Advances and Challenges in Quantitative Verification and Synthesis for Cyber-Physical Systems

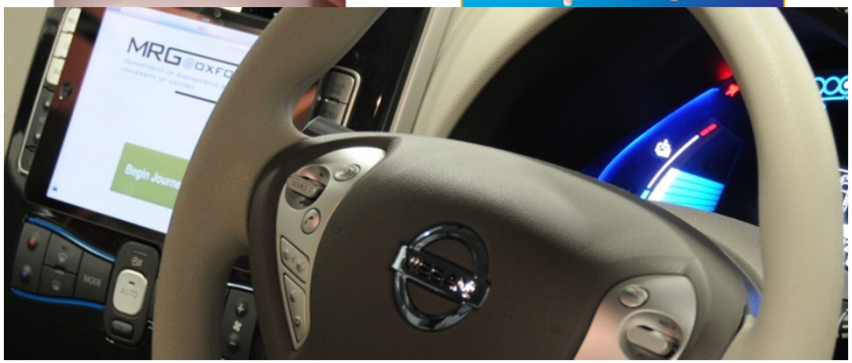
Marta Kwiatkowska

Department of Computer Science, University of Oxford

SoSCYPS, Vienna, 11th Apr 2016

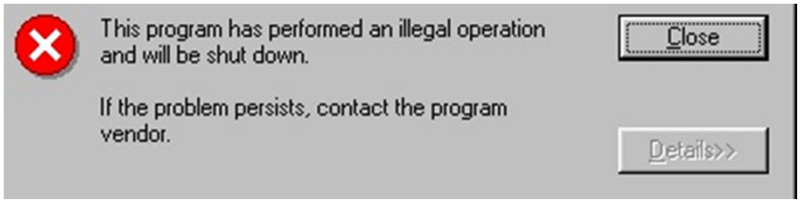
Cyber-Physical Systems

- Autonomous vehicles
  - self-parking, self-driving cars…
- Patient monitoring
  - closed-loop infusion pumps…
- Robotic assistants
- Multitude of devices
  - networked
  - Internet-enabled
  - organised into communities
- Main characteristics
  - sensors and actuators are often integral
  - increasingly autonomous behaviour
  - embedded software controls physical processes
  - combining discrete, continuous and stochastic dynamics



Are we safe?

- Embedded software at the heart of the device
- What if…
  - glucose monitor fails to trigger an alarm
  - infusion pump delivers the wrong dosage of drugs
  - self-parking car software crashes during the manouvre…



- Imagined or real?
  - February 2016: Nissan disables Leaf car app after discovering software could be hacked
  - May 2010 and each year since: FDA recalls programmable infusion pumps, over 710 patient deaths in five years, some because the device's software malfunctioned

Software quality assurance

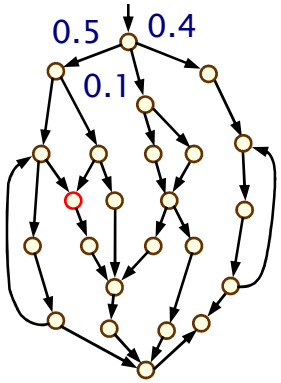
- Software is a critical component
  - embedded software failure costly and life endangering
- Need quality assurance methodologies
  - model-based development
  - rigorous software engineering
    - includes V&V
- Use formal techniques to produce guarantees for:
  - safety, reliability, performance, resource usage, trust, …
  - (safety) “probability of failure to raise alarm is tolerably low”
  - (reliability) “the smartphone will never execute the financial transaction twice”
- Focus on automated, tool-supported methodologies
  - automated verification via model checking
  - quantitative verification

Quantitative verification

Automatic verification (aka model checking) of quantitative properties of probabilistic system models

System

Probabilistic model
e.g. Markov chain



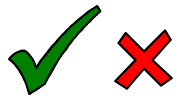
System requirements

$P_{<0.01}$ [$F^{\leq t}$ fail]

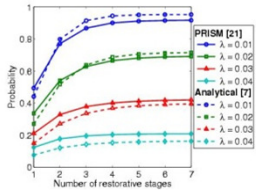
Probabilistic temporal logic specification
e.g. PCTL

Probabilistic model checker
e.g. PRISM

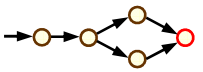
Result



Quantitative results

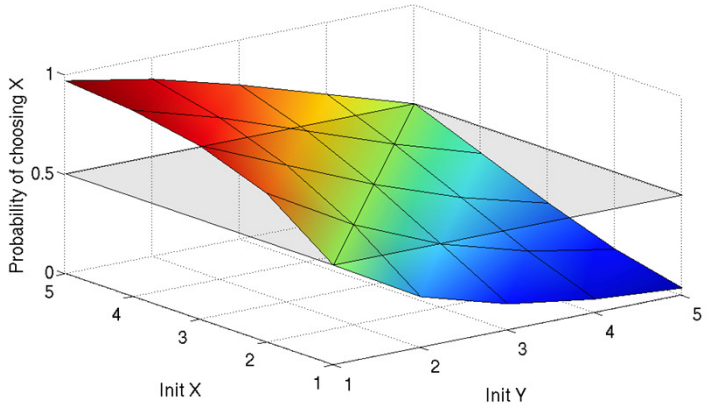


Counter-example



PRISM – Property specification

- Temporal logic-based property specification language
  - temporal order of events, in conjunction with probability and cost/rewards
- Simple examples:
  - $P_{\leq 0.01}$ [F "crash"] – "the probability of a crash is at most 0.01"
  - $S_{>0.999}$ ["up"] – "long-run probability of availability is >0.999"
- Usually focus on quantitative (numerical) properties:
  - $R_{\{\text{"energy"}\}\max=?}$ [$C^{\leq 7200}$] – "what is the worst-case expected energy consumption during the first 2 hours?"
  - then analyse trends in quantitative properties as system parameters vary



Quantitative probabilistic verification

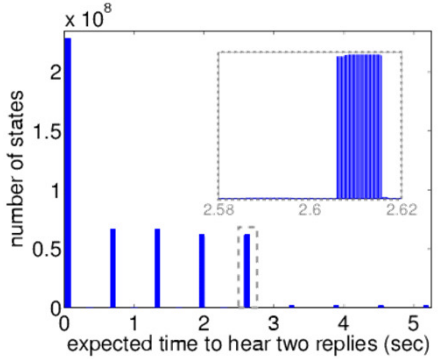
- What's involved
  - specifying, extracting and building of quantitative models
  - graph-based analysis: reachability + qualitative verification
  - numerical solution, e.g. linear equations/linear programming
  - typically computationally more expensive than the non-quantitative case
- The state of the art
  - fast/efficient techniques for a range of probabilistic models
  - feasible for models of up to 10^7 states (10^{10} with symbolic)
  - extension to probabilistic real-time systems
  - abstraction refinement (CEGAR) methods
  - compositional verification
  - statistical model checking
  - tool support exists and is widely used, e.g. PRISM, UPPAAL

