



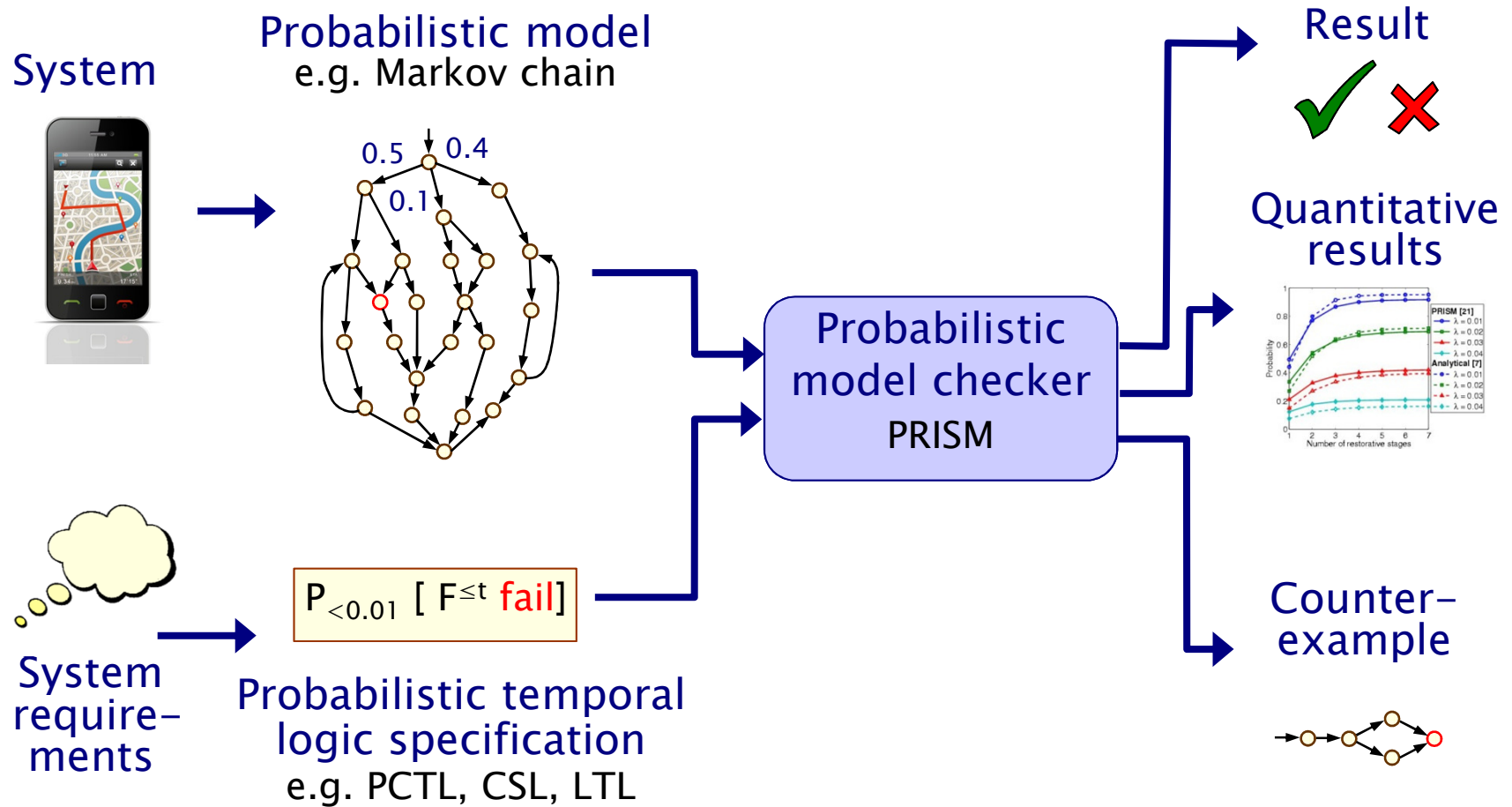
Parameter synthesis for probabilistic real-time systems

Marta Kwiatkowska

Department of Computer Science, University of Oxford

SynCoP 2015, London

Quantitative (probabilistic) verification



Historical perspective

- 
- 
- First algorithms proposed in 1980s
 - [Vardi, Courcoubetis, Yannakakis, ...]
 - algorithms [Hansson, Jonsson, de Alfaro] & first implementations
 - 2000: tools ETMCC (MRMC) & PRISM released
 - PRISM: efficient extensions of symbolic model checking [Kwiatkowska, Norman, Parker, ...]
 - ETMCC (now MRMC): model checking for continuous-time Markov chains [Baier, Hermanns, Haverkort, Katoen, ...]
 - Now mature area, of industrial relevance
 - successfully used by non-experts for many application domains, but full **automation** and good **tool support** essential
 - distributed algorithms, communication protocols, security protocols, biological systems, quantum cryptography, planning...
 - genuine **flaws** found and corrected in real-world systems



Quantitative probabilistic verification

- What's involved

- specifying, extracting and building of quantitative models
- graph-based analysis: reachability + qualitative verification
- numerical solution, e.g. linear equations/linear programming
- simulation-based statistical model checking
- typically computationally more **expensive** than the non-quantitative case

- The state of the art

- efficient techniques for a range of probabilistic real-time models
- feasible for models of up to **10^7 states** (10^{10} with symbolic)
- abstraction refinement (CEGAR) methods
- multi-objective verification
- assume-guarantee compositional verification
- **tool support** exists and is widely used, e.g. **PRISM**, **MRMC**

Tool support: PRISM

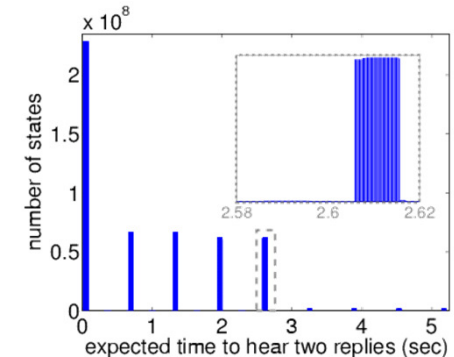
- **PRISM: Probabilistic symbolic model checker**
 - developed at Birmingham/Oxford University, since 1999
 - free, open source software (GPL), runs on all major OSs
- **Support for:**
 - models: DTMCs, CTMCs, MDPs, PTAs, SMGs, ...
 - properties: PCTL/PCTL*, CSL, LTL, rPATL, costs/rewards, ...
- **Features:**
 - simple but flexible high-level modelling language
 - user interface: editors, simulator, experiments, graph plotting
 - multiple efficient model checking engines (e.g. symbolic)
- **Many import/export options, tool connections**
 - MRMC, INFAMY, DSD, Petri nets, Matlab, ...
- **See: <http://www.prismmodelchecker.org/>**



Quantitative verification in action

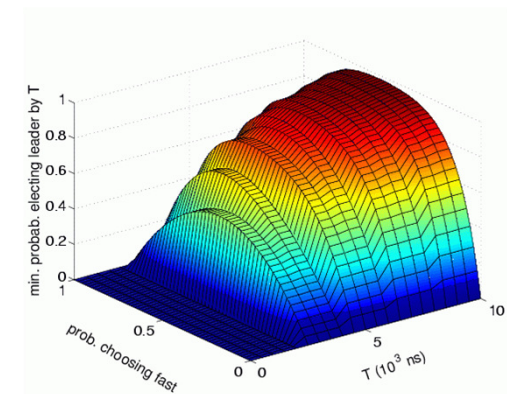
- Bluetooth device discovery protocol

- frequency hopping, randomised delays
- low-level model in PRISM, based on detailed Bluetooth reference documentation
- numerical solution of 32 Markov chains, each approximately 3 billion states
- identified **worst-case** time to hear one message



- FireWire root contention

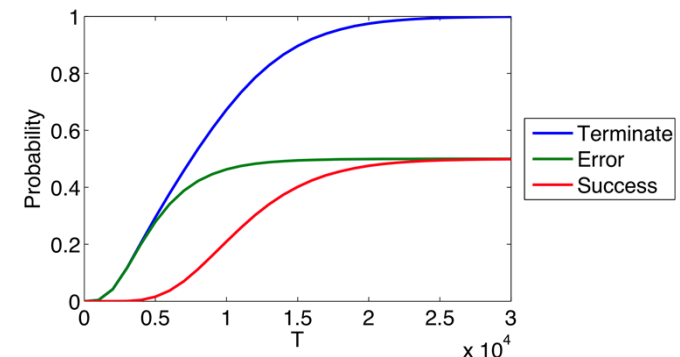
- wired protocol, uses randomisation
- model checking using PRISM
- optimum probability of leader election by time T for various coin biases
- demonstrated that a **biased coin** can improve performance



Quantitative verification in action

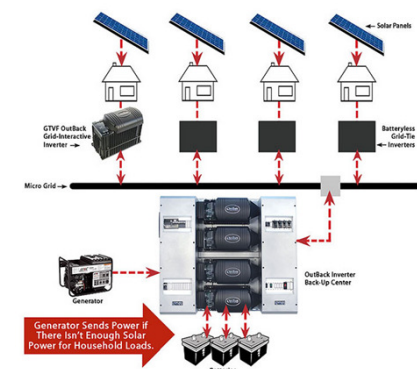
- DNA transducer gate [Lakin et al, 2012]

- DNA computing with a restricted class of DNA strand displacement structures
- transducer design due to Cardelli
- **automatically** found and fixed design error, using Microsoft's DSD and PRISM



- Microgrid demand management protocol [TACAS12,FMSD13]

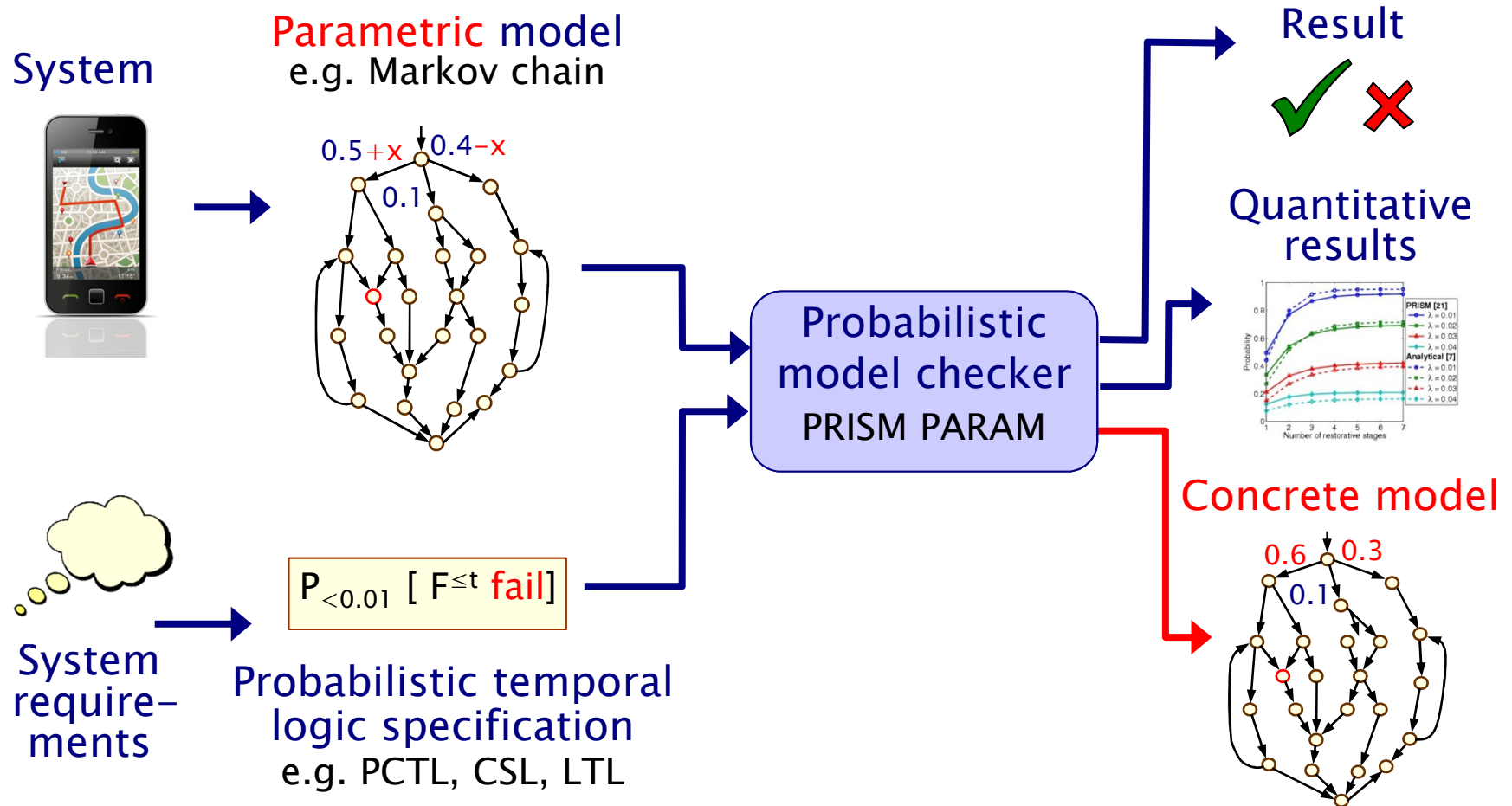
- designed for households to actively manage demand while accessing a variety of energy sources
- **found and fixed a flaw** in the protocol, due to lack of punishment for selfish behaviour
- implemented in PRISM-games



From verification to synthesis...

