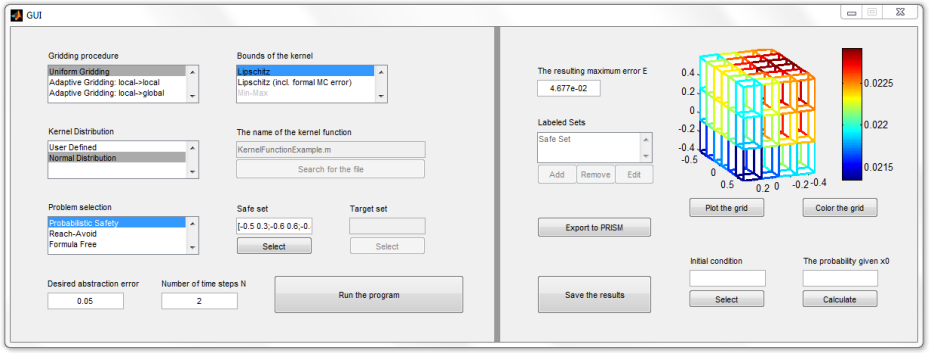
$\rightarrow$ improved and generalised error

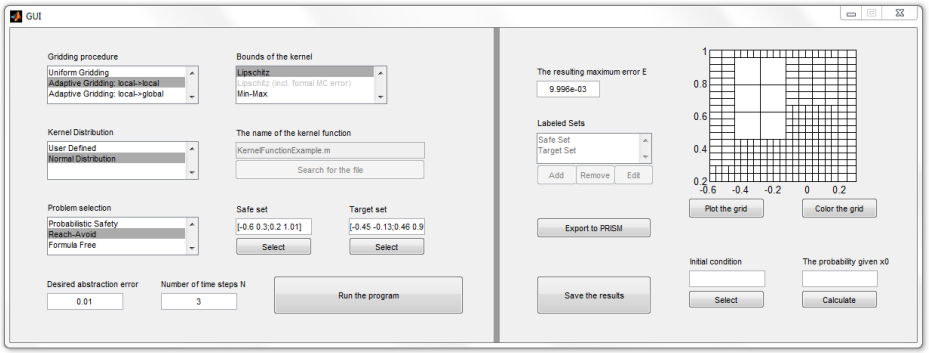
FAUST^2: software for formal abstractions



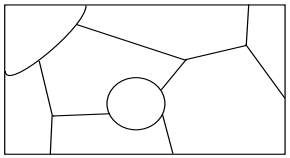
`http://sourceforge.net/projects/faust2`