PRISM

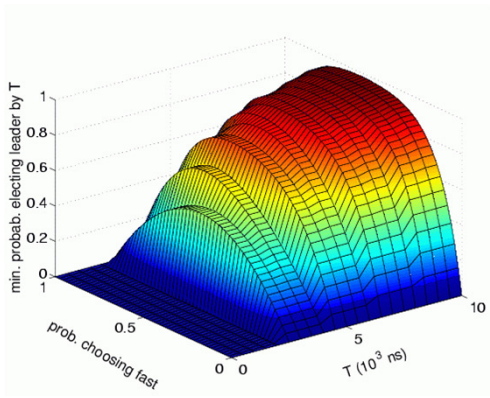
- PRISM: Probabilistic symbolic model checker
  - developed at Birmingham/Oxford University, since 1999
  - free, open source software (GPL), runs on all major Oss
  - simple but flexible high-level modelling language
- Various efficient model checking engines and techniques
  - symbolic methods (binary decision diagrams and extensions)
  - explicit-state methods (sparse matrices, etc.)
  - statistical model checking (simulation-based approximations)
- Graphical user interface
  - editors, simulator, experiments, graph plotting
- See: http://www.prismmodelchecker.org/
  - downloads, tutorials, case studies, papers,

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, 2.5 seconds



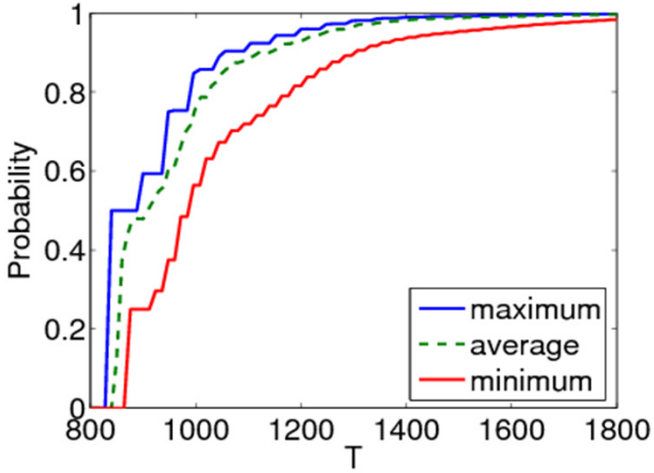
- FireWire root contention
  - wired protocol, uses randomisation
  - model checking using PRISM
  - optimum probability of leader election by time T for various coin biases
  - demonstrated that a biased coin can improve performance



Quantitative verification – Status

- Tools/techniques widely applicable, since real software/systems are quantitative
  - extensions/adaptations of model-based frameworks
  - new application domains

- Analysis "quantitative" & "exhaustive"
  - strength of mathematical proof
  - best/worst-case scenarios, not possible with simulation
  - identifying trends and anomalies



- But
  - the modelling phase time-consuming and error prone
  - often used as a debugging tool
  - scalability continues to be hard to overcome

From verification to synthesis

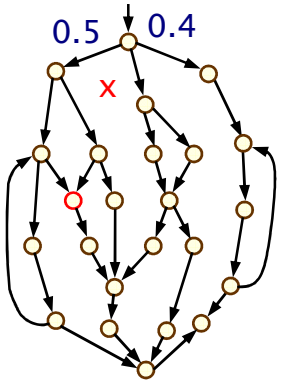
- Move towards quantitative model synthesis
  - i.e. "can we construct a model to guarantee that a given quantitative property is satisfied?"
  - instead of "does the model satisfy a given quantitative property?"
  - advantage: correct-by-construction
- We initially focus on simpler problems
  - strategy (controller) synthesis
  - parameter synthesis
  - etc
- Many application domains
  - robotics (controller synthesis from LTL/PCTL)
  - security (generating attacks or counter-strategies)
  - smart grid (optimal policy synthesis, coordination)

Quantitative verification and synthesis

Automatic synthesis of correct-by-construction strategies and models from quantitative properties/goals

System

Parametric model
e.g. Markov chain



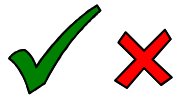
System require-ments

$P_{<0.01}$ [$F^{\leq t}$ fail]

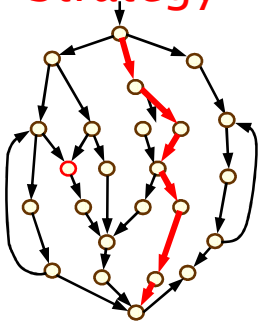
Probabilistic temporal logic specification goal
e.g. PCTL

Probabilistic model checker
e.g. PRISM

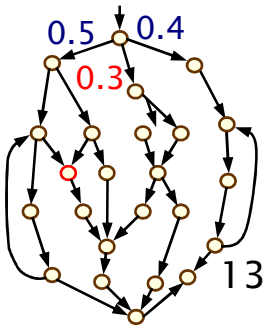
Result



Strategy



Parameters



The challenges of CPS

- Autonomous behaviour: electronic agents make decisions and act independently of humans, e.g. search and rescue
- Constrained resources: low power, processor speed and memory capacity, intermittent connectivity
- Adaptiveness: systems have to adapt to changing requirements in predictable fashion
- Monitoring and control of physical processes: needed in autonomous transport, robotic planning, implantable medical devices such as pacemakers, etc
- Communities of agents: need to model cooperation, competition and resource sharing, necessitating game-theoretic approaches
- Interfacing with the natural world: biosensing and DNA/molecular computation have important applications in disease detection and drug delivery

The challenges of CPS

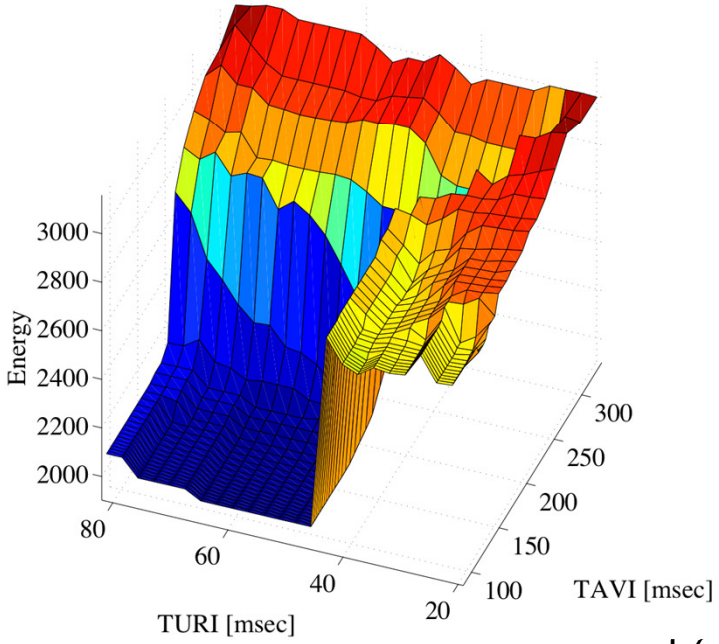
- Autonomous behaviour: electronic agents make decisions and act independently of humans, e.g. search and rescue
- Constrained resources: low power, processor speed and memory capacity, intermittent connectivity
- Adaptiveness: systems have to adapt to changing requirements in predictable fashion
- Monitoring and control of physical processes: needed in autonomous transport, robotic planning, implantable medical devices such as pacemakers, etc
- Communities of agents: need to model cooperation, competition and resource sharing, necessitating game-theoretic approaches
- Interfacing with the natural world: biosensing and DNA/molecular computation have important applications in disease detection and drug delivery

This lecture…

- Selected recent advances in quantitative verification
- Pacemaker case study: monitoring and control of physical processes
  - non-linear hybrid dynamics, stochasticity
  - (privacy and security essential)
  - ECG used in biometrics
  - optimal parameter synthesis
  - introducing HeartVerify
- Stochastic games to model cooperation, competition and resource sharing
  - strategy synthesis in adversarial situations
  - multi-objective specifications
  - introducing PRISM-games
- Beyond PRISM…

