Automated Verification and Control Synthesis of Complex Models

Alessandro Abate

Department of Computer Science, University of Oxford

www.oxcav.org

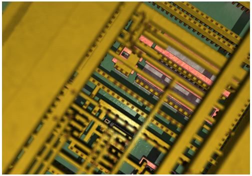
Simons Institute - April 2021

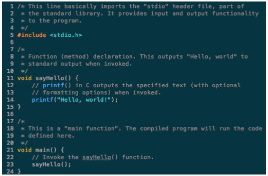
Formal verification: successes and frontiers

UNIVERSITY OF OXFORD

- industrial impact in checking correctness of

protocols, hardware circuits, and software





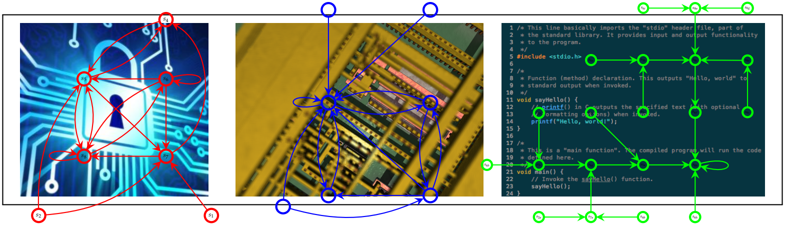
```c
/* This line basically imports the "stdio" header file, part of
 * the standard library. It provides input and output functionality
 * to the program.
 */
#include <stdio.h>

/*
 * 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();
}
```

- model-based, automated, and sound guarantees (formal certificates)

Formal verification: successes and frontiers

- industrial impact in checking correctness of

protocols, hardware circuits, and software

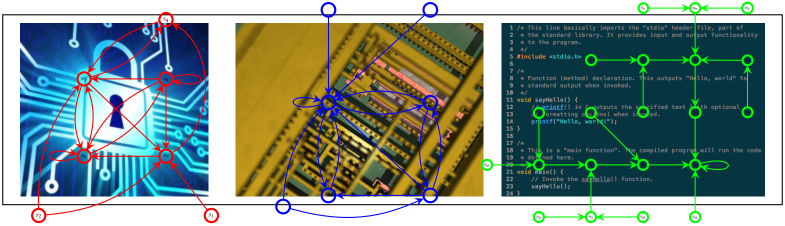


- model-based, automated, and sound guarantees (formal certificates)

Formal verification: successes and frontiers

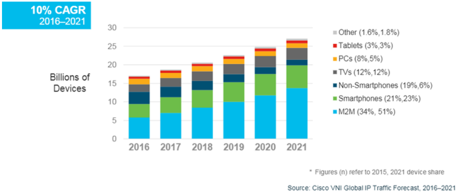
- industrial impact in checking correctness of

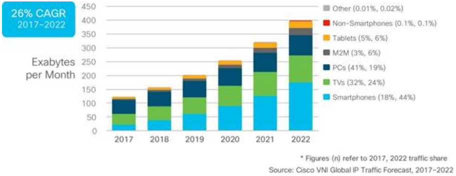
protocols, hardware circuits, and software



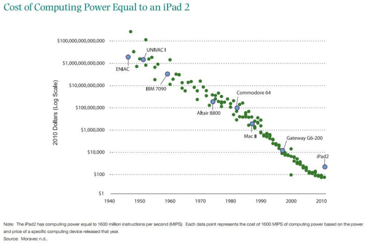
- model-based, automated, and sound guarantees (formal certificates)

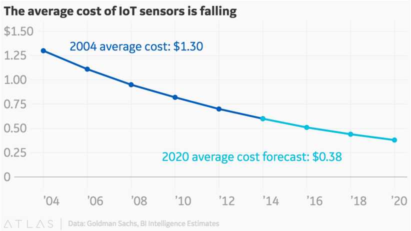
Formal verification and control in the real world