FAUST^2: software for formal abstractions



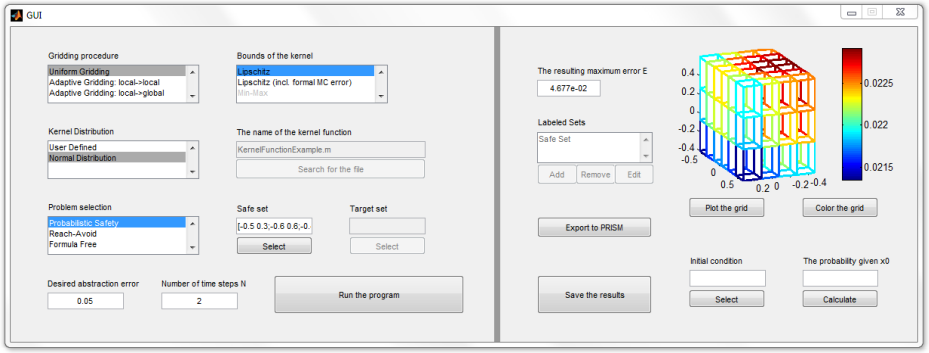


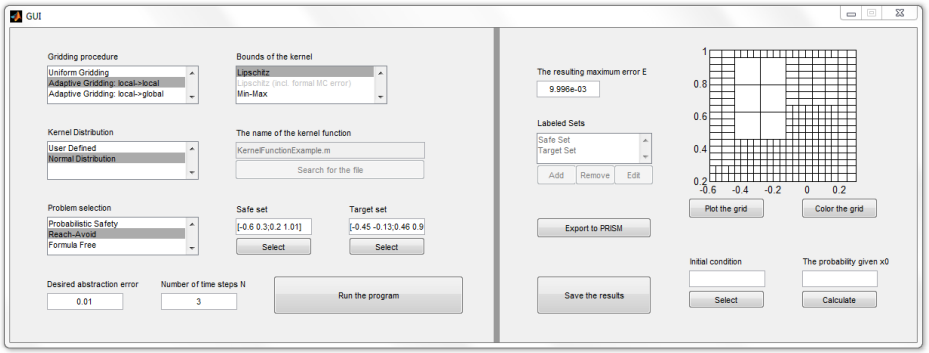
`http://sourceforge.net/projects/faust2`



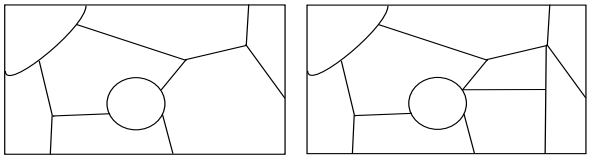
- sequential, adaptive, anytime

FAUST^2: software for formal abstractions



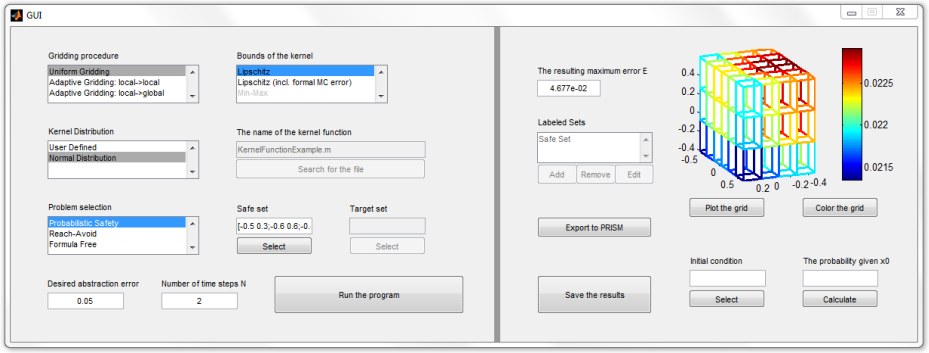


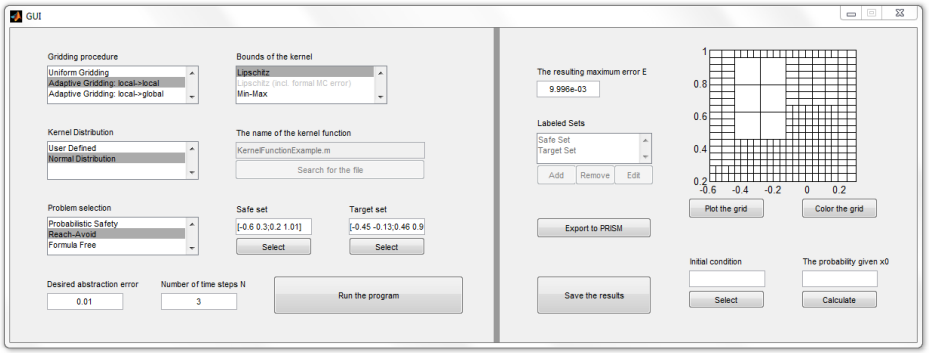
`http://sourceforge.net/projects/faust2`