Case study: Cardiac pacemaker

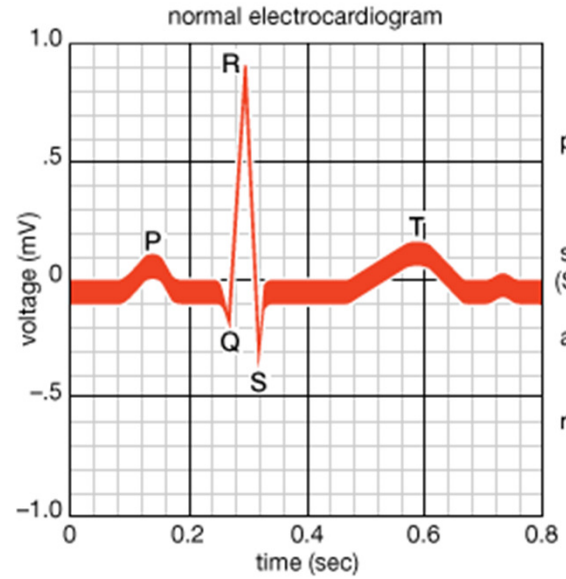
- Develop model-based framework
  - timed automata model for pacemaker software [Jiang et al]
  - hybrid heart models in Simulink, adopt synthetic ECG model (non-linear ODE) [Clifford et al]
- Properties
  - (basic safety) maintain 60-100 beats per minute
  - (advanced) detailed analysis energy usage, plotted against timing parameters of the pacemaker
  - parameter synthesis: find values for timing delays that optimise energy usage

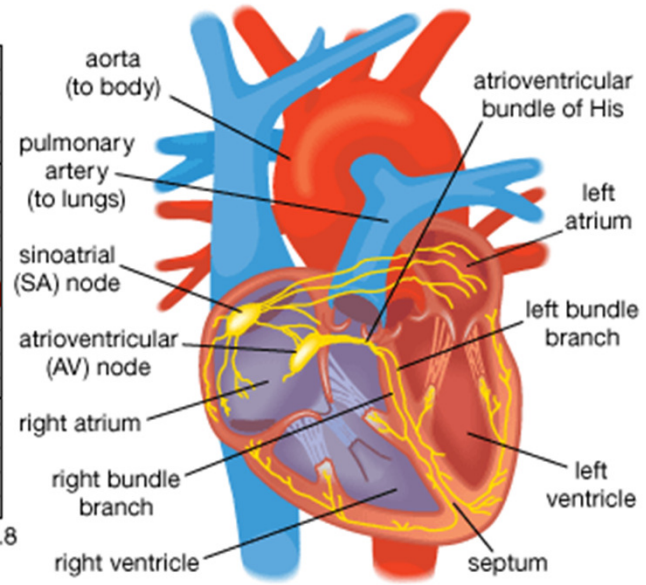


Modelling of the heart

- The heart maintains blood circulation by contracting the atria and ventricles
  - spontaneously generates electrical signal (action potential)
  - conducted through cellular pathways into atrium, causing contraction of atria then ventricles
  - repeats, maintaining 60–100 beats per minute

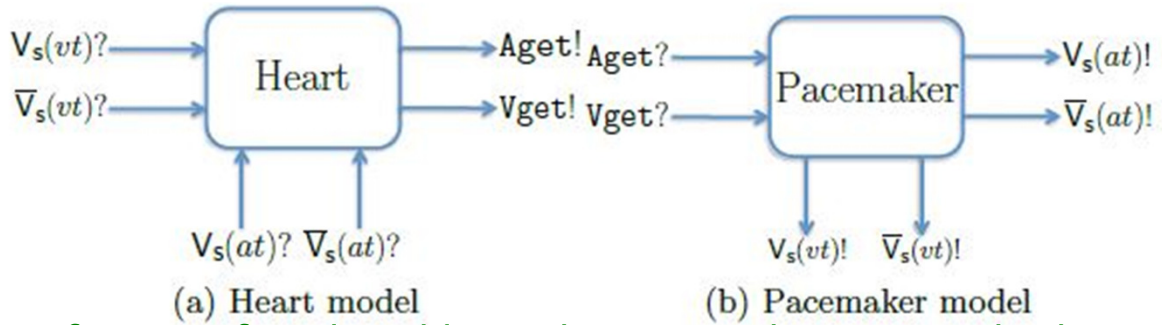
- Abnormalities in electrical conduction
  - missed/slow heart beat (Bradycardia)
  - treatable with pacemakers





Model-based framework

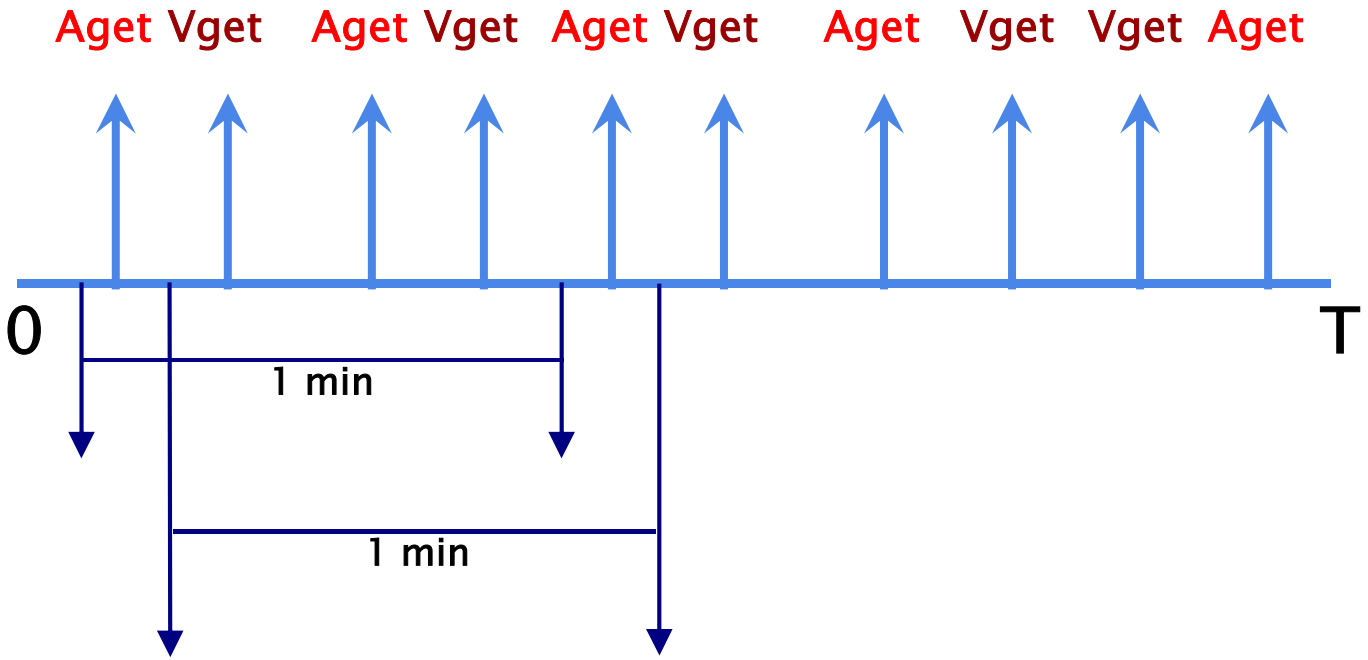
- We advocate a model-based framework
  - models are networks of communicating hybrid I/O automata, realised in Matlab Simulink
    - discrete mode switching and continuous flows: electrical conduction system
    - quantitative: energy usage and battery models
    - patient-specific parameterisation
  - framework supports plug-and-play composition of
    - **heart models** (timed/hybrid automata, some stochasticity)
    - **pacemaker models** (timed automata)