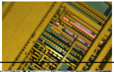
[courtesy M. Zamani]

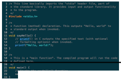




- tech trends: advances in sensing, networking and embedded computation

Formal verification and control in the real world

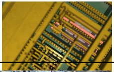
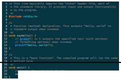




Formal verification and control in the real world

1. integration of learning from data within model-based verification & control
("learning for verification and control")

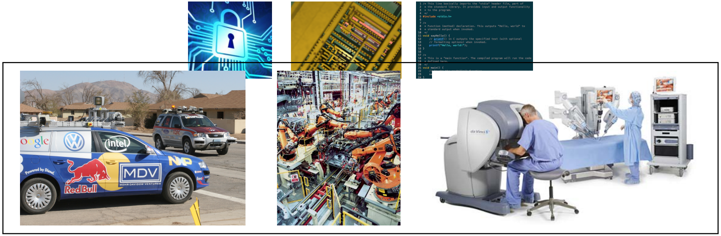
2. certified reinforcement learning for policy synthesis
("certified learning")



Formal verification and control in the real world

- verification and control of complex models
  - dynamical models with uncertainty, noise
  - via formal abstractions

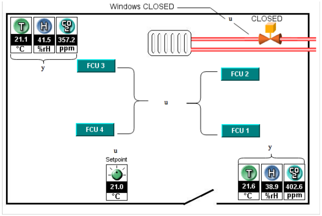


Building automation systems - a CPS exemplar

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

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

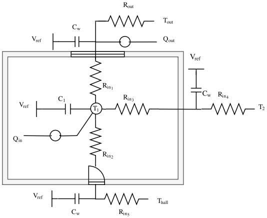
Building automation systems - a CPS exemplar



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 – a complex model

- model CO_2 dynamics, coupled with temperature evolution

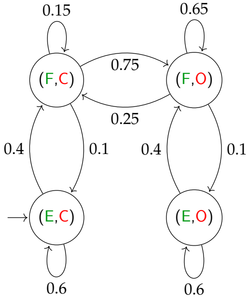
$$x_{k+1} = x_k + \frac{\Delta}{V}\left(-\mathbb{1}_{ON} m x_k + \mu_{\{O,C\}}(C_{out} - x_k)\right) + \mathbb{1}_F C_{occ} + \sigma_x w_k$$

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

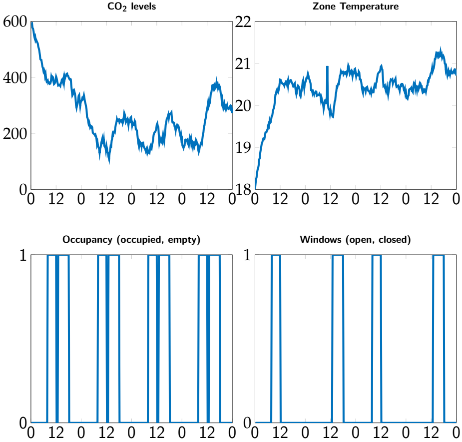
where $T_{occ,k} = \nu x_k + \zeta$

- 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

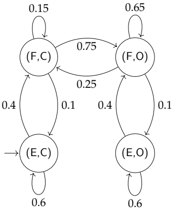
finite-space Markov chain

$(S, \mathbb{T})$

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

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

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



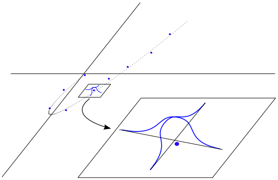
uncountable-space Markov process

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

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

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

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



Probabilistic model checking of complex models

- general specifications expressed as PCTL formulae, e.g.
- simplest instance: probabilistic safety is *the probability that the execution, started at s, stays in safe set A during the time horizon $[0, N]$*

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

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

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

- PCTL formula: $\mathbb{P}_{\leq 1-p} \left(\texttt{true}\ \mathsf{U}^{\leq N} \neg A\right)$