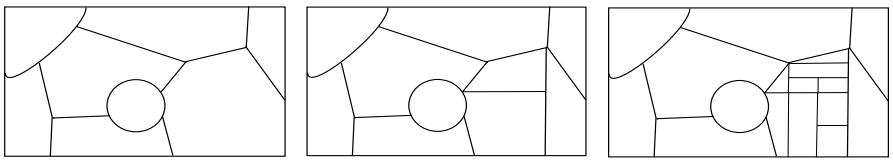
- sequential, adaptive, anytime

FAUST^2: software for formal abstractions



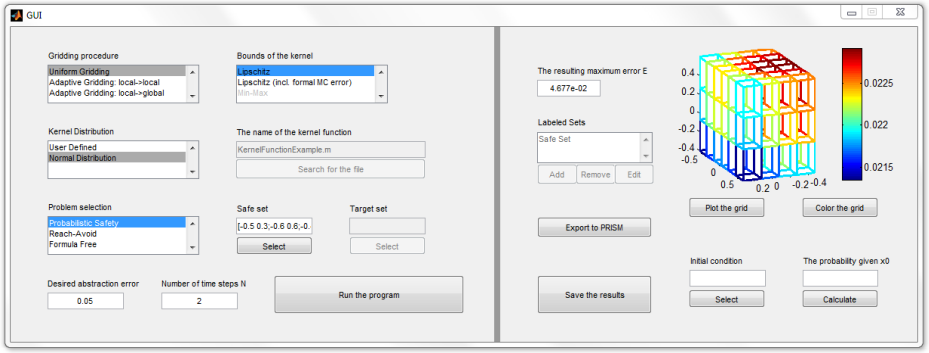


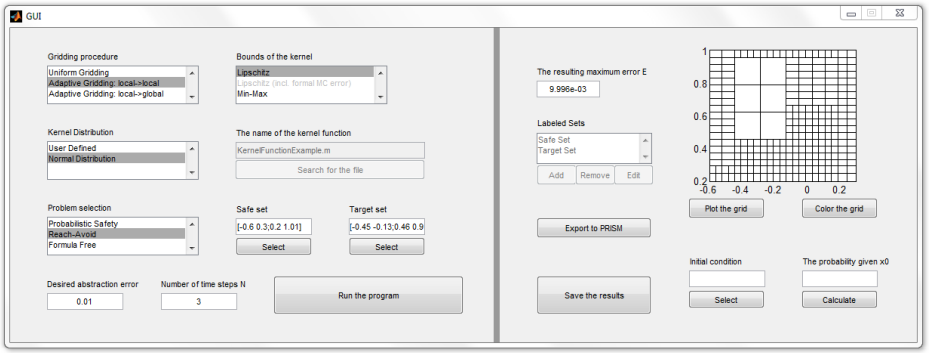
`http://sourceforge.net/projects/faust2`