(a) Heart model (b) Pacemaker model

Quantitative Verification of Implantable Cardiac Pacemakers over Hybrid Heart Models. Chen *et al*, *Information and Computation*, 2014

Property specification: Counting MTL



$$\Box^{[0,\tau]}\left(\#_0^{\tau}\mathtt{Vget} \geqslant B_1 \wedge \#_0^{\tau}\mathtt{Vget} \leqslant B_2\right)$$

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

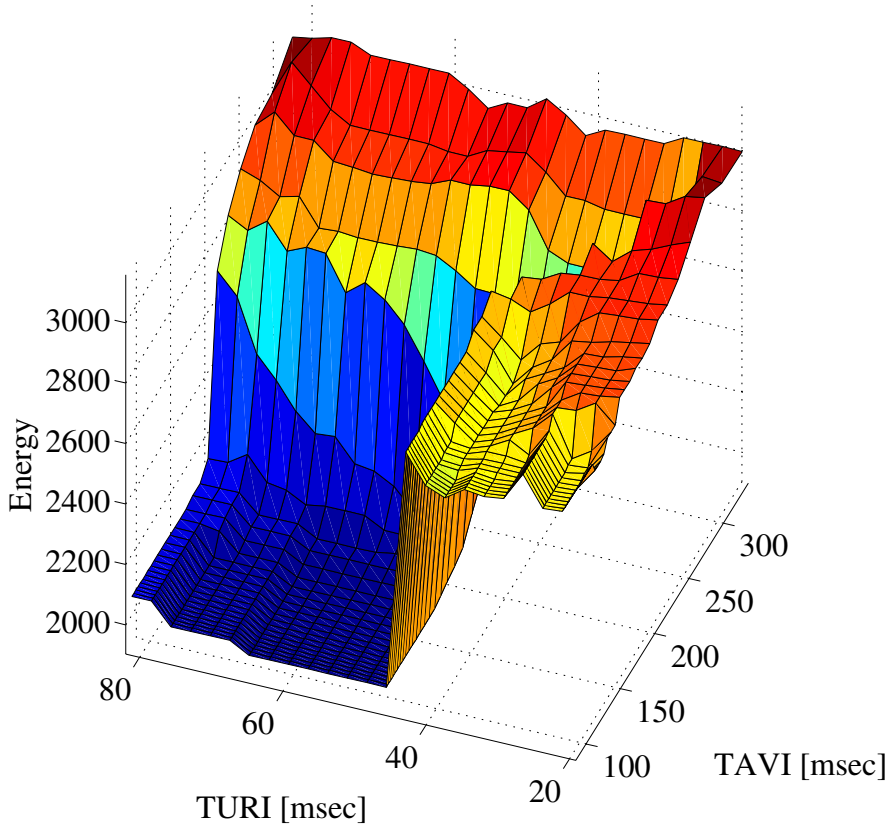
Event counting not expressible in MTL (Metric Temporal Logic)

Framework functionality

- Broad range of techniques
  - Monte-Carlo simulation of composed models
    - with (confidence level) guarantees for non-linear flows
  - (approximate) quantitative verification against variants of MTL
    - to ensure property is satisfied
  - parametric analysis
    - for in silico evaluation, to reduce need for testing on patients
  - automated synthesis of optimal timing parameters
    - to determine delays between paces so that energy usage is optimised for a given patient
  - patient-specific parameterisation
  - hardware-in-the-loop simulation
    - parameter optimisation with respect to real energy measurements

- See http://www.veriware.org/pacemaker.php

Energy consumption



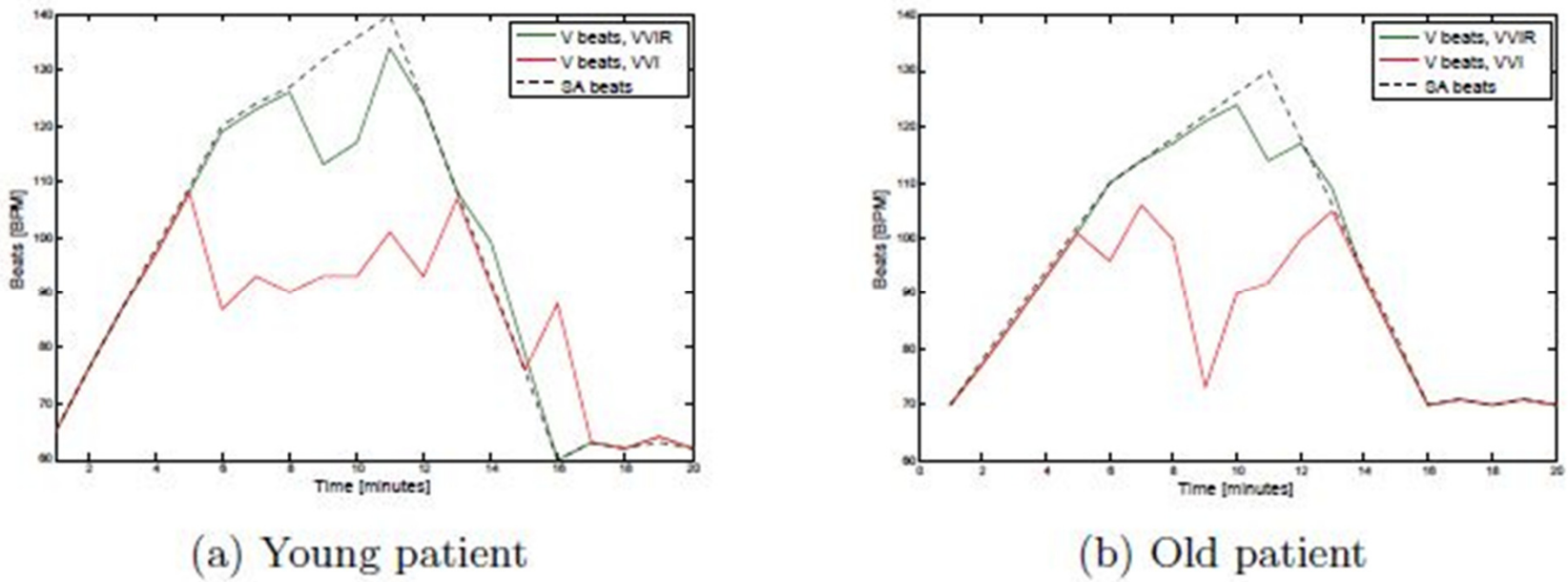
Efficiency "energy consumed must be below some fixed level"

Battery charge in 1 min under Bradycardia, varying timing parameters

Based on real power measurements

Hardware-in-the-loop simulation and energy optimization of cardiac pacemakers.
Barker *et al*, In *Proc EMBC*, 2015