- Majority of research to date has focused on verification
 - scalability and performance of algorithms
 - extending expressiveness of models and logics
 - real-world case studies
- Automated verification aims to establish if a property holds for a given model
- What to do if quantitative verification fails?
 - counterexamples difficult to represent compactly
- Can we synthesise a model so that a property is satisfied?
 - difficult...
- Simpler variants of synthesis:
 - **parameter synthesis**
 - controller/strategy synthesis

Quantitative parameter synthesis



Parametric model checking in PRISM

- Parametric Markov chain models in PRISM
 - **probabilistic** parameters expressed as unevaluated constants
 - e.g. `const double x;`
 - transition probabilities are **expressions** over parameters, e.g. `0.4 + x`
- Properties are given in PCTL, with parameter constants
 - new construct `constfilter (min, x1*x2, phi)`
 - filters over parameter values, rather than states
- Implemented in 'explicit' engine
 - returns **mapping** from parameter regions (e.g. `[0.2,0.3],[−2,0]`) to rational functions over the parameters
 - filter properties used to find parameter values that **optimise** the function
 - **reimplementation** of PARAM 2.0 [Hahn et al]

This lecture...

- Parameter synthesis for **probabilistic real-time systems**
- The **parameter synthesis problem** we consider
 - given a parametric model and property ϕ
 - find the **optimal** parameter values, with respect to an **objective function** O , such that the property ϕ is satisfied, if such values exist
- Parameters: timing delays, rates
- Objectives: optimise probability, reward/volume

Overview

1. Timed automata: find **optimal timing delays** [EMSOF2014]
 - solution: constraint solving, discretisation + sampling
 2. Probabilistic timed automata: find delays to **optimise probability** [RP2014]
 - solution: parametric symbolic abstraction-refinement
 3. Continuous-time Markov chains: find **optimal rates** [CMSB2014]
 - solution: constraint solving, uniformisation + sampling
- Focus on practical implementation and real-world applications

1. Optimal timing delays

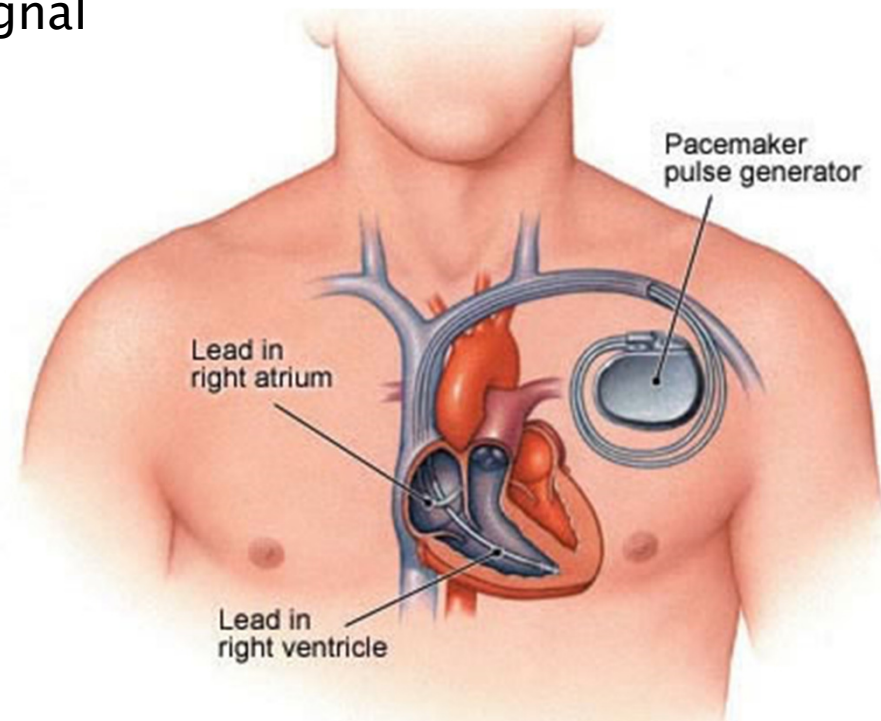
- Models: networks of timed I/O automata
 - dense real-time
 - extend with parameters on **guards**
 - **synchronise** on matching input-output
 - **no nondeterminism** (add priority and urgency of output)
- Properties: Counting Metric Temporal Logic (CMTL)
 - linear-time, real-valued time bounds
 - **event counting** in an interval of time, reward weighting

$$\Box^{[0,\tau]}(\#_0^\tau \text{Vget} \geq B_1 \wedge \#_0^\tau \text{Vget} \leq B_2)$$

$$1 \cdot \#_0^\tau \text{AP} + 2 \cdot \#_0^\tau \text{VP} \leq E$$

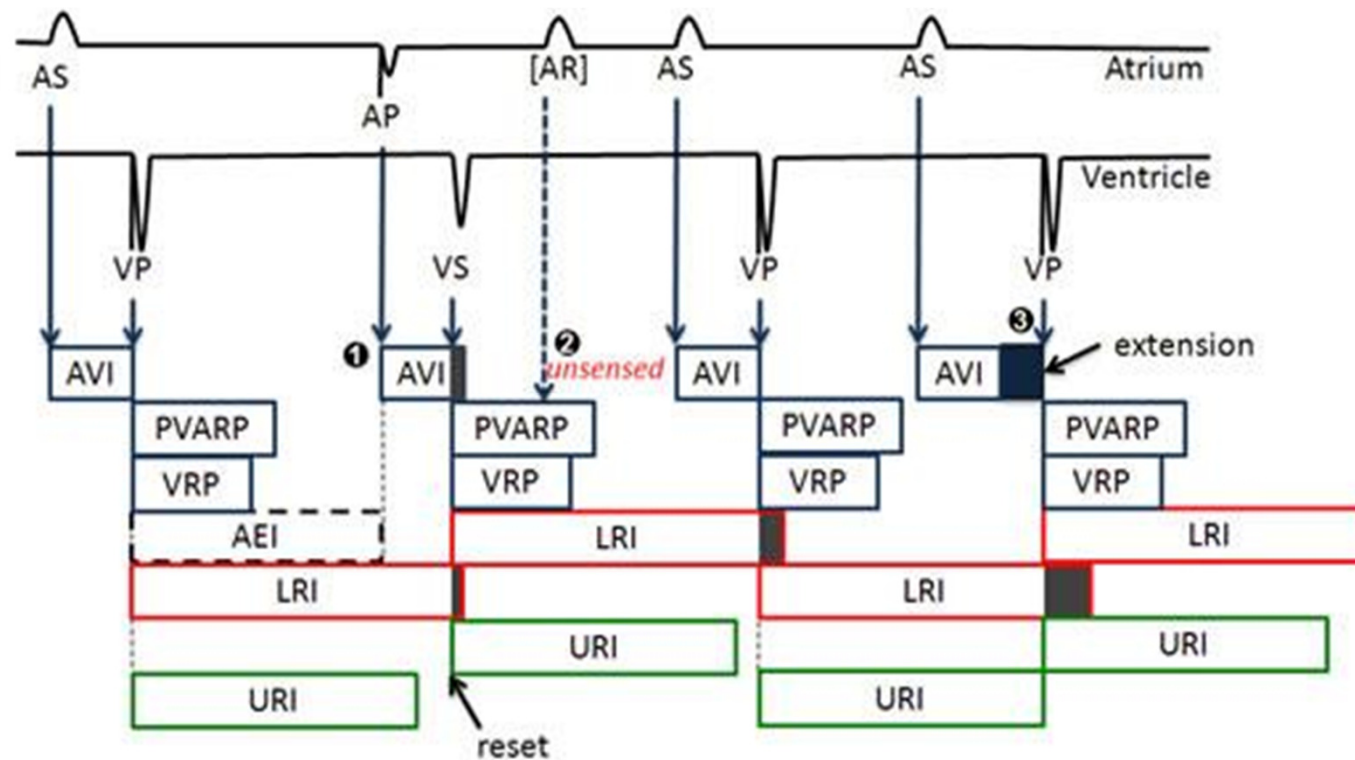
Implantable pacemaker

- How it works
 - **reads** electrical (action potential) signals through sensors placed in the right atrium and right ventricle
 - monitors the **timing** of heart beats and local electrical activity
 - generates **artificial** pacing signal as necessary
- Real-time system!
- Core specification by Boston Scientific
- Basic pacemaker can be modelled as a network of timed automata [Ziang et al]



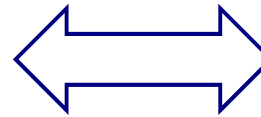
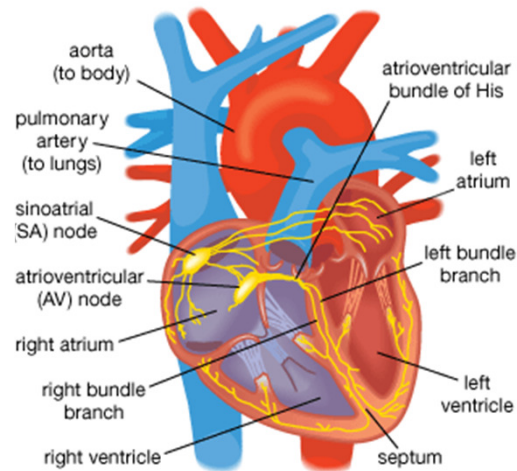
Pacemaker timing cycle

- Atrial and ventricular events



Quantitative verification for pacemakers

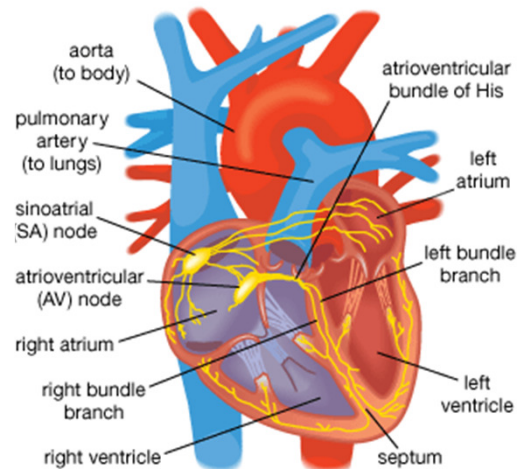
- Model the pacemaker and the heart as timed I/O automata
- Compose and **verify**



Copyright ©2008 Boston Scientific Corporation All rights reserved.

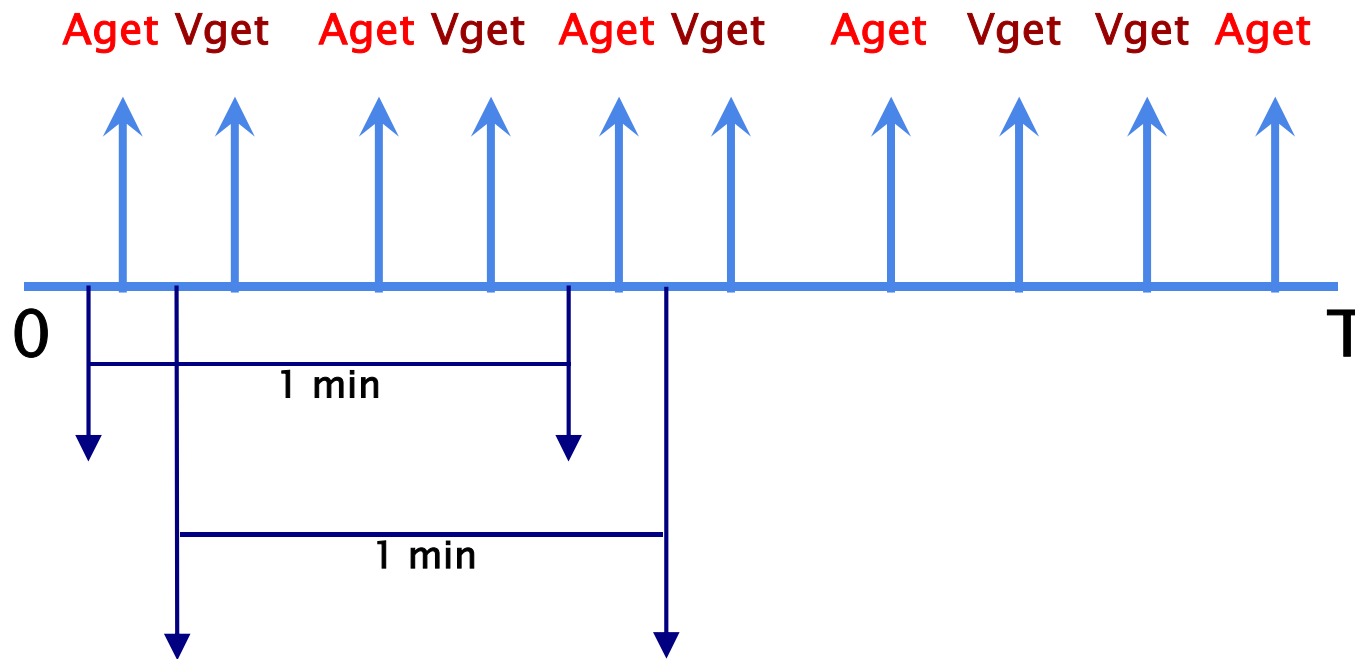
Quantitative verification for pacemakers

- Model the pacemaker and the heart as timed I/O automata
- Compose and **verify**



- Can we **synthesise** (controllable) timing delays to minimise energy, without compromising **safety**?