Probabilistic model checking of complex models

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

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

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

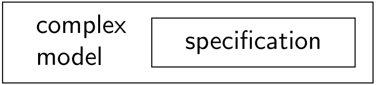
- select $p \in [0, 1]$; probabilistic safe set with safety level p is

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

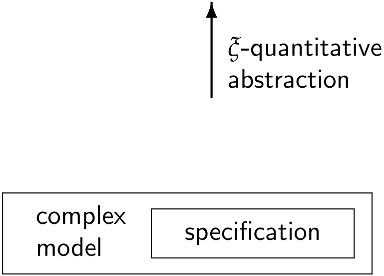
- PCTL formula: $\mathbb{P}_{\leq 1-p}\left(\texttt{true}\ \mathsf{U}^{\leq N} \neg A\right)$

- $\mathcal{P}_s(A)$ can be fully characterised (and optimised)
- issues with computation of $\mathcal{P}_s(A)$ and of $S(p)$

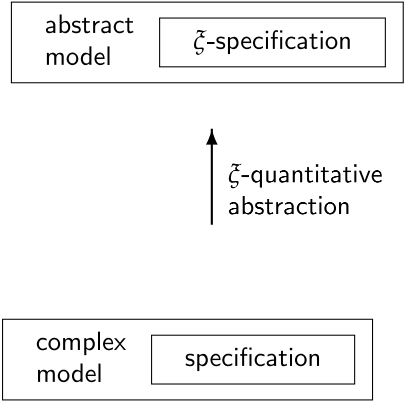
Formal abstractions



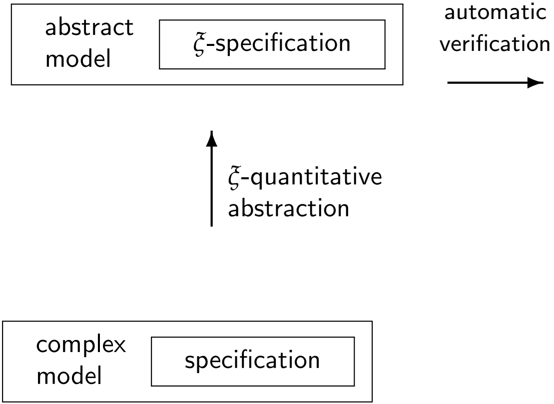
Formal abstractions



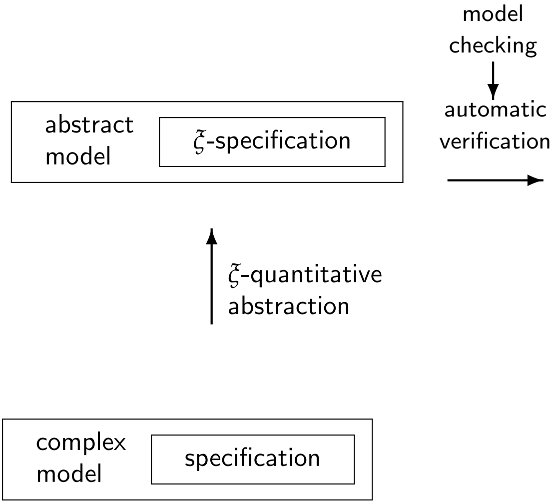
Formal abstractions



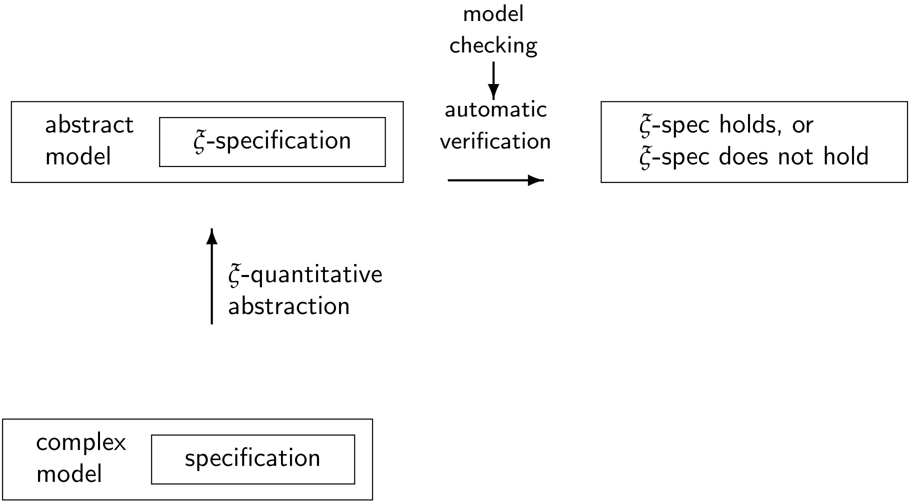
Formal abstractions



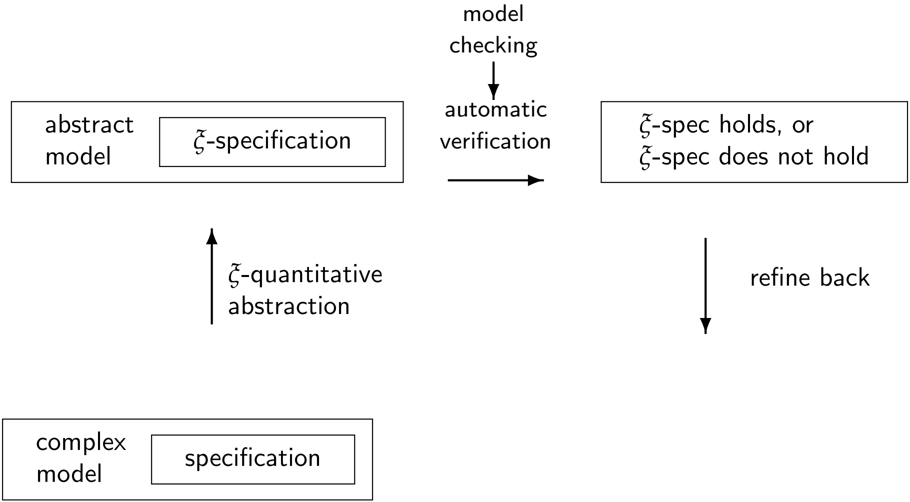
Formal abstractions



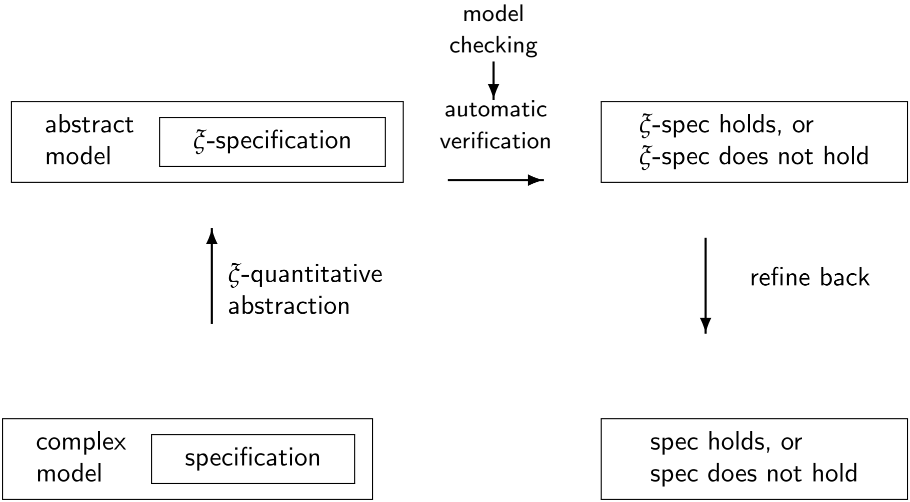
Formal abstractions



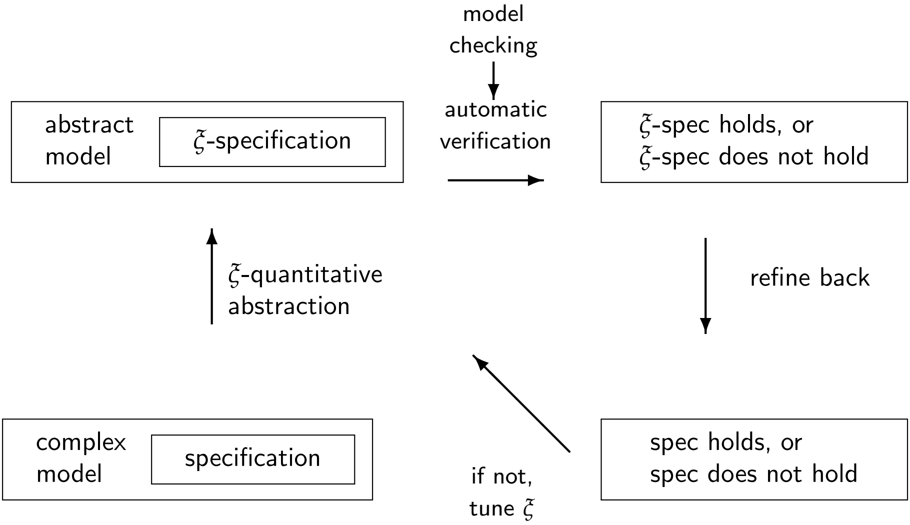