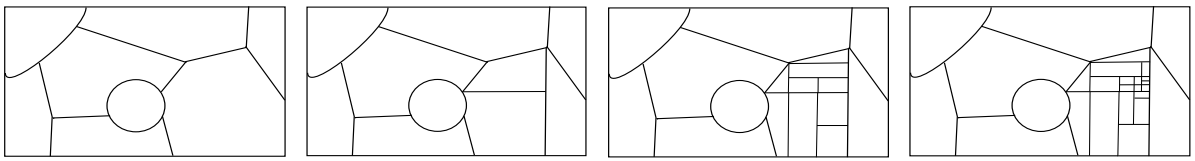
- sequential, adaptive, anytime

FAUST^2: software for formal abstractions



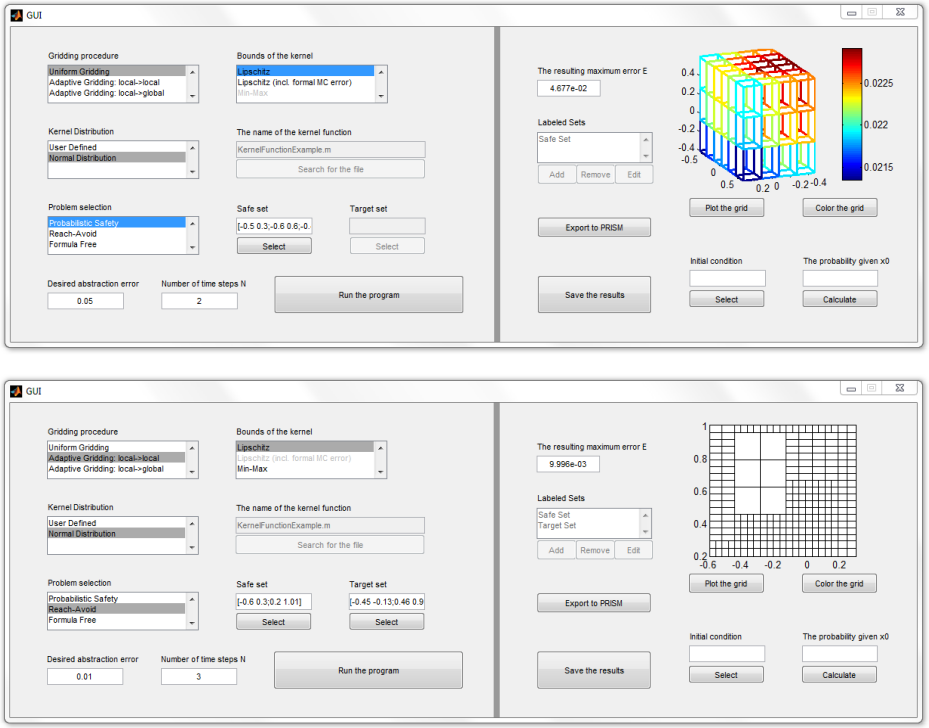


`http://sourceforge.net/projects/faust2`

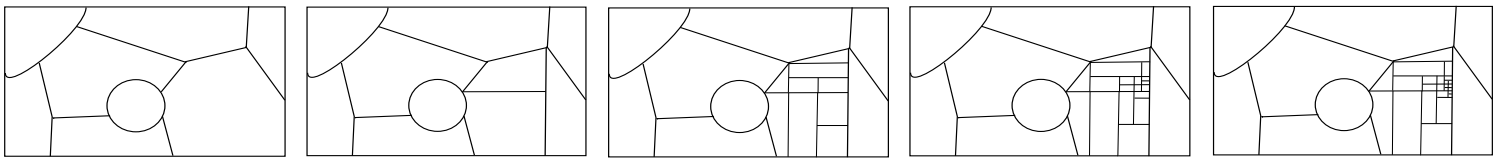


- sequential, adaptive, anytime

FAUST^2: software for formal abstractions



`http://sourceforge.net/projects/faust2`



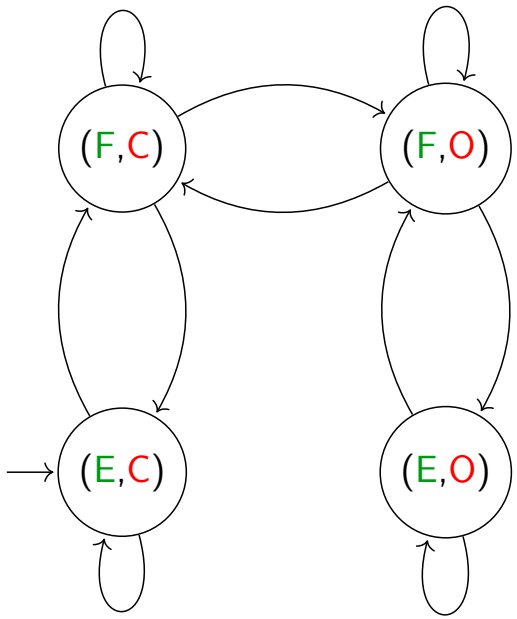
- sequential, adaptive, anytime

Building automation systems – case study

$$x_{k+1} = x_k + \frac{\Delta}{V}\left(-\mathbb{1}_{ON} m x_k + \mu_{\{O,C\}}(C_{out} - x_k)\right) + \mathbb{1}_F C_{occ} + \sigma_x w_k$$

$$y_{k+1} = y_k + \frac{\Delta}{C}\left(\mathbb{1}_{ON} m (T_{set} - y_k) + \mu_{\{O,C\}} \frac{1}{R}(T_{out} - y_k)\right) + \mathbb{1}_F T_{occ,k} + \sigma_y w_k$$