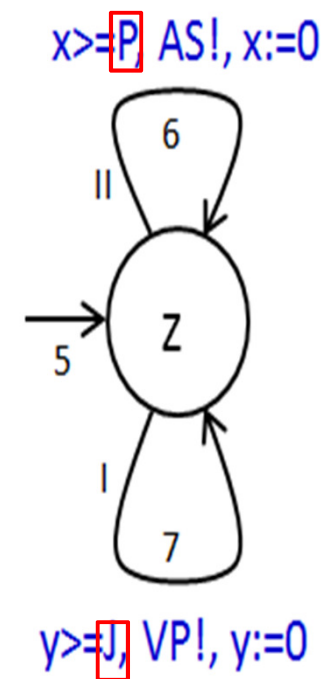
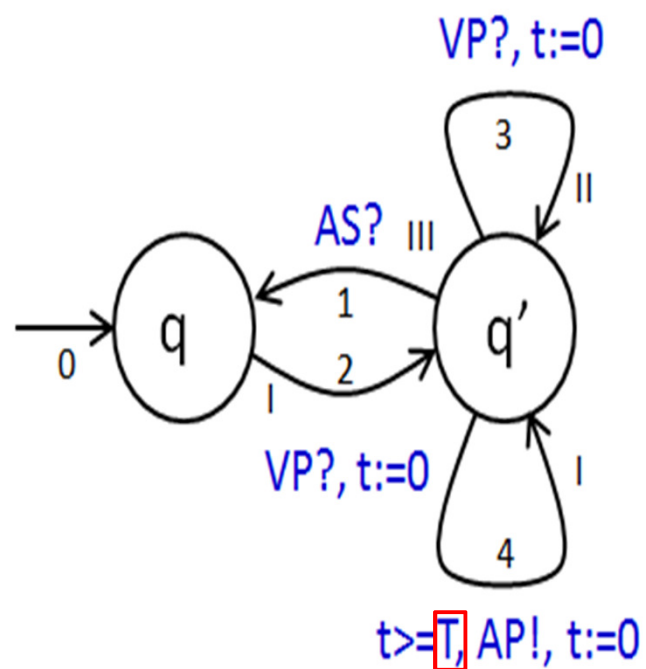
Property patterns: Counting MTL



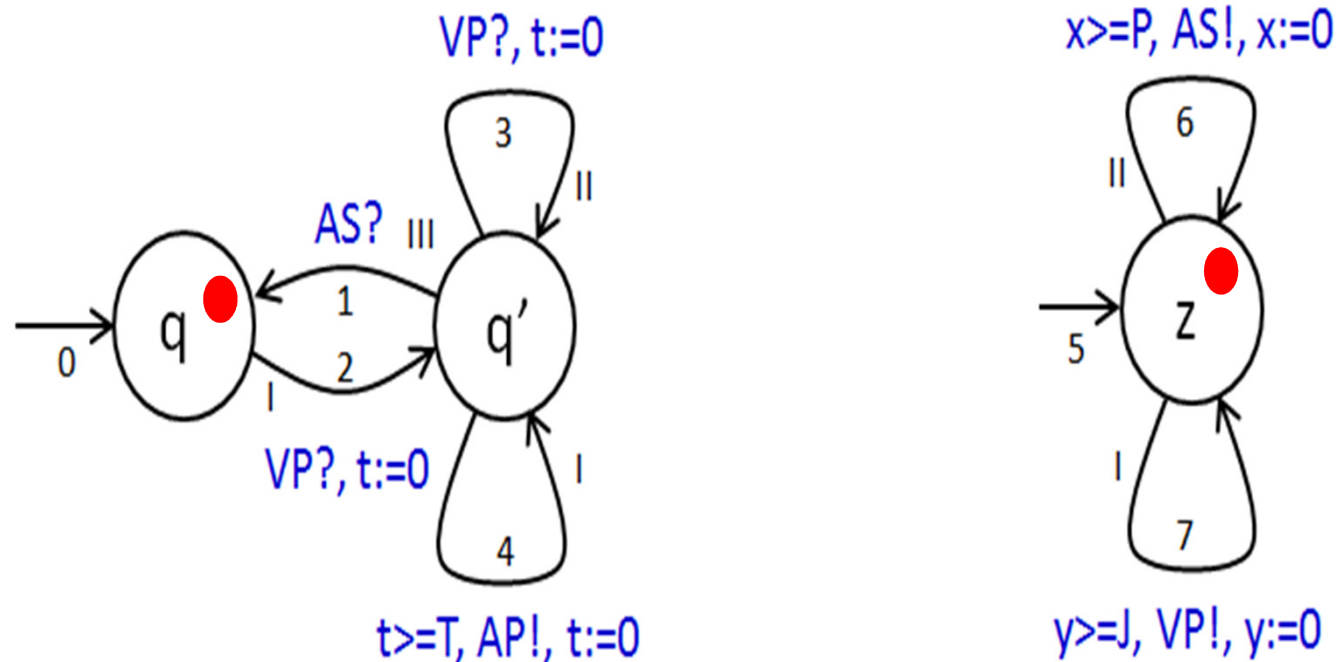
$$\Box^{[0,\tau]}(\#_0^\tau Vget \geq B_1 \wedge \#_0^\tau Vget \leq B_2)$$

Safety “for any 1 minute window, heart rate is in the interval [60,100]”

Example: timed I/O automata

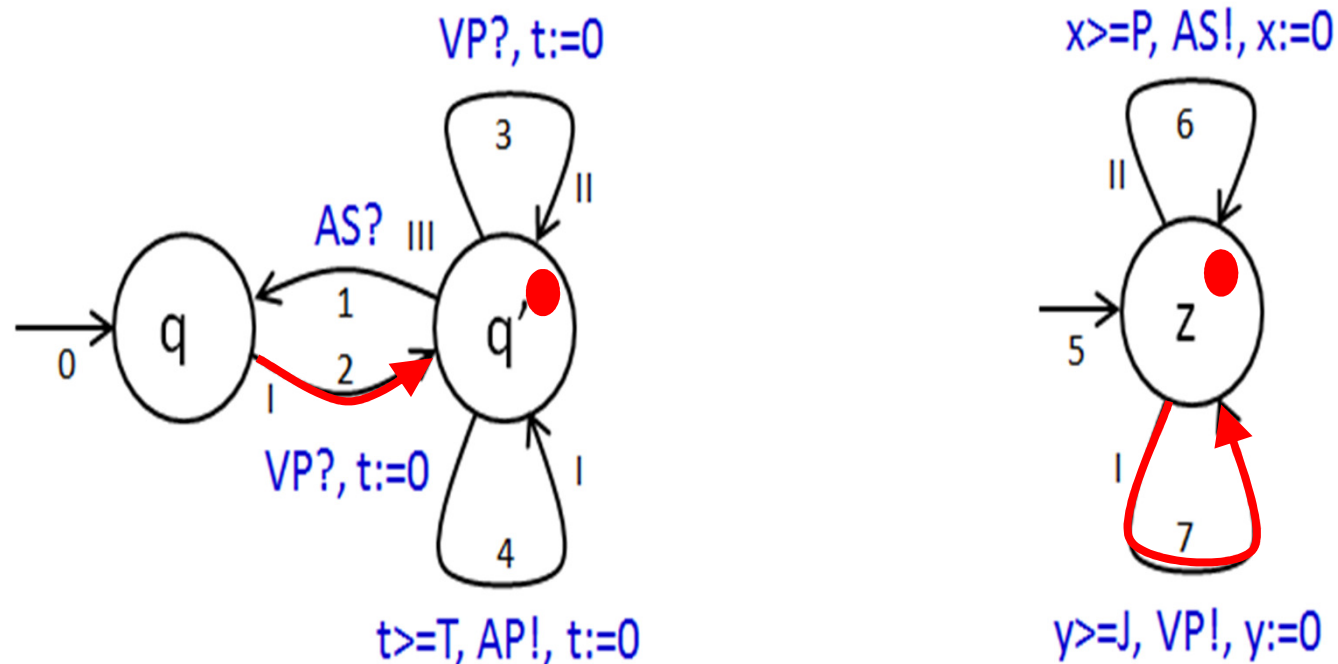


Example: timed I/O automata



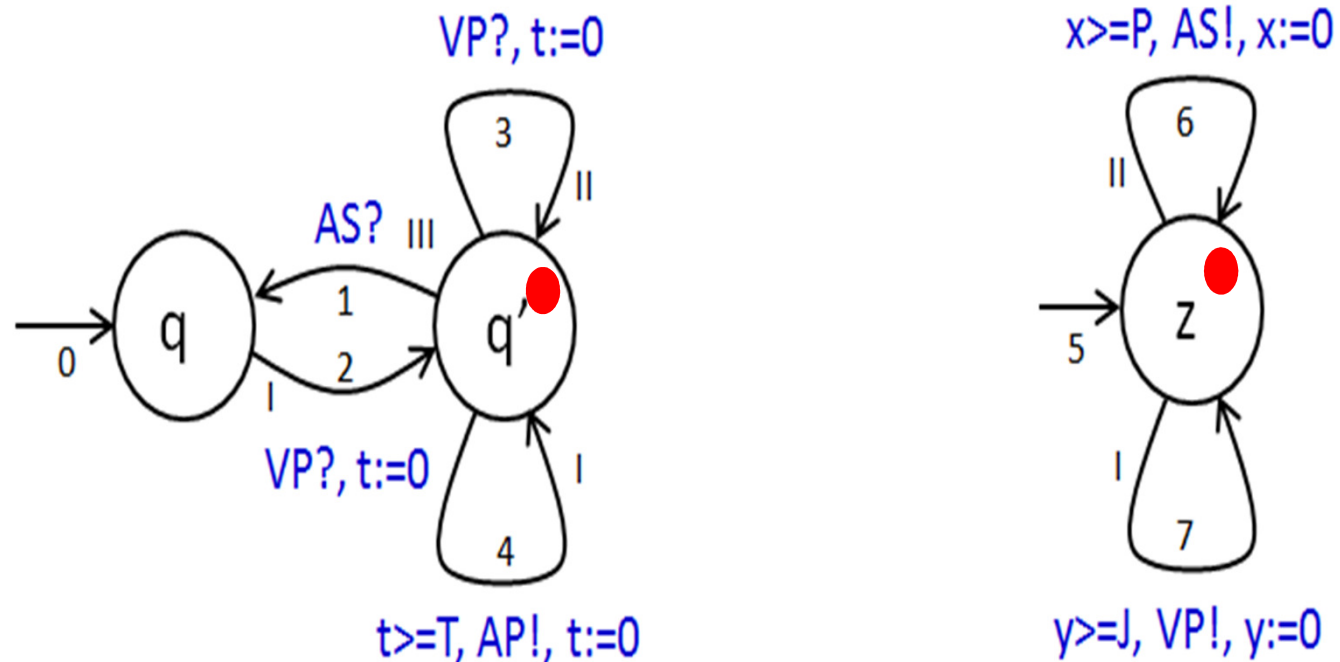
$$\rho = \boxed{(q, z)} \xrightarrow{t_0} (q', z) \xrightarrow{t_1} (q, z)$$

Example: timed I/O automata



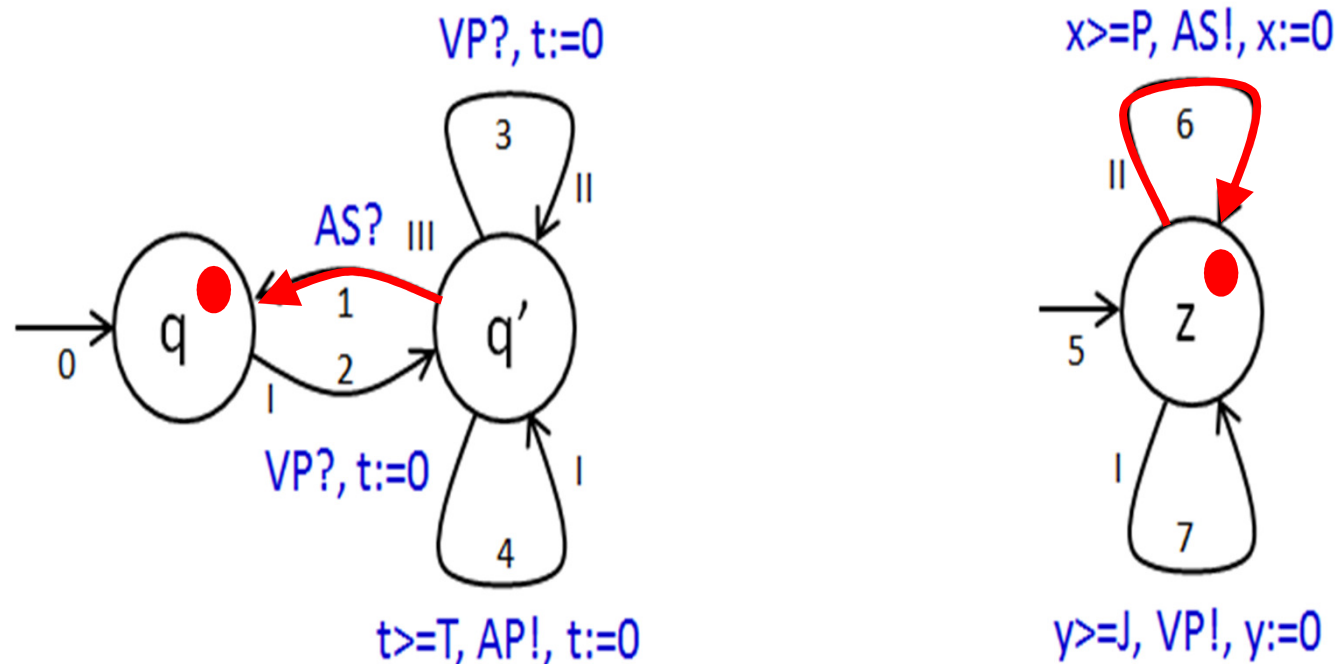
$$\rho = (q, z) \xrightarrow{t_0} (q', z) \xrightarrow{t_1} (q, z)$$