Modulation during physical activity



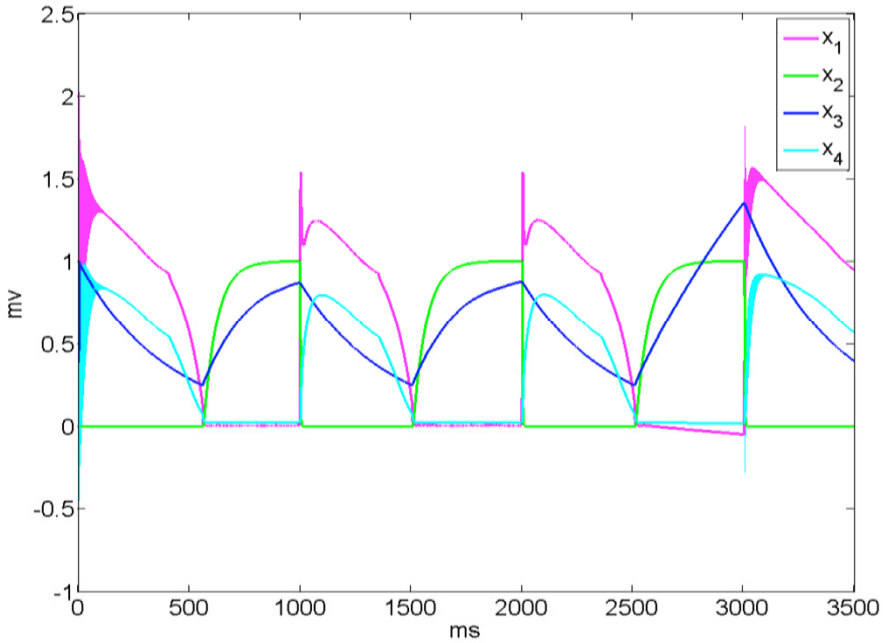
(a) Young patient

(b) Old patient

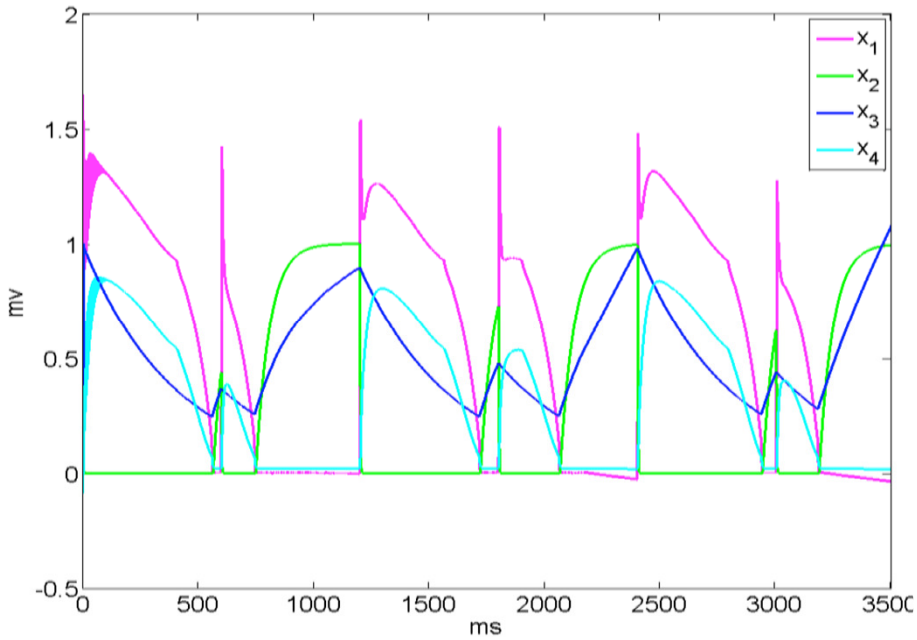
Rate modulation during exercise. Black dashed line indicates metabolic demand, and the green and red curves show rate-adaptive VVIR and fixed-rate VVI pacemakers.

Formal Modelling and Validation of Rate-Adaptive Pacemakers, Kwiatkowska *et a*l. In *IEEE International Conference on Healthcare Informatics*, ACM. 2014

Alternans in the heart



Pacing rate: 1s



Pacing rate: 0.6s

We plot the reach set from a set of initial states observe whether the AP durations alternate

Invariant Verification of Nonlinear Hybrid Automata Networks of Cardiac Cells. Huang *et al* [25]
In *CAV*, volume 8559 of LNCS, pages 373–390, Springer, 2014.

Optimal parameter synthesis

- Automated verification aims to establish if a property holds for a given model
- Can we find a model so that a property is satisfied?
  - difficult…
- The parameter synthesis problem 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
- Objective function
  - maximise volume, or ensure robustness

Synthesising Optimal Timing Delays for Timed I/O Automata. Diciolla et al. In *14th International Conference on Embedded Software (EMSOFT'14)*, ACM. To appear. 2014

Optimal timing delays

- Bi-level optimisation problem
- Safe heart rhythm CMTL property (inner problem)

$$\phi = \Box^{[0,T]} \left(vPeriod \in [500, 1000] \right)$$

  - at any time in [0,T] any two consecutive ventricular beats are between 500 and 1000 ms, i.e. heart rate of 60 and 120 BPM
- Cost function (outer problem)

$$2 \cdot \#_0^{60000} \left(act = AP \right) + 3 \cdot \#_0^{60000} \left(act = VP \right)$$

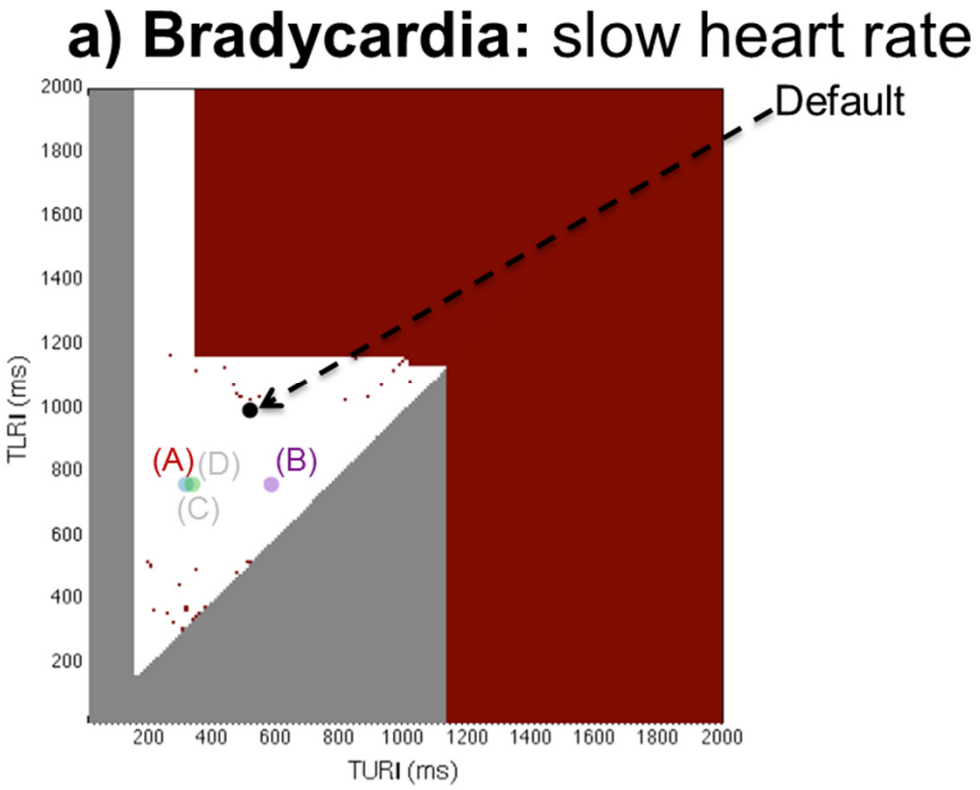
  - energy consumption in 1 minute

$$\frac{\sum_{(\mathbf{q},\eta) \in Vbeat(\rho')} |\eta(CO) - \overline{CO}|}{|Vbeat(\rho')|}$$