Formal abstractions



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
    - $\mathbb{T} : \mathcal{S} \times \mathcal{S} \to [0, 1]$ – 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
  - $\mathbb{T} : \mathcal{S} \times \mathcal{S} \to [0, 1]$ – transition probability matrix
- algorithm:

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

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

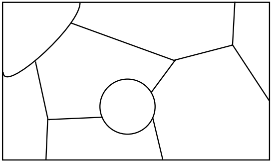
Formal abstractions: algorithm

- 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
  - $\mathbb{T} : \mathcal{S} \times \mathcal{S} \to [0,1]$ – transition probability matrix
- algorithm:

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

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

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:

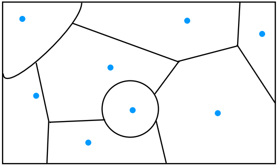
input: stochastic process $(\mathcal{S}, \mathcal{T})$

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 $S := \{z_i, i = 1, ..., p\}$

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

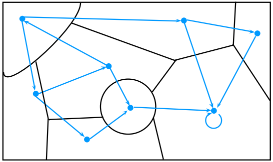
1 select finite partition $\mathcal{S} = \cup_{i=1}^{p} S_i$

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 $(S, \mathbb{T})$



Formal abstractions: algorithm

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

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

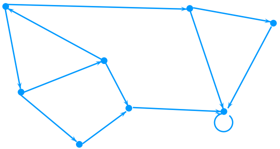
1 select finite partition $\mathcal{S} = \cup_{i=1}^{p} S_i$

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

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

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

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

Formal abstractions: error ζ

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

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

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 ζ

- 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)*

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

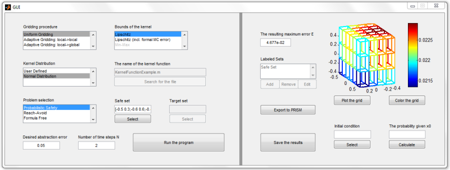
  - δ – max diameter of partition sets
  - $\mathscr{L}(A)$ – volume of set of interest