Example: timed I/O automata



$$\rho = (q, z) \xrightarrow{t_0} (q', z) \xrightarrow{t_1} (q, z)$$

Example: timed I/O automata

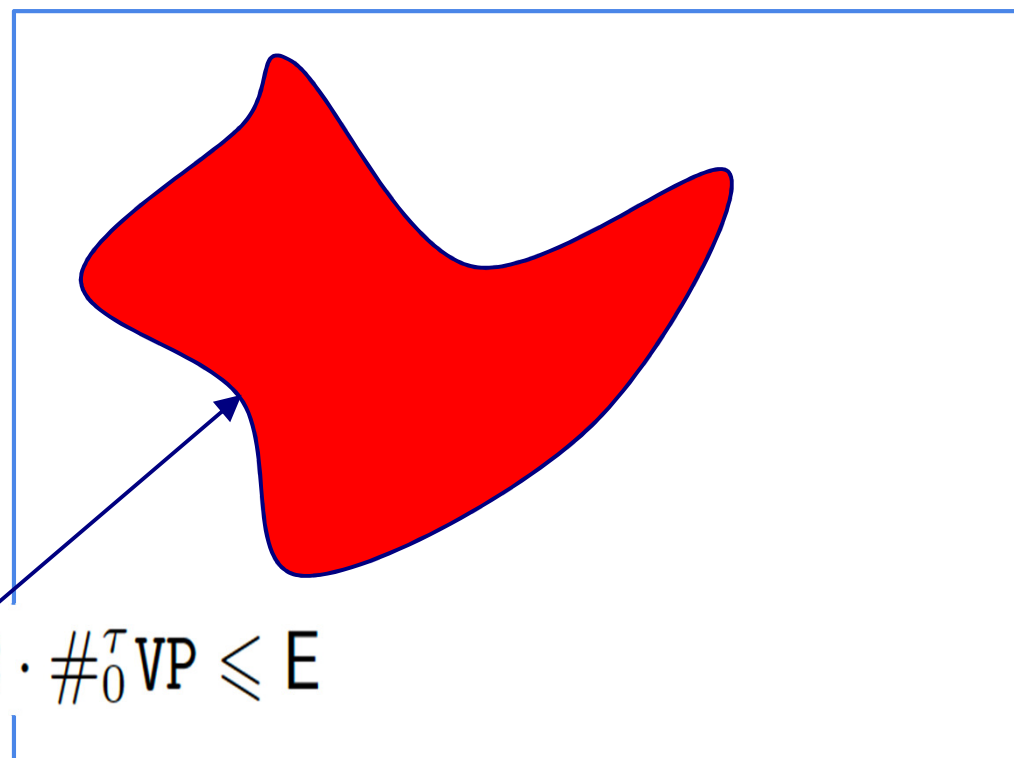


$$\rho = (q, z) \xrightarrow{t_0} (q', z) \xrightarrow{t_1} \boxed{(q, z)}$$

Optimal timing delays problem

- The **parameter synthesis problem** solved is
 - given a parametric network of timed I/O automata, set of controllable and uncontrollable parameters, CMTL property ϕ and length of path n
 - find the **optimal controllable** parameter values, for any uncontrollable parameter values, with respect to an **objective function** O , such that the property ϕ is satisfied on paths of length n , if such values exist
- Consider family of objective functions
 - maximise volume, minimise energy
- Discretise parameters, assume bounded integer parameter space and path length
 - decidable but high complexity (high time constants)

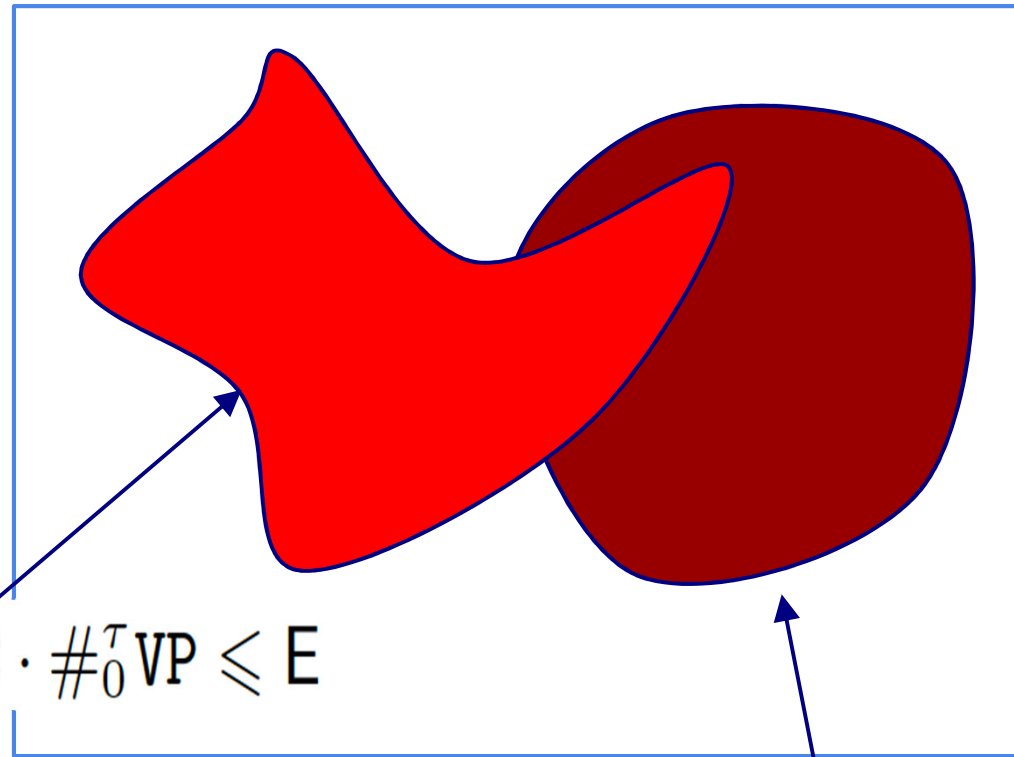
Parameter synthesis



$$1 \cdot \#_0^T \text{AP} + 2 \cdot \#_0^T \text{VP} \leq E$$

energy

Parameter synthesis



$$1 \cdot \#_0^\tau \text{AP} + 2 \cdot \#_0^\tau \text{VP} \leq E$$

energy

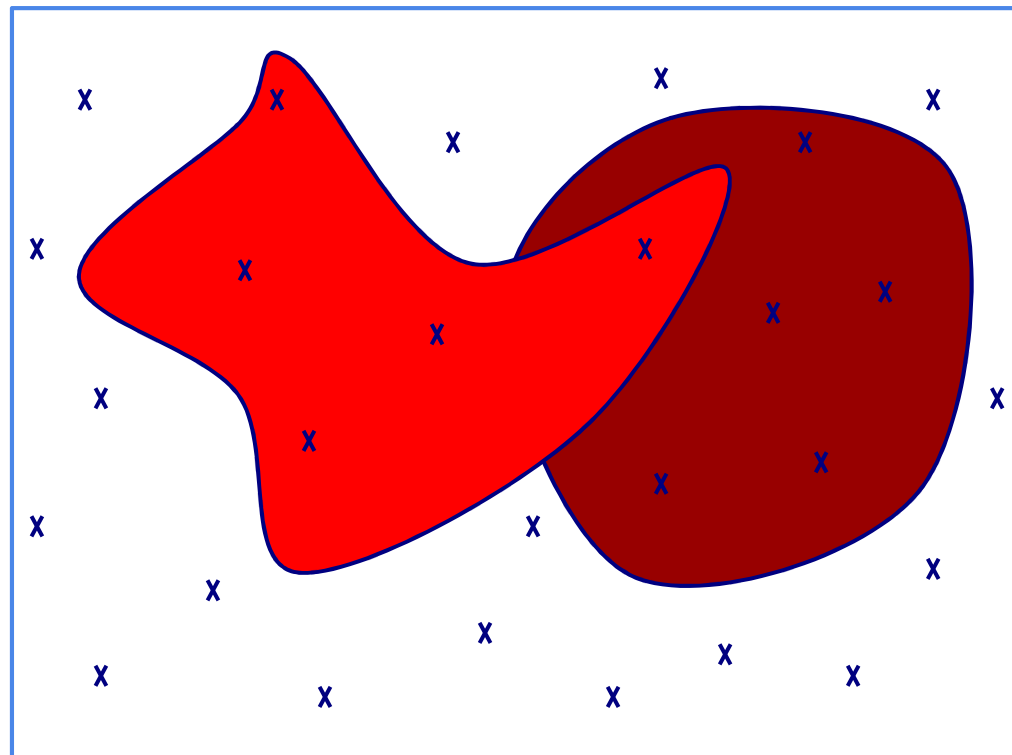
$$\Box^{[0,\tau]} (\#_0^\tau \text{Vget} \geq B_1 \wedge \#_0^\tau \text{Vget} \leq B_2);$$

safety

Our approach

- Constraints generation: **all** valuations that satisfy property
- Parameter optimisation: select **best** parameter values
- Sample the domain of the model parameter in order to generate a discrete path

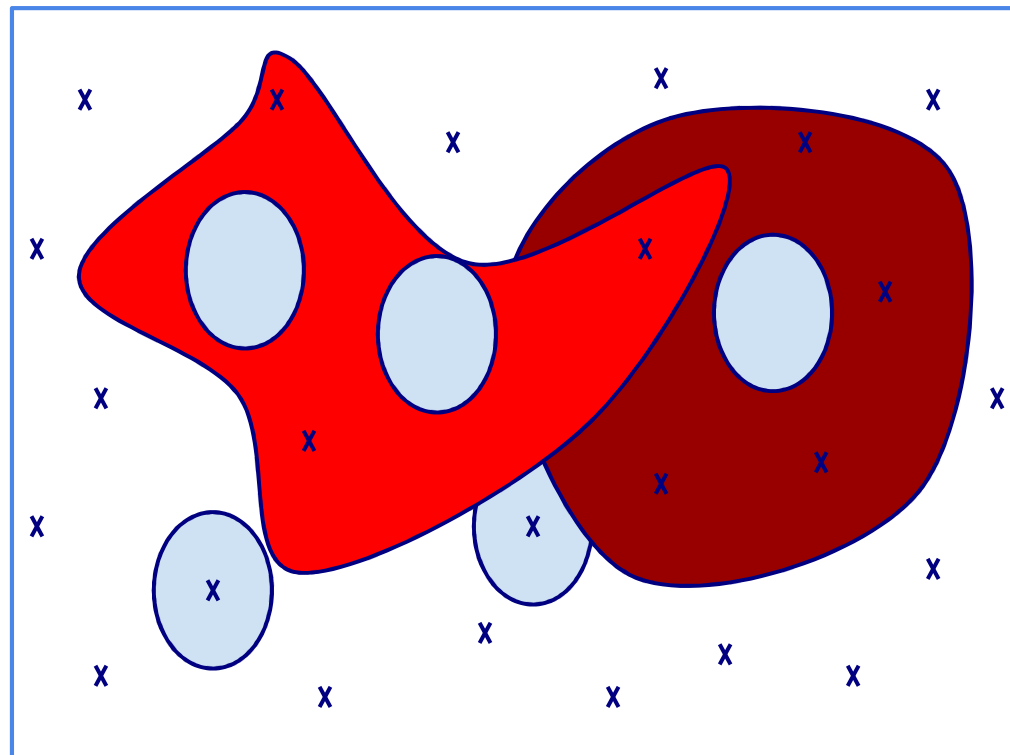
Parameter sampling



Our approach

- Constraints generation: **all** valuations that satisfy property
- Parameter optimisation: select **best** parameter values
- Sample the domain of the model parameter in order to generate a **discrete** path
- For each sampled parameter:
 - generate the **untimed** path
 - generate all **inequalities** which satisfy the CMTL property
- Advantage: **more** behaviours can be covered
 - need high coverage, but also need to consider robustness