  - mean difference between cardiac output and reference value

Synthesis results

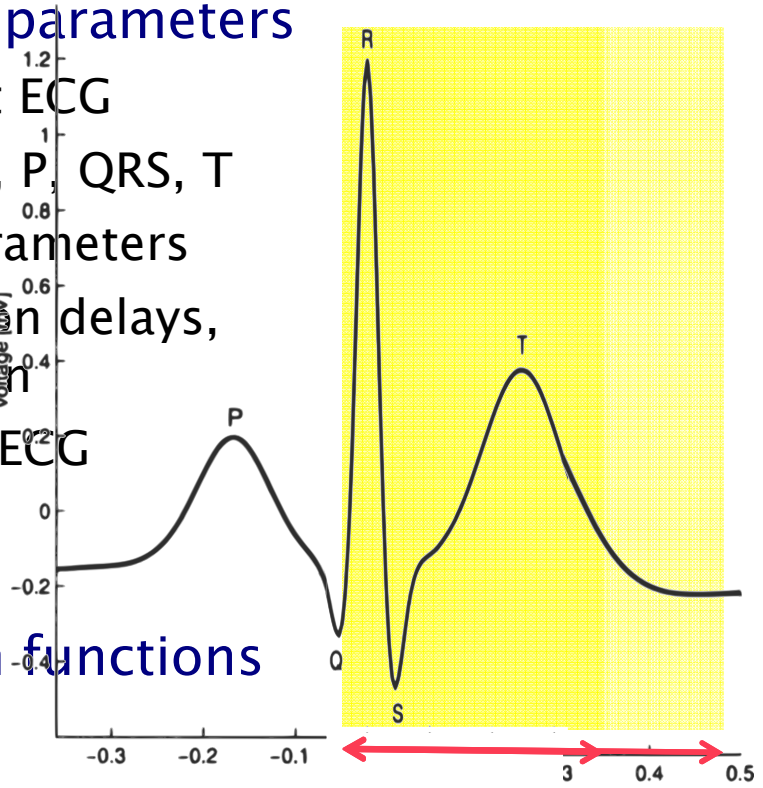
- Solved through SMT encoding (inner problem) combined with evolutionary computation (outer problem)
- Pacemaker parameters:
  - TLRI: time the PM waits before pacing atrium
  - TURI: time before pacing ventricle after atrial event
- Significant improvement (>50%) over default values
  - path 20
- A (exact),B (evo) energy
- C (exact),D (evo) CO
  - evo faster, less precise



Synthesising robust and optimal parameters for cardiac pacemakers using symbolic and evolutionary computation techniques, Kwiatkowska et al.,In *Proc* HSB 2015

Estimation from ECG data

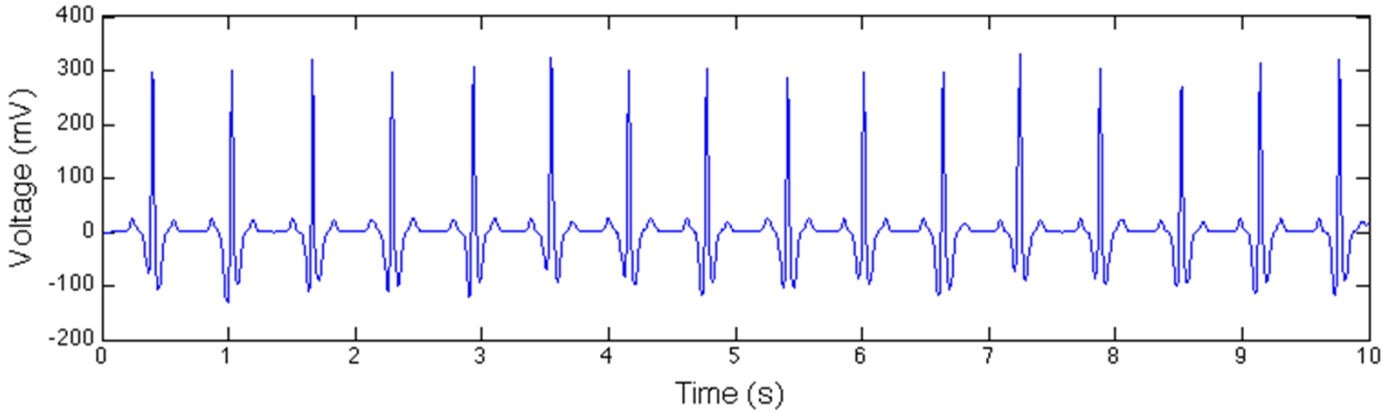
- Steps towards personalisation of parameters
  - filtering and analysis of the input ECG
  - detection of characteristic waves, P, QRS, T
  - mapping of intervals: explicit parameters
  - implicit parameters, eg conduction delays, use Gaussian Process optimisation
  - compare synthetic ECG with real ECG using statistical distance



- Synthetic ECG = sum of Gaussian functions centred at each wave l_i

$$\text{synthECG}(t) = \sum_{i \in \{P,Q,R,S,T\}} \sum_{l_i \in \text{Peaks}_i} a_i \cdot \exp\left(-\frac{(t-l_i)^2}{2c_i^2}\right).$$

Case study: Personalisation



- Personalisation of wearable devices
  - estimate parameters for a heart model based on ECG data
  - generate synthetic ECG
  - useful for model-based development of personalised devices
- Developed HeartVerify based on Simulink/Stateflow
  - variety of tools and techniques
  - http://www.veriware.org/pacemaker.php

Stochastic game modelling

- Control is playing a game
  - find a controller working under all environment conditions, including adversarial
  - some uncertainty is quantifiable: stochastic games
- System built from several components:
  - manage design/engineering complexity
  - need theory for compositional analysis
- Engineering is the art of making trade-offs:
  - designs have to meet multiple quantitative objectives
  - contracts (if I can assume A, I guarantee B)

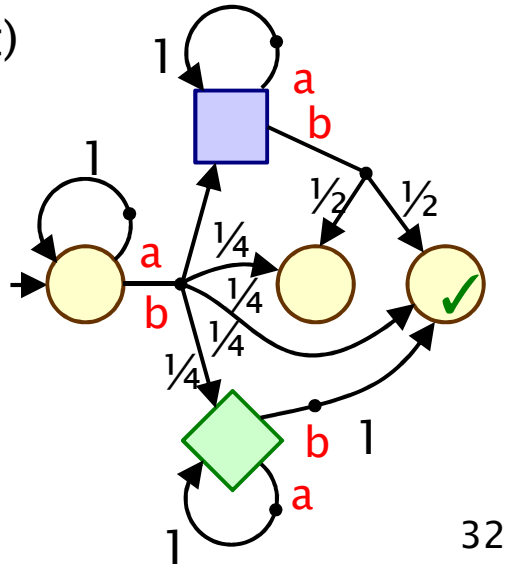
Quantitative Verification and Strategy Synthesis for Stochastic Games, Svorenova et al., *European Control Journal*, to appear 2016

Stochastic multi-player games (SMGs)

- Key ingredients:
  - probability (failures, noisy sensors, randomisation)
  - nondeterminism (concurrency, control, environment)
  - multiple players (competitive or collaborative behaviour)