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



Building automation systems – case study

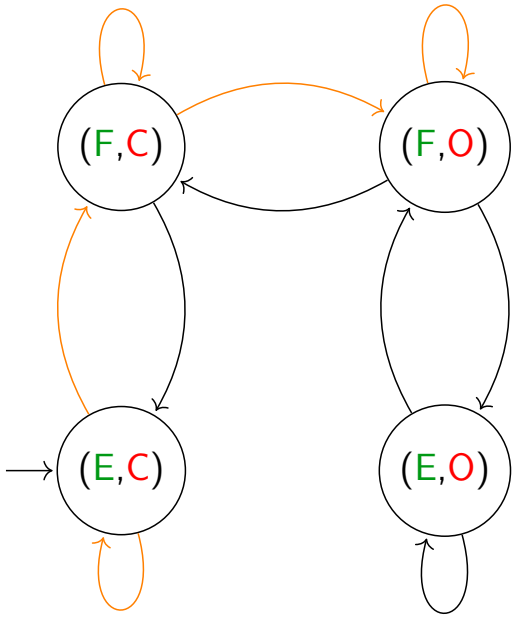
- safe set $A = [300\ 700]ppm \times [19\ 21]^oC$
- air circulation: closed-loop control policy at $k+1$

$$\begin{cases} OFF & \text{if } (x_k, y_k) \leq A \\ ON & \text{if } (x_k, y_k) \geq A \\ \text{stay put} & \text{else} \end{cases}$$

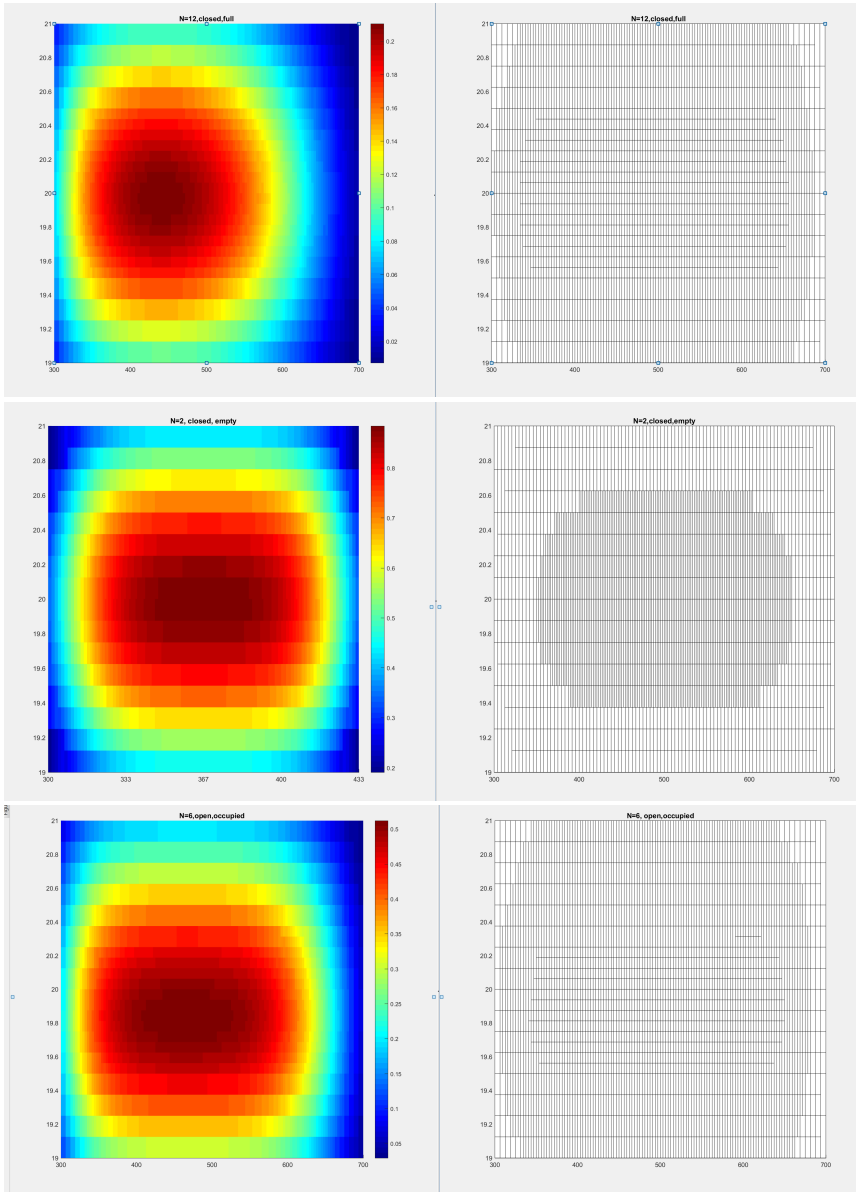
- specification:

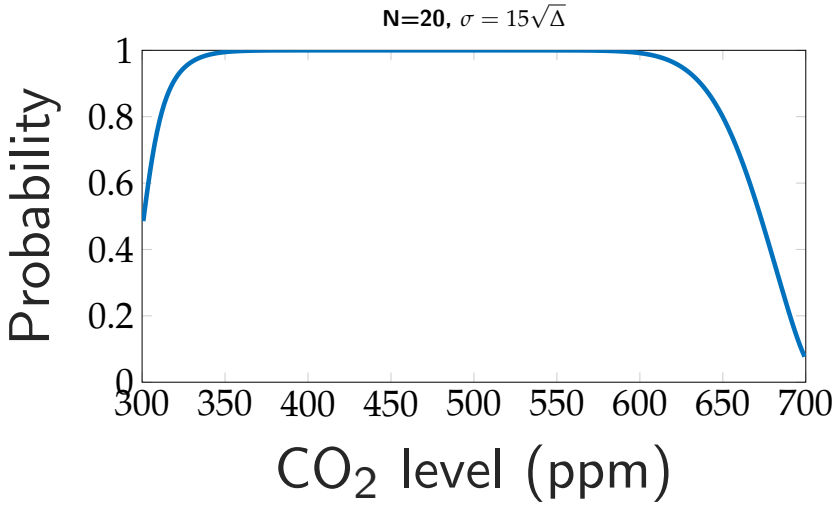
$$\mathbb{P}_{=?}\left[\Box^{\leq 20}(x, y) \in A\right]$$

- 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)



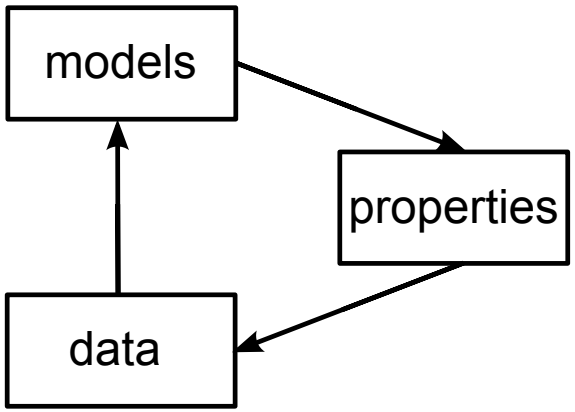
Building automation systems – case study