Constraints generation



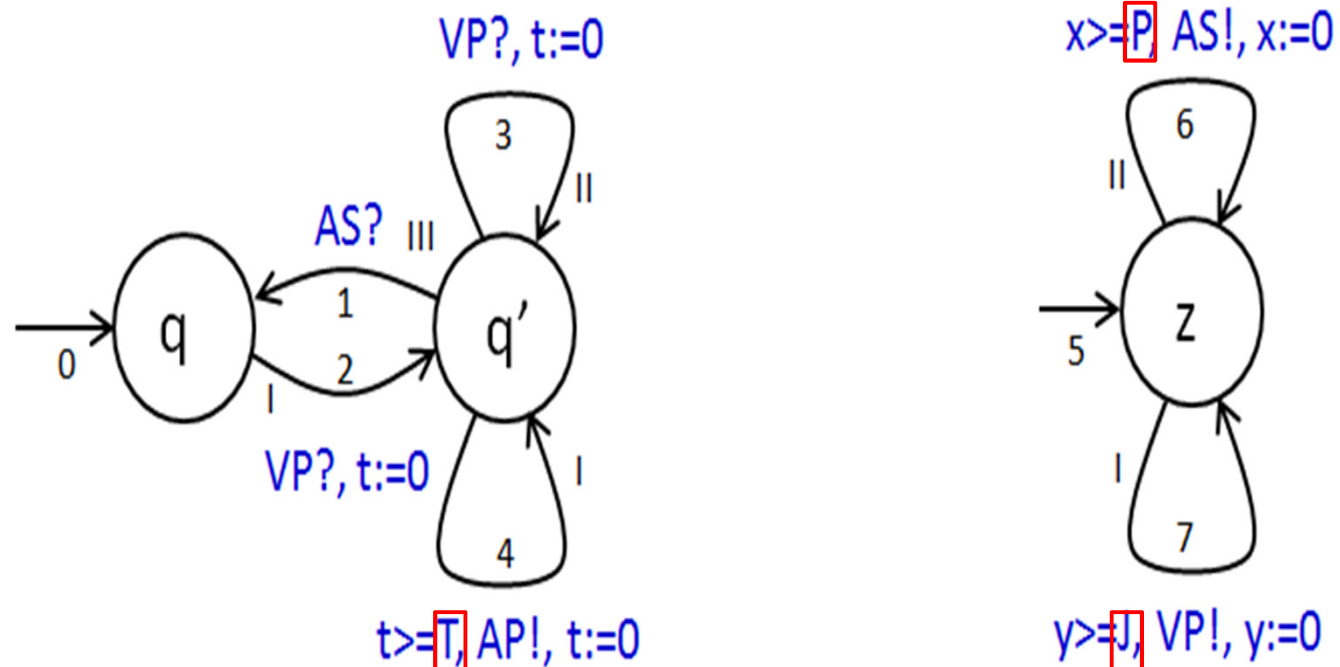
Parameter synthesis algorithm

Require: Network $\mathcal{N}$, formula φ and path length n

Ensure: Formula $\mathcal{S}$

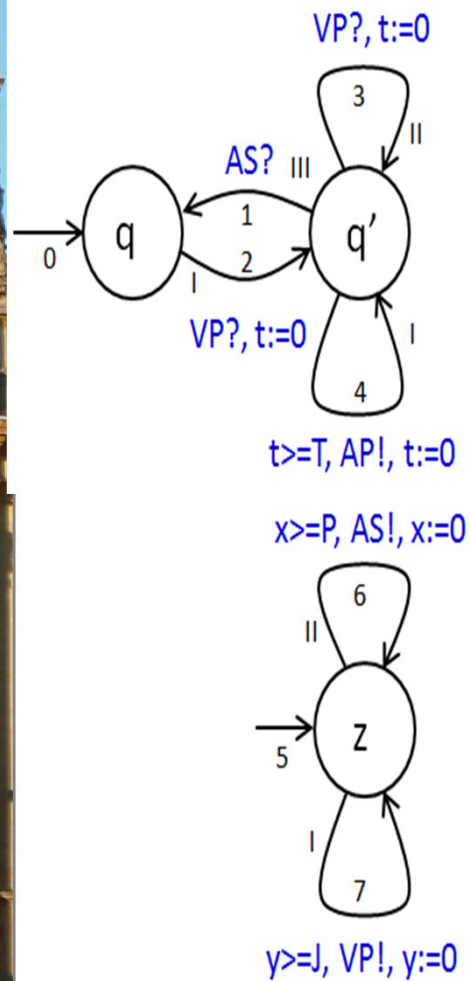
```
1: Function Sat( $\mathcal{N}, \varphi, n$ )
2:  $\bar{\Gamma} := \text{Sample}(\Gamma)$ 
3: for  $\vartheta \in \bar{\Gamma}$  do
4:   if  $\vartheta \notin \mathcal{S}$  then
5:      $\rho := \text{Gen\_path}(\mathcal{N}, n, \vartheta)$ 
6:      $(\mathcal{S}_\rho, \mathcal{T}) := \text{Path\_Constr\_Gen}(\mathcal{N}, \rho)$ 
7:      $\mathcal{S}_\varphi := \text{Constr\_Gen}(\rho, 0, \varphi, \mathcal{T})$ 
8:      $\mathcal{S} := \mathcal{S} \vee (\mathcal{S}_\rho \wedge \mathcal{S}_\varphi)$ 
9:   end if
10: end for
11: return  $\mathcal{S}$ 
```

Back to example...



$$\rho = (q, z)[2, 7](q', z)[4, -](q', z)$$

CMTL property

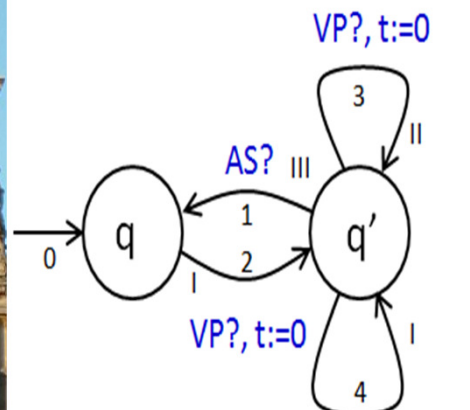


$$\varphi = \#_5^7 VP \geq 1$$

$(t_0 > 5)$	false
-------------	-------

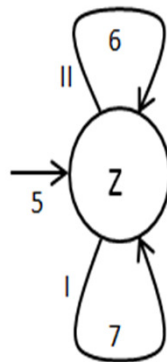
$$\rho = (q, z)[2, 7](q', z)[4, -](q', z)$$

CMTL property



$t \geq T, AP!, t := 0$

$x \geq P, AS!, x := 0$



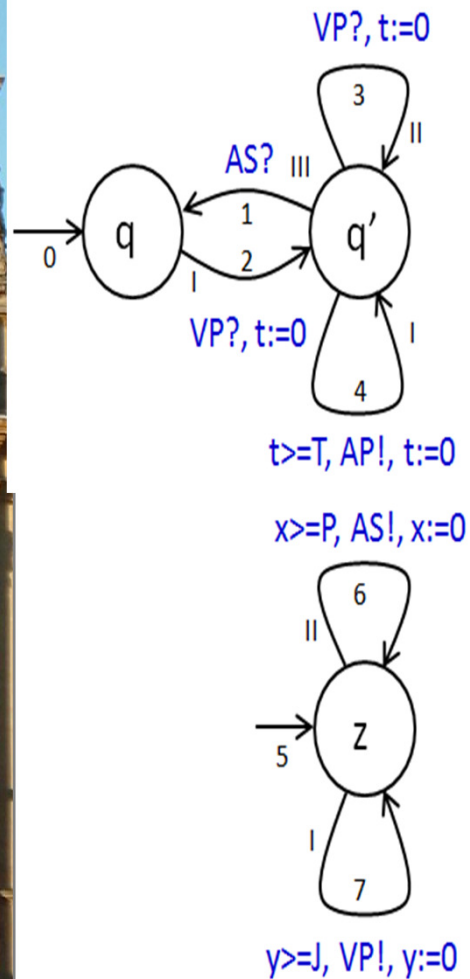
$y \geq J, VP!, y := 0$

$$\varphi = \#_5^7 VP \geq 1$$

$(t_0 > 5)$	false
$(t_0 > 5) \wedge$ $(t_0 + t_1 > 7 \wedge t_0 < 7)$	true

$$\rho = (q, z)[2, 7](q', z)[4, -](q', z)$$

CMTL property

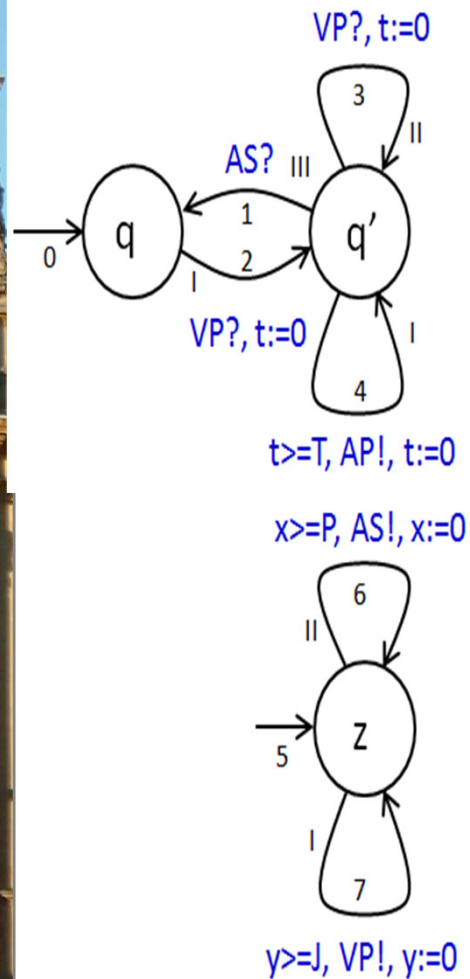


$$\varphi = \#_5^7 VP \geq 1$$

$(t_0 > 5)$	false
$(t_0 > 5) \wedge$ $(t_0 + t_1 > 7 \wedge t_0 < 7)$	true
$(t_0 + t_1 > 5 \wedge t_0 < 5) \wedge$ $(t_0 + t_1 > 7 \wedge t_0 < 7)$	false

$$\rho = (q, z)[2, 7](q', z)[4, -](q', z)$$

CMTL property



$$\varphi = \#_5^7 VP \geq 1$$

$(t_0 > 5)$	false
$(t_0 > 5) \wedge$ $(t_0 + t_1 > 7 \wedge t_0 < 7)$	true
$(t_0 + t_1 > 5 \wedge t_0 < 5) \wedge$ $(t_0 + t_1 > 7 \wedge t_0 < 7)$	false

$$t_0 = \mathcal{T}[0, 7] \text{ and } t_1 = \mathcal{T}[1, 4]$$

$$\rho = (q, z)[2, 7](q', z)[4, -](q', z)$$

Parameter optimisation

- Have obtained constraints on parameters that satisfy formula

- Maximal

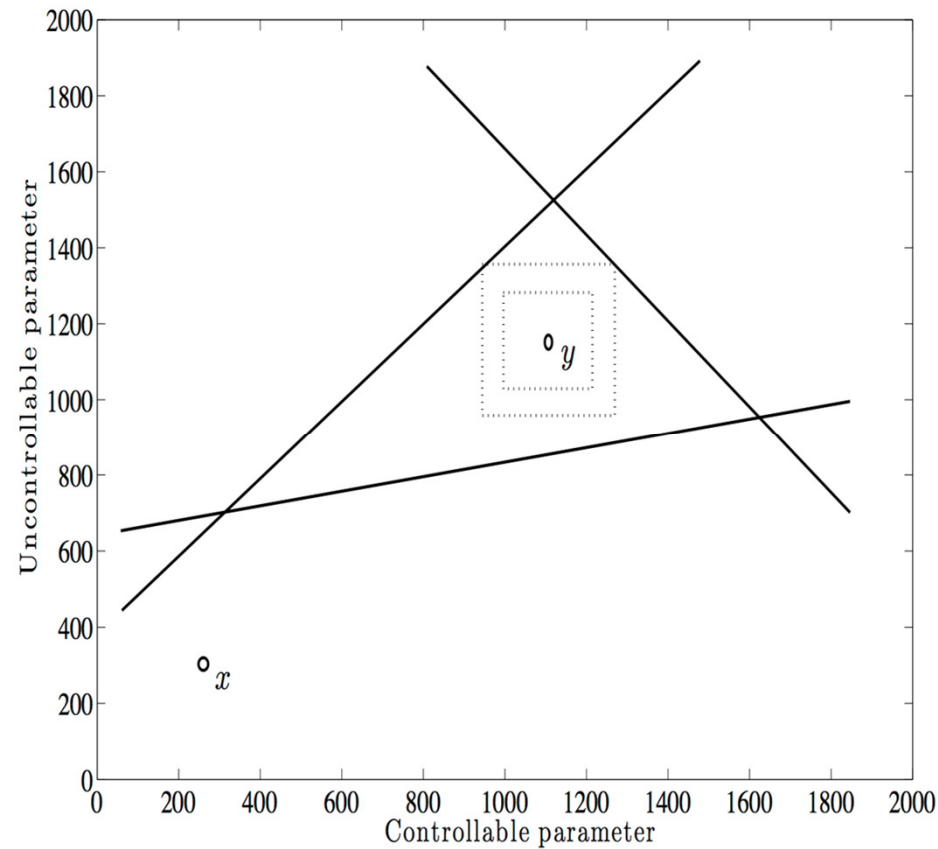
$$\text{opt}_v := \underset{\vartheta_c \in \mathcal{V}(\Gamma_c)}{\text{argsup}} \int_{\vartheta_u \in \mathcal{V}(\Gamma_u), (\vartheta_c, \vartheta_u) \in \mathcal{S}} \text{Distr}_{\Gamma_u}(d\vartheta_u)$$

- Robust objective function

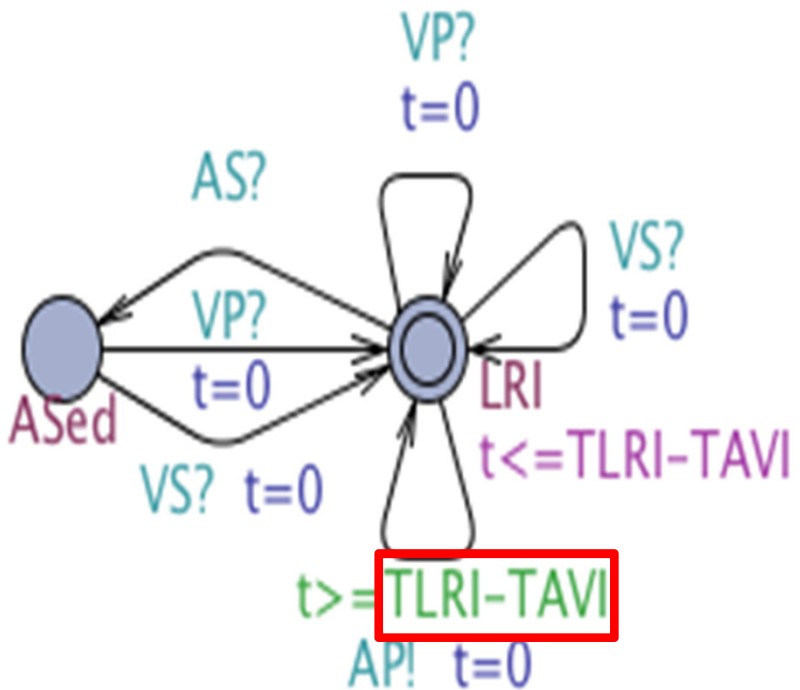
$$B_\epsilon(\vartheta) = \{\vartheta' \in \mathcal{V}(\Gamma) \mid \|\vartheta' - \vartheta\|_\infty \leq \epsilon\},$$

$$\text{opt}_r := \underset{\vartheta_c \in \mathcal{V}(\Gamma_c)}{\text{argsup}} \left\{ \sup_{\epsilon} \{ \epsilon \mid \vartheta_u \in \mathcal{V}(\Gamma_u), B_\epsilon((\vartheta_c, \vartheta_u)) \subseteq \mathcal{S} \} \right\}$$