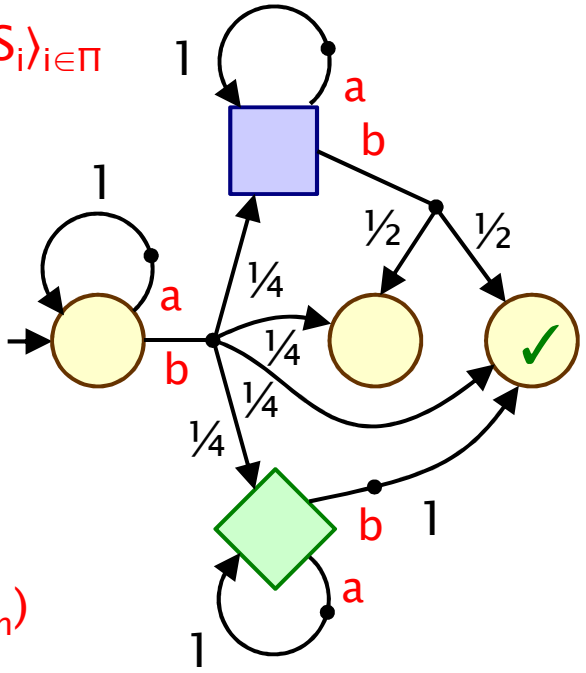
- Applications
  - distributed coordination (selfish agents vs unselfish)
  - controller synthesis (system vs. environment)
  - security (defender vs. attacker)

- Modelling of SMGs
  - PRISM modelling language
  - extension of Reactive Modules
  - guarded command notation

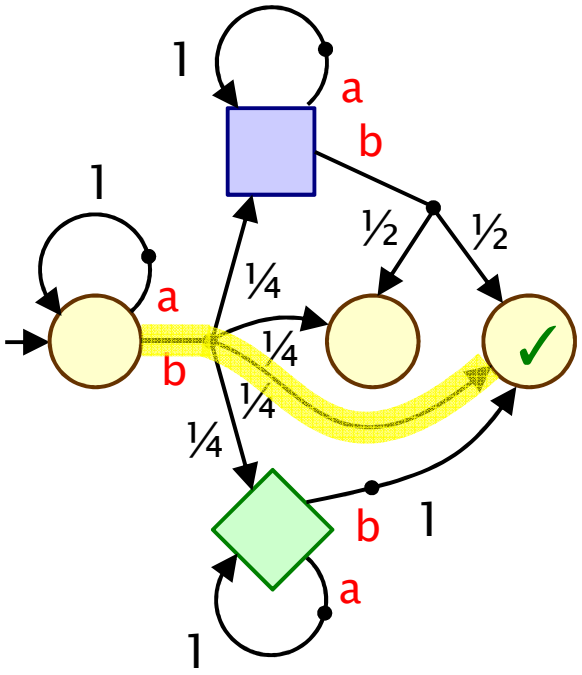


Stochastic multi-player games

- Stochastic multi-player game (SMGs)
  - probability + nondeterminism + multiple players
- A (turn-based) SMG is a tuple $(\Pi, S, \langle S_i \rangle_{i \in \Pi}, A, \Delta, L)$:
  - Π is a set of n players
  - S is a (finite) set of states, with partition $\langle S_i \rangle_{i \in \Pi}$
  - A is a set of action labels
  - $\Delta : S \times A \to \text{Dist}(S)$ is a (partial) transition probability function
  - $L : S \to 2^{AP}$ labels states with atomic propositions from AP
- Strategy for player i: choose in S_i states
  - based on history, i.e. $\sigma_i : (SA)^* S_i \to \text{Dist}(A)$
  - strategy profile: for all players $\sigma = (\sigma_1, \ldots, \sigma_n)$
- Probability measure over paths: $\Pr_s^{\sigma}$



Example



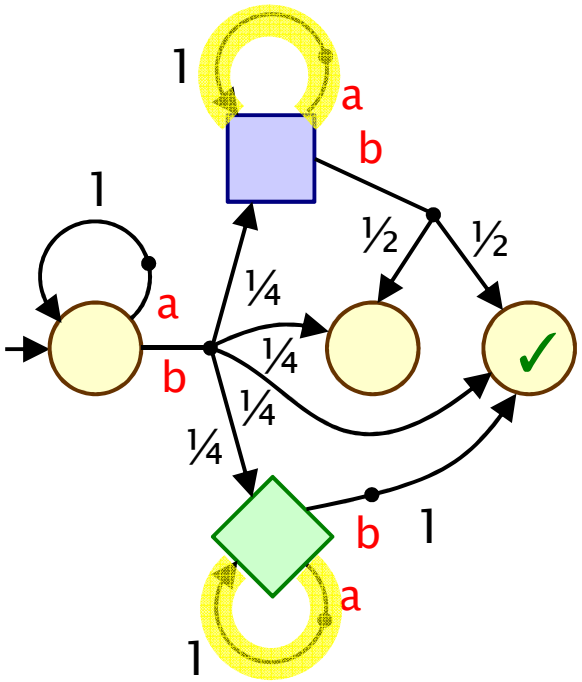
$\langle\langle \bigcirc \rangle\rangle P_{\geq ¼}[\ F\ ✓\ ]$

true in initial state

$\langle\langle \bigcirc \rangle\rangle P_{\geq ⅓}[\ F\ ✓\ ]$

$\langle\langle \bigcirc, \square \rangle\rangle P_{\geq ⅓}[\ F\ ✓\ ]$

Example



$\langle\langle \bigcirc \rangle\rangle P_{\geq ¼}[\ F\ \checkmark\ ]$

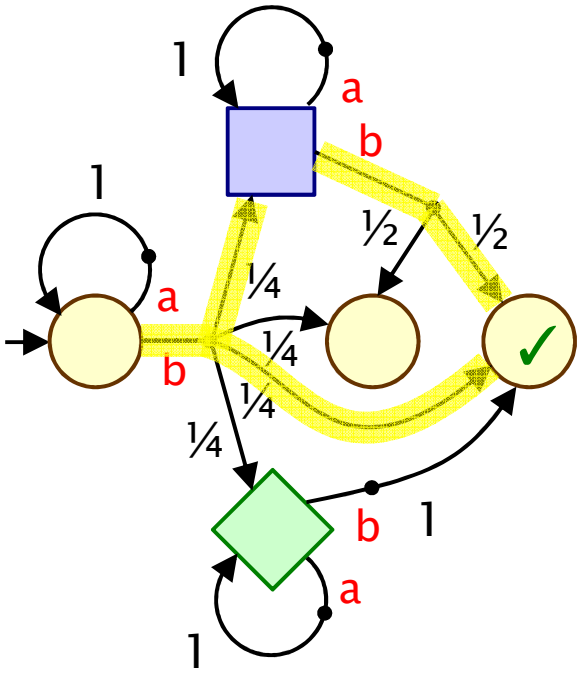
true in initial state

$\langle\langle \bigcirc \rangle\rangle P_{\geq ⅓}[\ F\ \checkmark\ ]$

false in initial state

$\langle\langle \bigcirc, \square \rangle\rangle P_{\geq ⅓}[\ F\ \checkmark\ ]$

Example



$\langle\langle \bigcirc \rangle\rangle P_{\geq 1/4}[\ F\ \checkmark\ ]$

true in initial state

$\langle\langle \bigcirc \rangle\rangle P_{\geq 1/3}[\ F\ \checkmark\ ]$

false in initial state

$\langle\langle \bigcirc, \square \rangle\rangle P_{\geq 1/3}[\ F\ \checkmark\ ]$

true in initial state

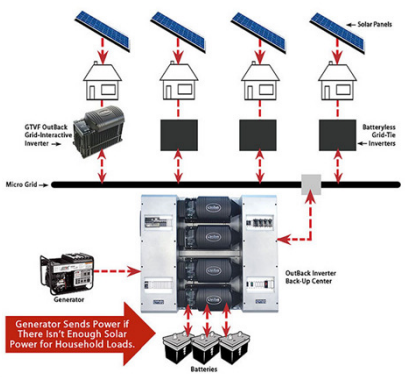
Property specification

- Basis: the temporal logic rPATL
  - combines operators from PCTL with rewards (probabilistic, reward), ATL (strategies of player/coalitions) and

- Examples
  - <<4,5>> P<0.95 [F<=100 "end"]
    "Players 4 and 5 can ensure that the probability of reaching an "end"-state within 100 time-steps is < 0.95"
  - <<p1,p3>> Pmax=? [F "goal"]
    "The maximum probability with which players p1 and p3 can guarantee that a "goal"-state is reached?"
  - <<p1>> R{"r"}max=? [F "success"]
    "Maximum expected value of reward "r" accumulated before reaching "success" that can be guaranteed by player p1?"