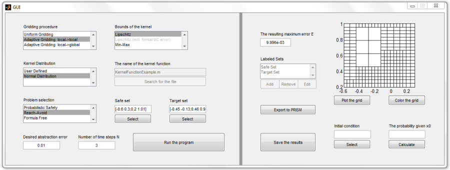
- N-step error *(tuneable via δ)*

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

$\rightarrow$ improved and generalised error

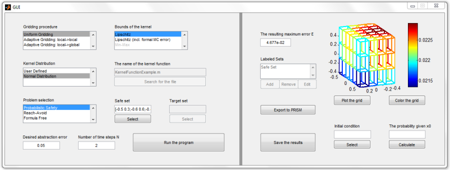
FAUST^2: software for formal abstractions

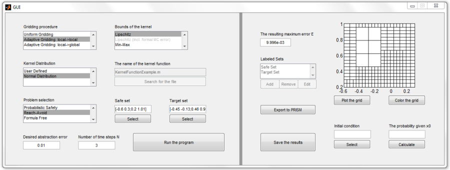




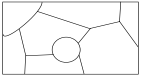
http://sourceforge.net/projects/faust2

FAUST^2: software for formal abstractions



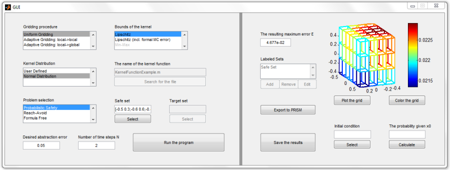


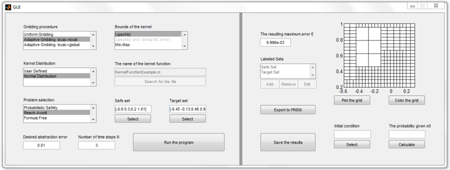
`http://sourceforge.net/projects/faust2`