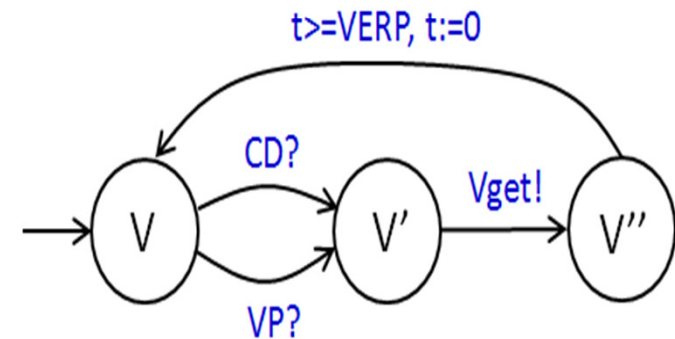
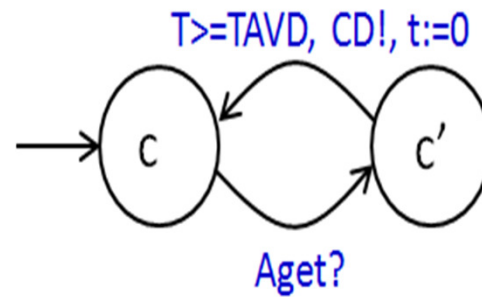
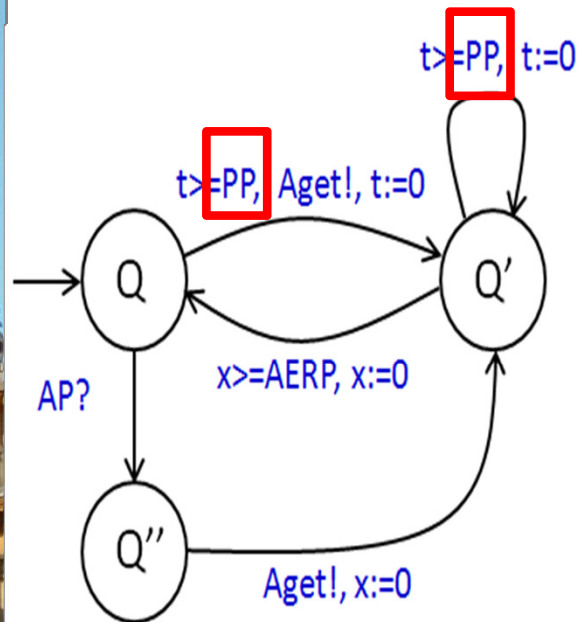
Robust objective function



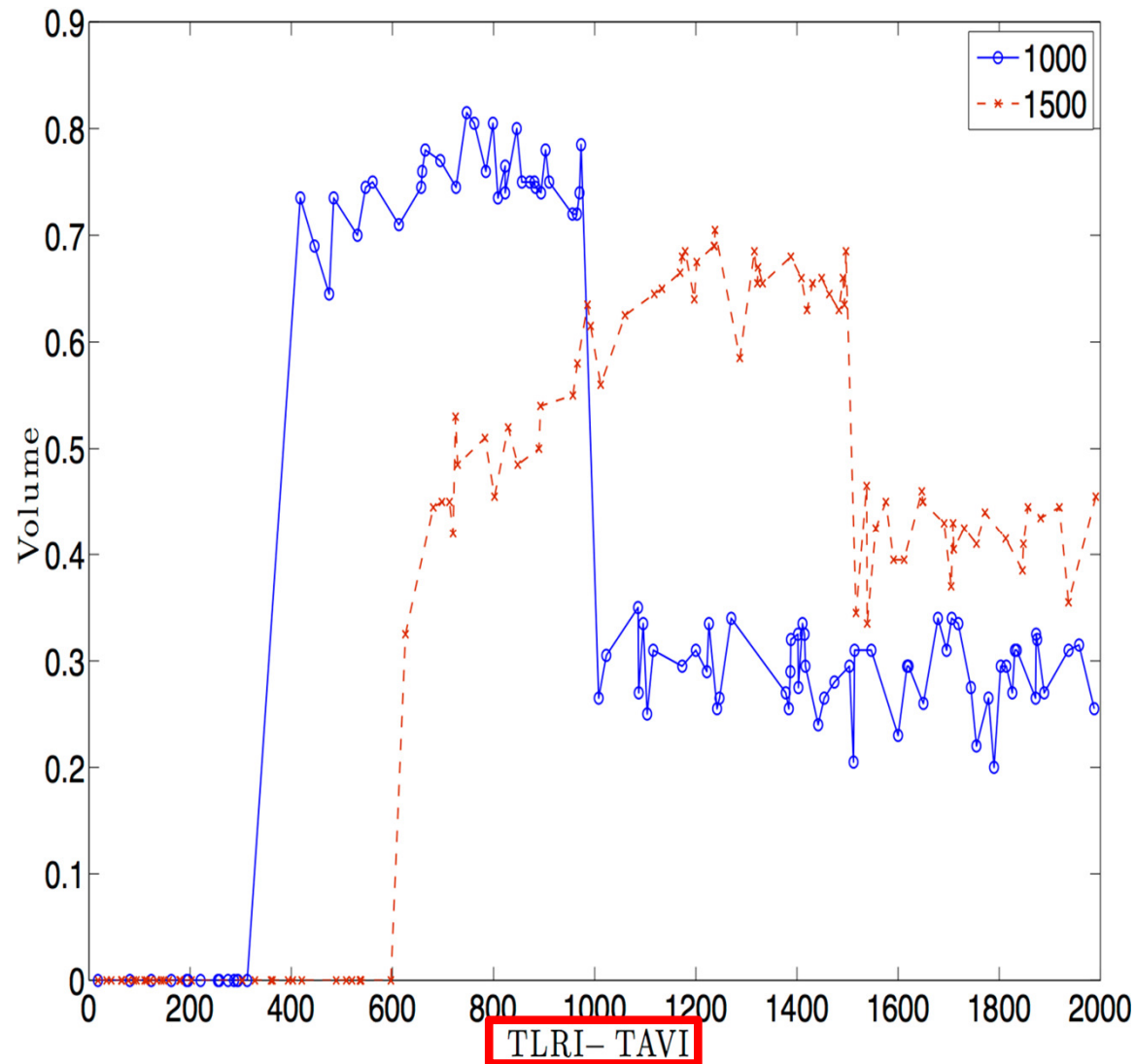
- For each sample point (controllable and uncontrollable)
 - generate path, safety and energy constraints
 - take disjunction, conjuncted with parameter bounds



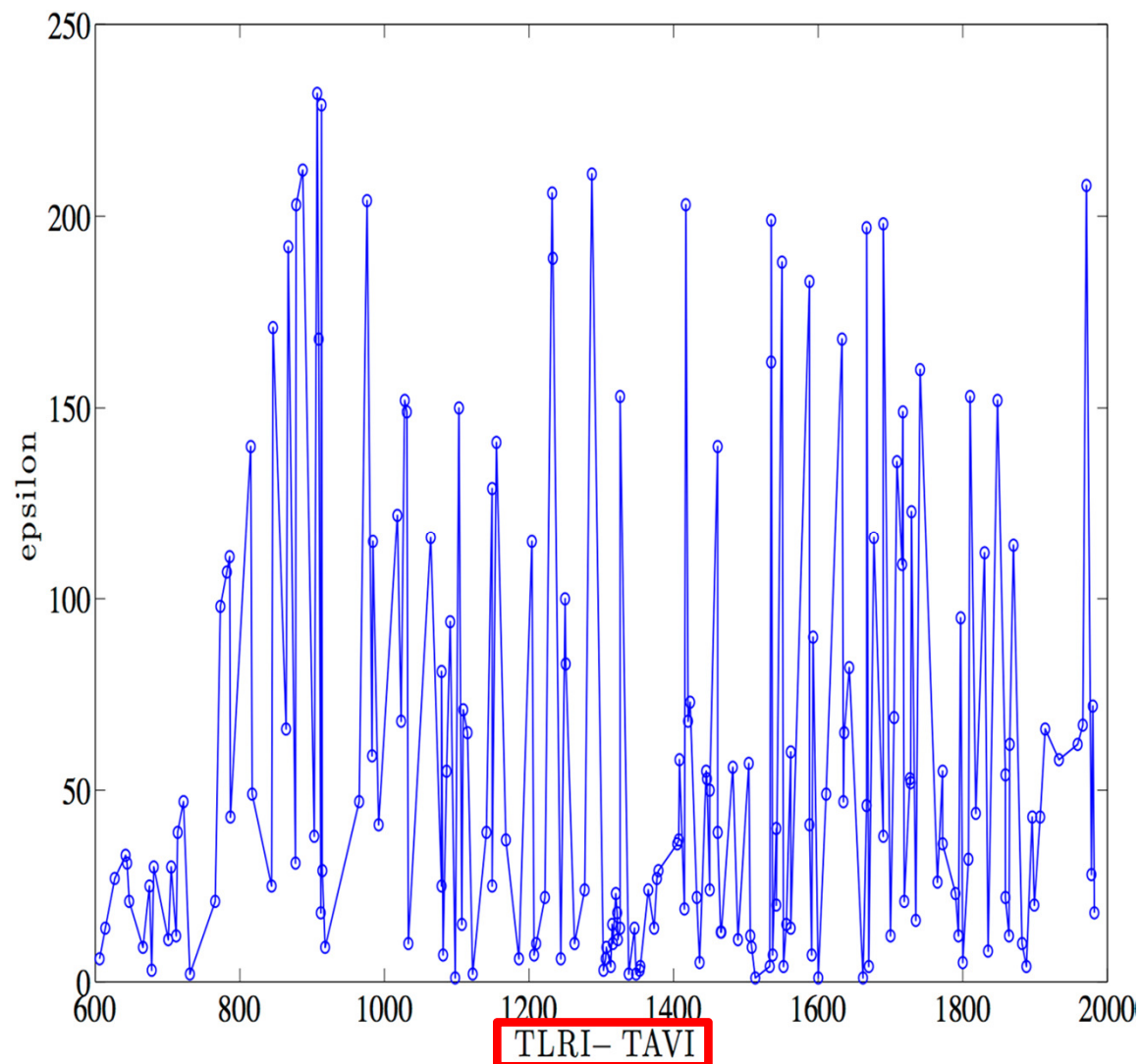
Human heart timed I/O automata model



Results: maximal volume objective (PP)



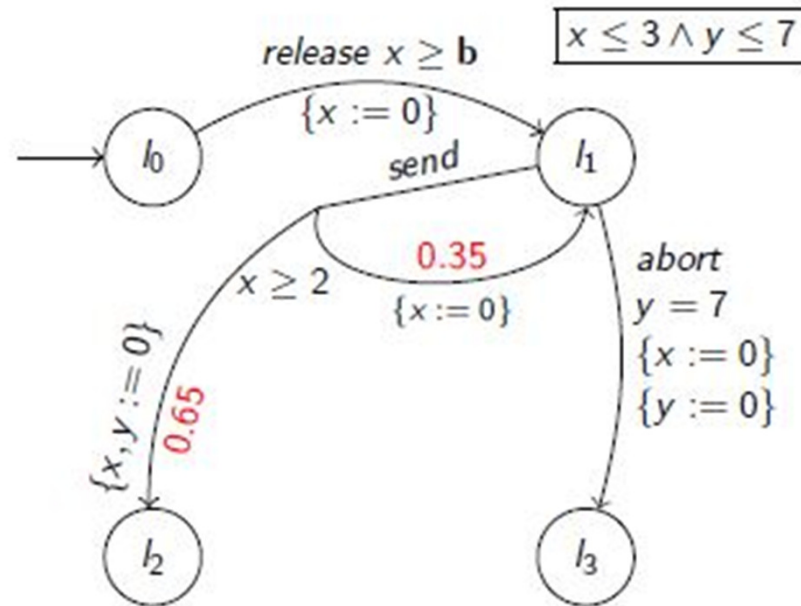
Results: robust objective (PP)



2. Optimal probability timing delays

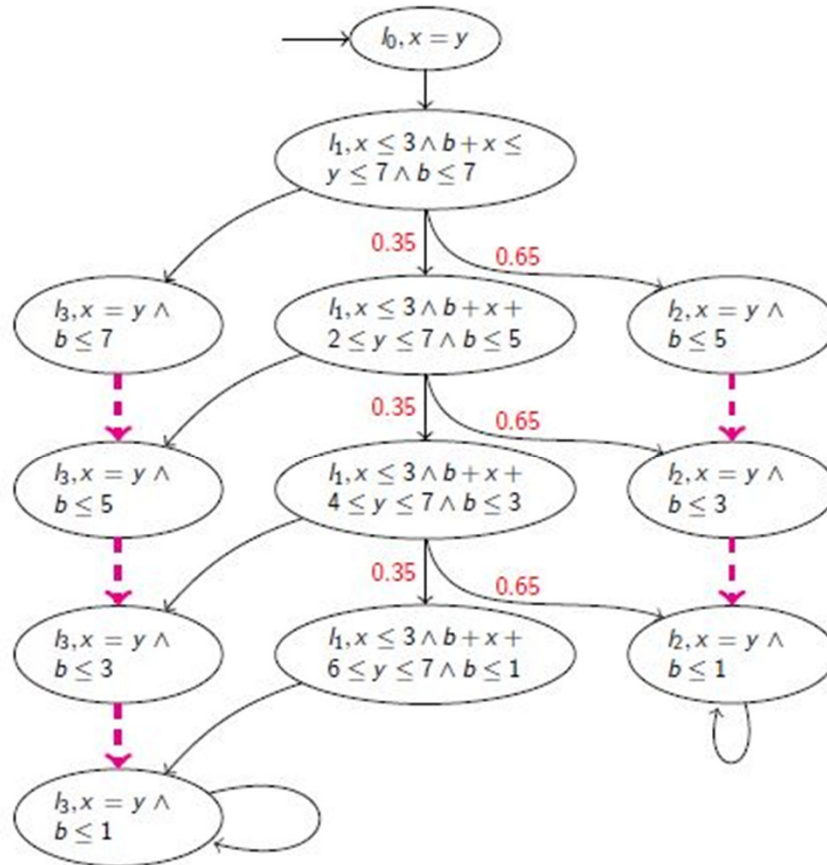
- Previously, **no** nondeterminism and **no** probability in the model considered
- Consider **parametric probabilistic timed automata** (PPTA),
 - e.g. guards of the form $x \leq b$,
- can we synthesise optimal timing parameters to **optimise** the reachability probability?
- Semi-algorithm
 - exploration of **parametric symbolic** states, i.e. location, time zone and parameter valuations
 - forward exploration only gives upper bounds on maximum probability (resp. lower for minimum)
 - but stochastic **game** abstraction yields the precise solution...
 - expected time challenging
- Implementation in progress

Example: parametric PTA



- Consider maximum probability of reaching l_2
 - $b = 0, 1$: 0.957125
 - $b = 2, 3$: 0.8775
 - $b = 4, 5$: 0.65
 - $b > 6$: 0

Example (MDP abstraction)



max probab of l_2

- $b = 0, 1$: 0.957125
- $b = 2, 3$: 0.8775
- $b = 4, 5$: 0.65
- $b > 6$: 0

3. Optimal rates

- Motivation: **systems and synthetic biology**
 - signalling pathways, gene regulation, epidemic models
 - DNA logic gates, DNA walker circuits
 - low molecular counts => stochastic dynamics
 - semantics given by continuous-time Markov chains (CTMCs)
- **Uncertain kinetic parameters**
 - limited knowledge of rate parameters
 - parameters affect behaviour and functionality of systems
 - NB safety-critical if used in biosensors...
- Can we find **rate values** so that a reliability property is satisfied?