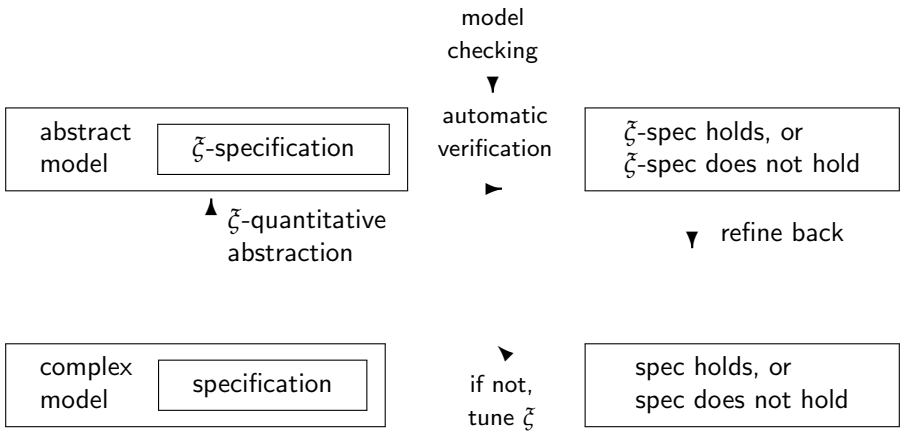


Take away message

- verification of physical systems: Integration of learning and verification



- verification of complex models with formal abstractions



Acknowledgments

My students: V. Wijesurija, N. Cauchi, E. Polgreen, A. Peruffo, K. Lesser, M. Zamani, S. Haesaert, I. Tkachev, D. Adzkiya, S. Soudjani

and collaborators

Selected journal references

E. Polgreen, V. Wijesuriya, S. Haesaert and A. Abate, "Automated Experiment Design for Efficient Verification of Parametric Markov Decision Processes," QEST17, 2017.

E. Polgreen, V. Wijesuriya, S. Haesert and A. Abate, "Data-efficient Bayesian verification of parametric Markov chains," QEST16, LNCS 9826, pp. 35–51, 2016.

S. Haesaert, S.E.Z. Soudjani, and A. Abate, "Verification of general Markov decision processes by approximate similarity relations and policy refinement," SIAM Journal on Control and Optimisation, vol. 55, nr. 4, pp. 2333-2367, 2017.

I. Tkachev, A. Mereacre, J.-P. Katoen, and A. Abate, "Quantitative Model Checking of Controlled Discrete-Time Markov Processes," Information and Computation, vol. 253, nr. 1, pp. 1–35, 2017.

S. Haesaert, at al., P.M.J. V.d. Hof, and A. Abate, "Data-driven and Model-based Verification via Bayesian Identification and Reachability Analysis," Automatica, vol. 79, pp. 115–126, 2017.

S.E.Z. Soudjani and A. Abate, "Aggregation and Control of Populations of Thermostatically Controlled Loads by Formal Abstractions," IEEE Transactions on Control Systems Technology. vol. 23, nr. 3, pp. 975–990, 2015.

S.E.Z. Soudjani and A. Abate, "Quantitative Approximation of the Probability Distribution of a Markov Process by Formal Abstractions," Logical Methods in Computer Science, Vol. 11, nr. 3, Oct. 2015.

M. Zamani, P. Mohajerin Esfahani, R. Majumdar, A. Abate, and J. Lygeros, "Symbolic control of stochastic systems via approximately bisimilar finite abstractions," IEEE Transactions on Automatic Control, vol. 59 nr. 12, pp. 3135-3150, Dec. 2014.

I. Tkachev and A. Abate, "Characterization and computation of infinite horizon specifications over Markov processes," Theoretical Computer Science, vol. 515, pp. 1-18, 2014.

S. Soudjani and A. Abate, "Adaptive and Sequential Gridding for Abstraction and Verification of Stochastic Processes," SIAM Journal on Applied Dynamical Systems, vol. 12, nr. 2, pp. 921-956, 2013.

A. Abate, et al., "Approximate Model Checking of Stochastic Hybrid Systems," European Journal of Control, 16(6), 624-641, 2010.

A. Abate, et al., "Probabilistic Reachability and Safety Analysis of Controlled Discrete-Time Stochastic Hybrid Systems," Automatica, 44(11), 2724-2734, Nov. 2008.

Thank you for your attention

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