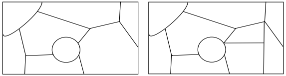
- sequential, adaptive, anytime

FAUST^2: software for formal abstractions



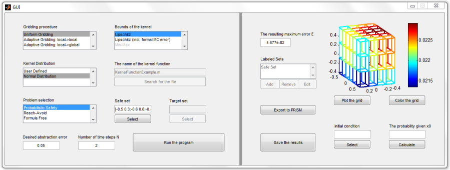


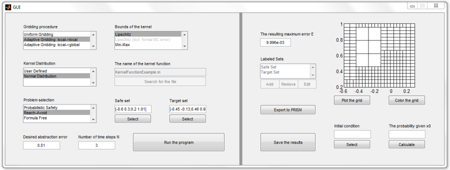
http://sourceforge.net/projects/faust2



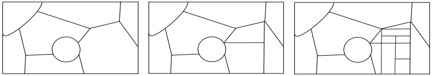
- sequential, adaptive, anytime

FAUST^2: software for formal abstractions



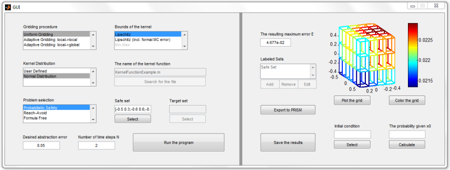


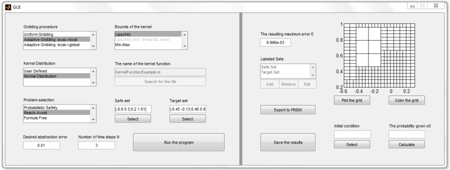
http://sourceforge.net/projects/faust2



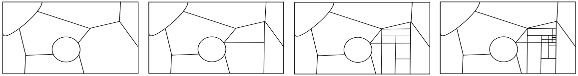
- sequential, adaptive, anytime

FAUST^2: software for formal abstractions



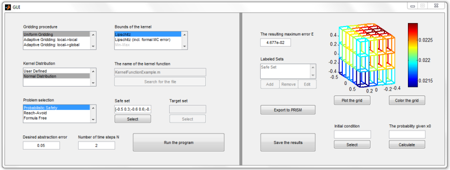


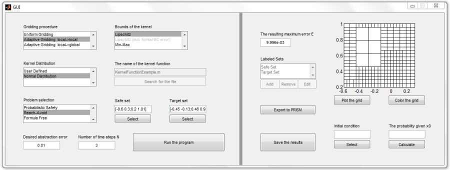
http://sourceforge.net/projects/faust2



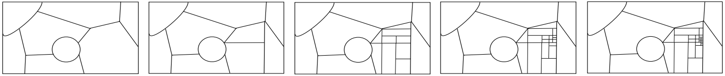
- sequential, adaptive, anytime

FAUST^2: software for formal abstractions





http://sourceforge.net/projects/faust2



- sequential, adaptive, anytime

StocHy: software for formal abstractions

verification

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

`gitlab.com/natchi92/StocHy`

StocHy: software for formal abstractions

UNIVERSITY OF OXFORD

verification

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

synthesis

- abstraction based
- optimisation via sparse matrices

StocHy

`gitlab.com/natchi92/StocHy`

StocHy: software for formal abstractions

verification

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

simulation

- automatically generates statistics
- visualisation via time varying histograms

StocHy

synthesis

- abstraction based
- optimisation via sparse matrices

features

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

gitlab.com/natchi92/StocHy

StocHy: software for formal abstractions

verification

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

simulation

- automatically generates statistics
- visualisation via time varying histograms

synthesis

- abstraction based
- optimisation via sparse matrices

StocHy

features

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

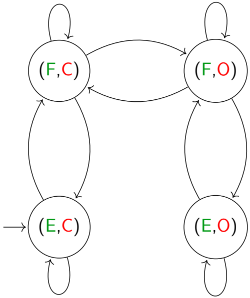
`gitlab.com/natchi92/StocHy`

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

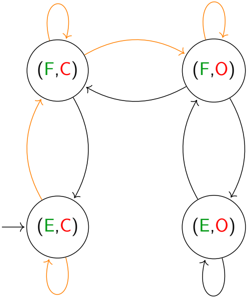
- safe set $A = [300\ 700]ppm \times [19\ 21]^oC$
- air circulation: closed-loop control policy at $k+1$

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

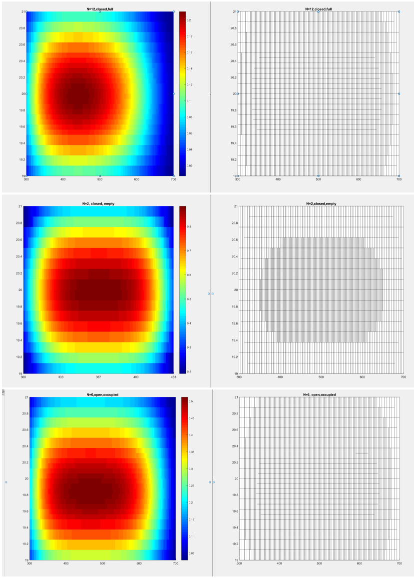
- specification:

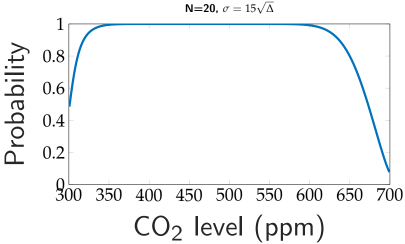
$$\mathbb{P}_{=?}\left[\Box^{\leq 20}(x, y) \in A\right]$$

- 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





Selected journal references

L. Laurenti, M. Lahijanian, A. Abate, L. Cardelli and M. Kwiatkowska, "Formal and Efficient Control Synthesis for Continuous-Time Stochastic Processes," IEEE Transactions on Automatic Control, vol. 66, no. 1, pp. 17-32, Jan 2021.

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, N. Cauchi and A. Abate, "Certified policy synthesis for general Markov decision processes: An application in building automation systems," Performance Evaluation, vol. 117, pp. 75-103, 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`