Precise Parameter Synthesis for Stochastic Biochemical Systems. Ceska et al. In *Proc. 12th 46 Conference on Computational Methods in Systems Biology (CMSB'14)*, 2014.

Optimal rates

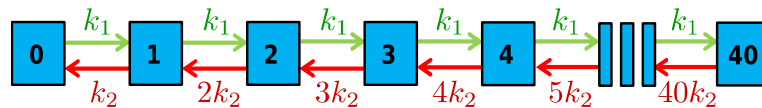
- **Models: continuous-time Markov chains**
 - real-time, exponentially distributed delays
 - extend with **rate parameters**, bounded parameter space
 - no nondeterminism (add priority and urgency of output)
- **CTMCs for biochemical reaction networks**
 - state = vector of **populations**
 - transition rates given by rate parameters using **rate functions**
 - low degree polynomial functions (mass action kinetics, etc.)
- **Properties: Continuous Stochastic Logic**
 - time-bounded fragment, branching-time logic
 - probability and reward operators
 - example path formula $\phi = F^{[1000;1000]} 15 \leq X \leq 20$
 - Two variants: find rates so that the probability/reward of ϕ meets **threshold** (say 40%), or is **optimised**

Problem formulation

- Parametric CTMC pCTMC
 - transition rates depend on a set of variables K
 - parametric rate matrix R^K (**polynomials** with variables in K)
 - describes set of all instantiations C
- Satisfaction function Λ
 - let ϕ be a CSL path formula
 - $\Lambda(p)$ yields probability of ϕ being satisfied in states s of C
 - analytical computation of Λ is **intractable**
 - can be discontinuous due to nested probabilistic operators

Example: satisfaction function

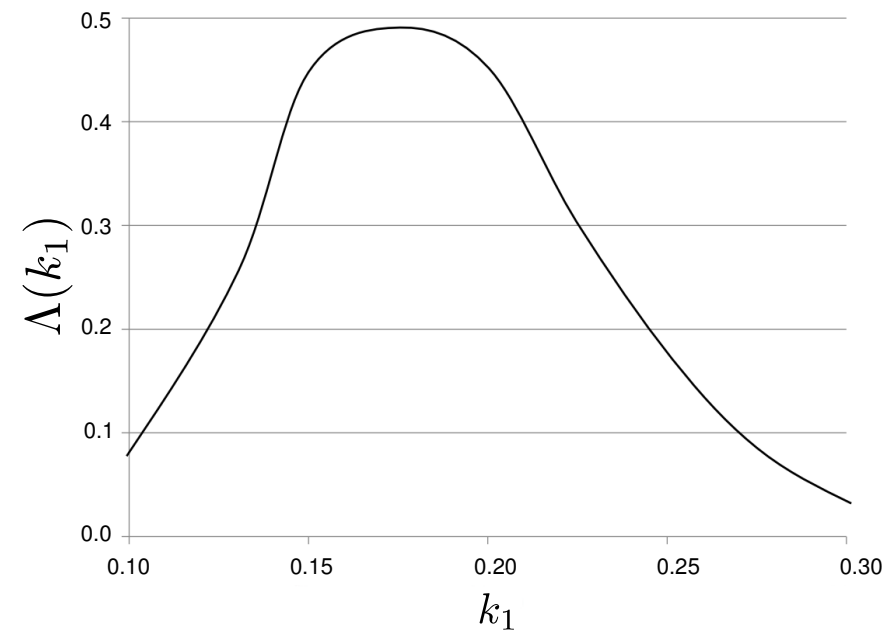
pCTMC + property



$$\phi = F^{[1000, 1000]}(X \geq 15 \wedge X \leq 20)$$

$$k_1 \in [0.1, 0.3] \quad k_2 = 0.01 \quad s_0 = [X]_0 = 15$$

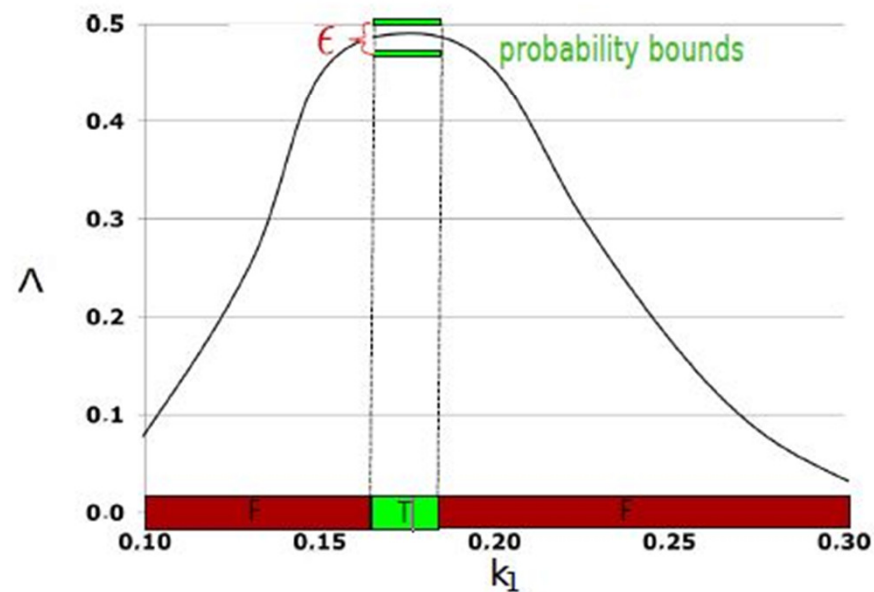
Satisfaction function



Max synthesis problem

For a given $\mathcal{P}$, ϕ and probability tolerance ϵ the problem is finding a partition $\{T, F\}$ of $\mathcal{P}$ and probability bounds $\Lambda^\perp$, $\Lambda^\top$ such that:

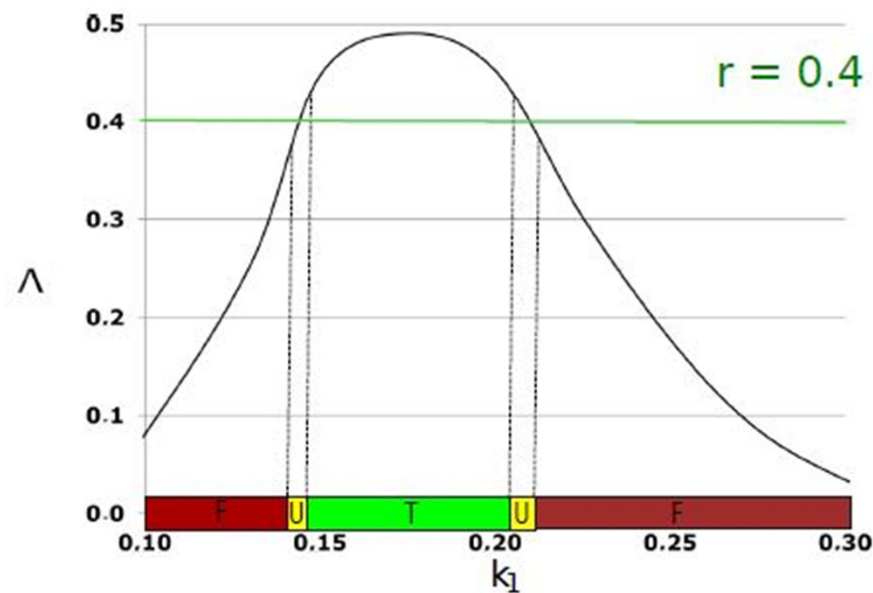
- ① $\Lambda^\perp - \Lambda^\top \leq \epsilon$;
- ② $\forall p \in T. \Lambda^\perp \leq \Lambda(p) \leq \Lambda^\top$; and
- ③ $\exists p \in T. \forall p' \in F. \Lambda(p) > \Lambda(p')$.



Threshold synthesis

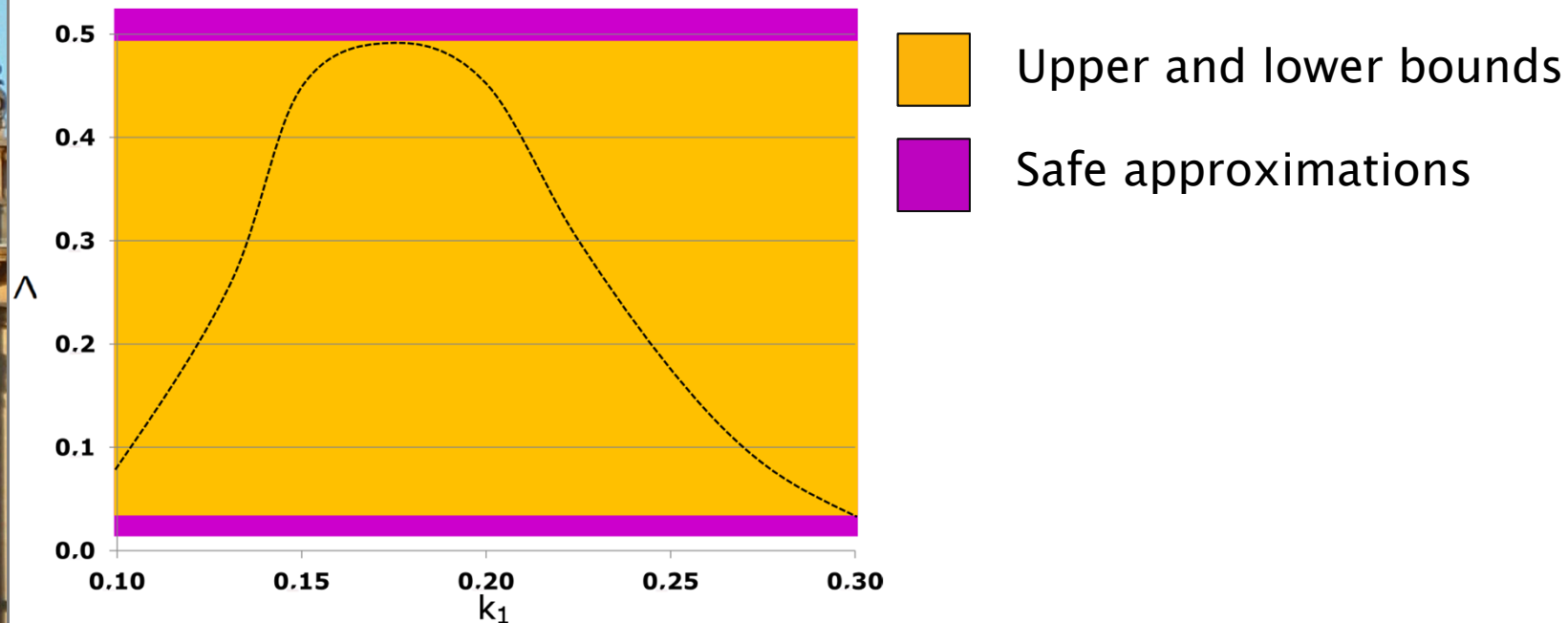
For a given $\mathcal{P}$, ϕ , probability threshold r and volume tolerance ε , the problem is finding a partition $\{T, U, F\}$ of $\mathcal{P}$ such that

- ① $\forall p \in T. \Lambda(p) \geq r$; and
- ② $\forall p \in F. \Lambda(p) < r$; and
- ③ $\text{vol}(U)/\text{vol}(\mathcal{P}) \leq \varepsilon$ ($\text{vol}(A)$ is the volume of A).



Solution approach

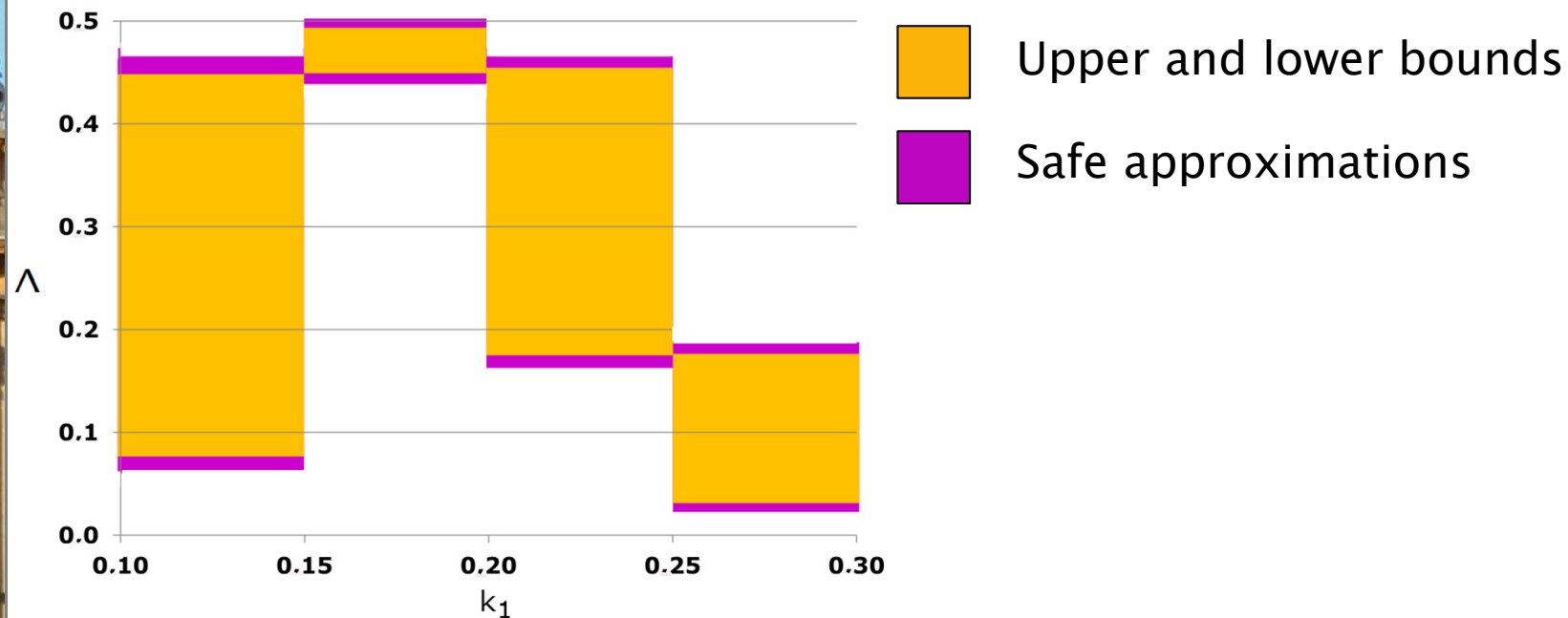
1. Method to compute **safe approximations** to min and max probabilities over a fixed parameter region



Iterative procedure, safe approximations computed for each subregion, same asymptotic complexity as transient analysis

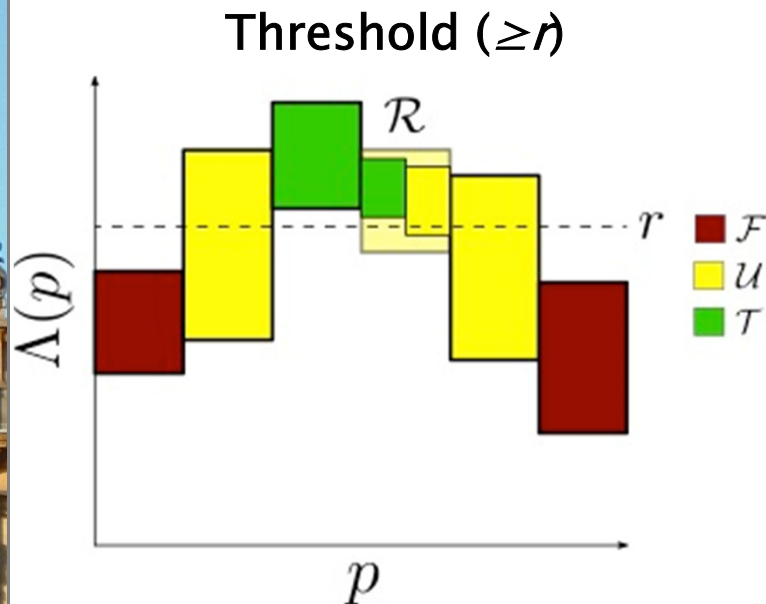
Solution approach

1. Method to compute **safe approximations** to min and max probabilities over a fixed parameter region
2. Parameter space decomposition, improves accuracy

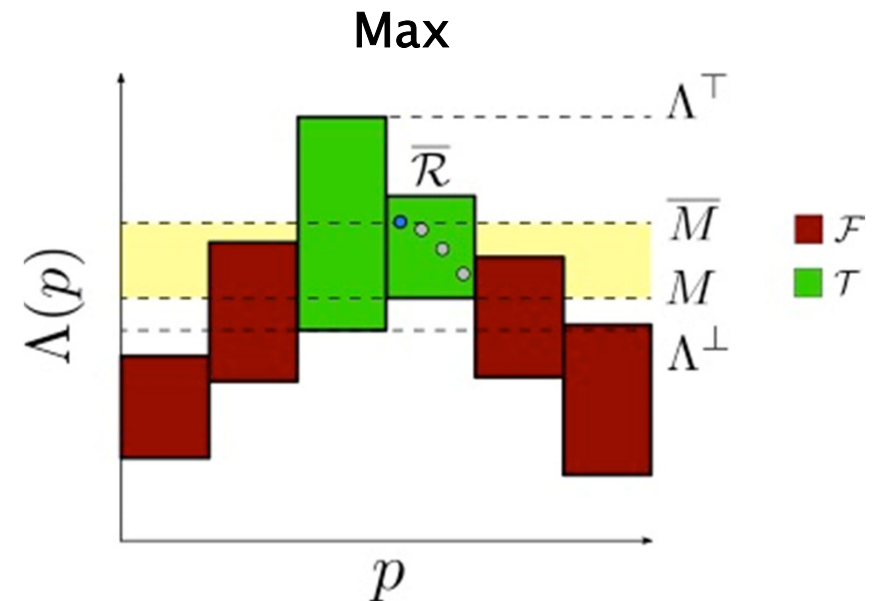


Λ is piecewise polynomial function, additional checks for jump discontinuities needed

Example: synthesis



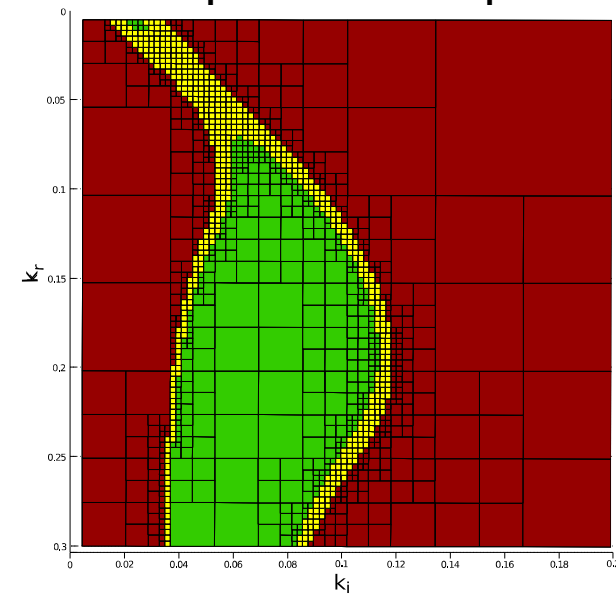
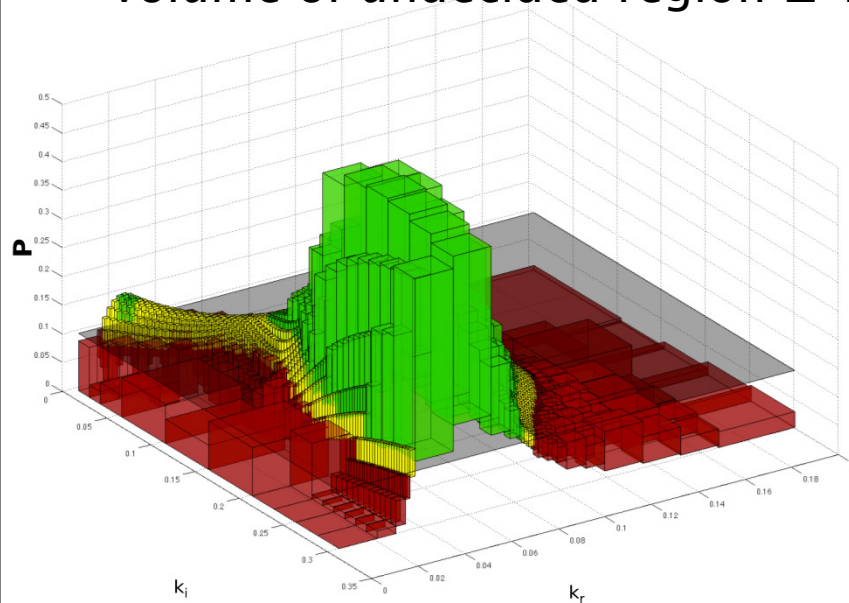
- True if lower bound above r
- False if upper bound below r
- Undecided otherwise (to refine)



- False if upper bound below under-approximation of max prob M
- True otherwise (to refine)

Epidemic model: threshold synthesis

- probability of property $\geq 10\%$
- volume of undecided region $\leq 10\%$ volume of the parameter space

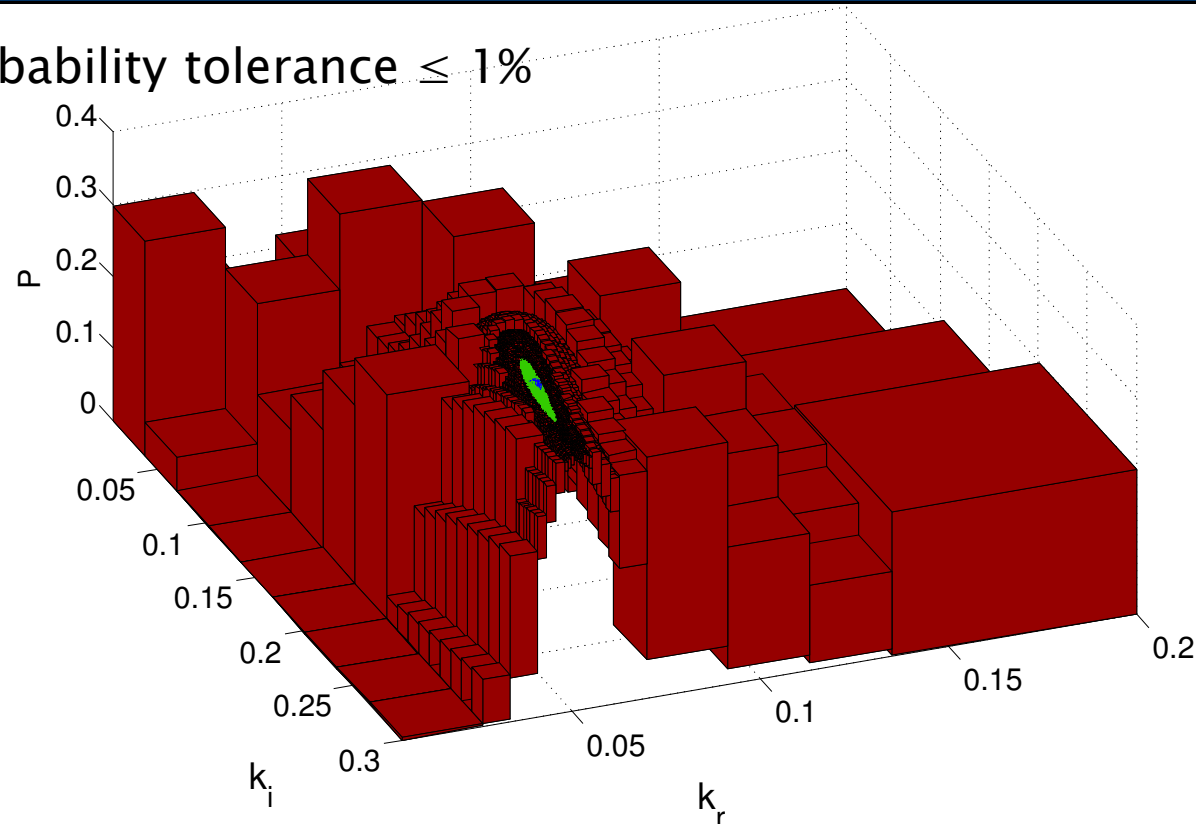


k_i k_r uncertain parameters

Property: $\phi = (I > 0)U^{[100,120]}(I = 0)$ (infection lasts at least 100 time units and ends within 120 time units)

Epidemic model: max synthesis

- probability tolerance $\leq 1\%$



k_i k_r uncertain parameters



Property: $\phi = (I > 0) U^{[100, 120]} (I = 0)$ (infection lasts at least 100 time units and ends within 120 time units)



Conclusions

- Formulated and proposed solutions to **parameter synthesis problems** for probabilistic real-time systems
 - **parametric** timing delays and rates
 - synthesise constraints or optimal parameters
 - variety of objectives
- **Techniques**
 - discretisation and integer parameters
 - constraint solving, including parametric symbolic constraints
 - iterative refinement to improve accuracy
 - sampling to improve efficiency
 - **but** scalability is still the biggest challenge
- **Implementation**
 - using tool combination involving Z3, python, PRISM

Other work and future directions

- Many challenges remain
 - timed automata models with data
 - hybrid automata models
 - effective model combinations of techniques
 - parallelisation and approximate methods
 - model synthesis from specifications
- More work not covered in this lecture
 - controller synthesis from multiobjective specifications
 - compositional controller synthesis
 - controller synthesis using machine learning
 - code generation
 - new application domains, ...
- and more...

Acknowledgements

- My collaborators in this work
- Project funding
 - ERC, EPSRC LSCITS
 - Oxford Martin School, Institute for the Future of Computing
- See also
 - PRISM www.prismmodelchecker.org
 - **VERIWARE** www.veriware.org