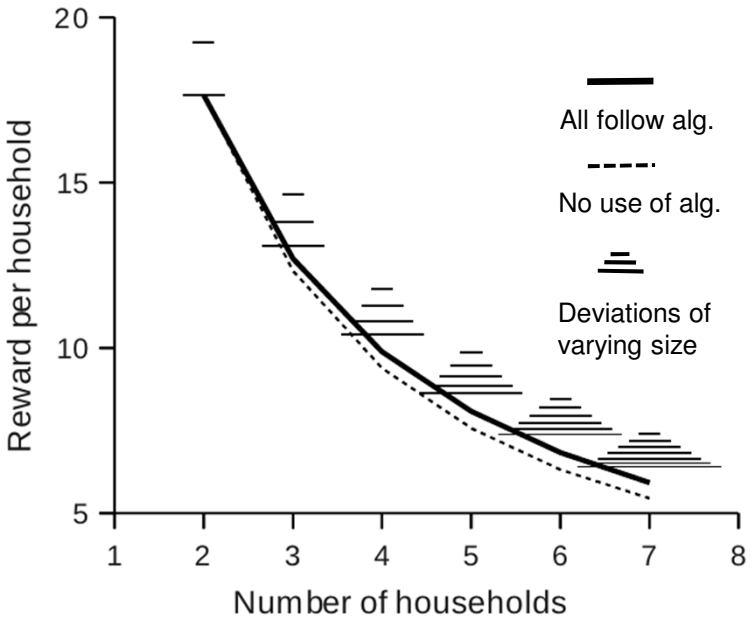
Case study: Energy management

- Energy management protocol for Microgrid
  - Microgrid: local energy management
  - randomised demand management protocol [Hildmann/Saffre'11]
  - probability: randomisation, demand model, …



- Existing analysis
  - simulation-based
  - assumes all clients are unselfish

- Our analysis
  - stochastic multi-player game
  - clients can cheat (and cooperate)
  - exposes protocol weakness
  - propose/verify simple fix



Automatic Verification of Competitive Stochastic Systems, Chen et al., In *Proc* TACAS 2012

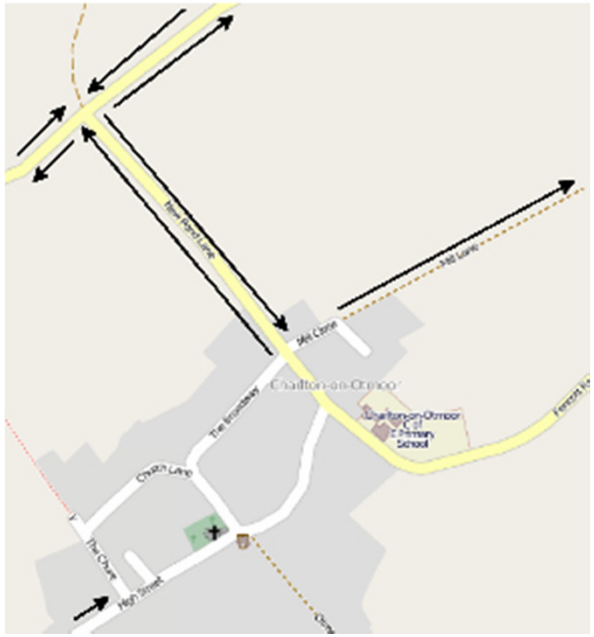
Multi-objective strategy synthesis

- Explore trade-offs
  - e.g. between performance and resource usage
- Extension of rPATL: Boolean combinations of objectives
  - expected total rewards (for stopping games)
  - expected mean-payoffs or ratios (controllable multi-chain)
  - conjunctions of almost sure mean-payoffs/ratios (all games)
- Examples
  - "Player 1 can guarantee that the expected total reward values for reward structures "r1" and "r2" are at least v1 and v2, resp."

    <<1>> (R{"r1"}>=v1 [C] & R{"r2"}>=v2 [C])
  - "Player 1 can guarantee that, whenever the expected ratio of long-run average values for "r1" and "c" is at most v1, then the ratio for "r2" and "c" is at least v2
  - <<1>> (R{"r1"/"c"}<=v1 [S] => R{"r2"/"c"}>=v2 [S])

Case study: Autonomous urban driving

- Inspired by DARPA challenge
  - represent map data as a stochastic game, with environment active, able to select hazards
  - express goals as conjunctions of probabilistic and reward properties
  - e.g. “maximise probability of avoiding hazards and minimise time to reach destination”
- Solution (PRISM-games 2.0)
  - synthesise a probabilistic strategy to achieve the multi-objective goal
  - enable the exploration of trade-offs between subgoals
  - applied to synthesise driving strategies for English villages



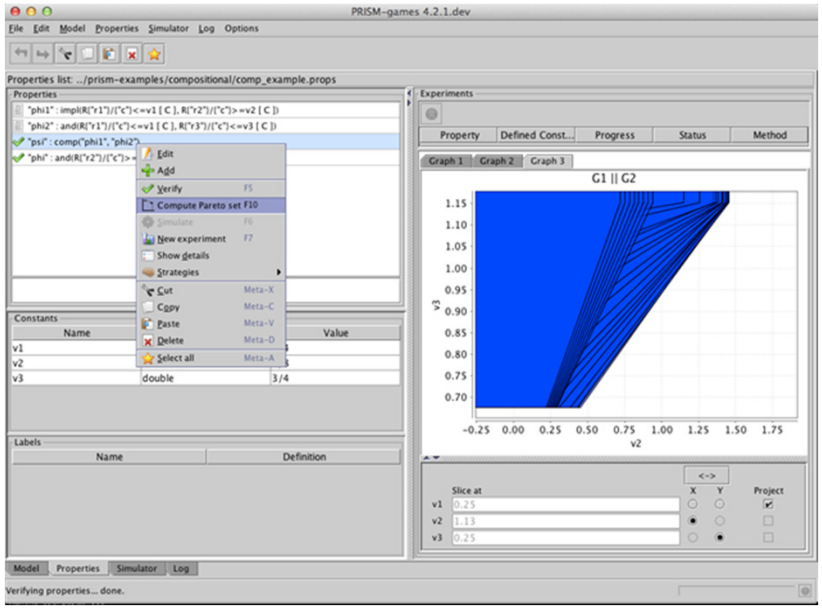
Synthesis for Multi-Objective Stochastic Games: An Application to Autonomous Urban Driving, Chen et al., In *Proc* QEST 2013

Compositional strategy synthesis

- Components
  - reduce design complexity, increase reliability via redundancy
  - improve scalability of analysis, avoid product state space
- Assume-guarantee synthesis:
  - need a strategy for the full system satisfying a global property
  - synthesise one strategy per component, for local properties
  - use assume-guarantee rules to compose local strategies
- Example: local strategies for $G_1 \models \phi^A$ and $G_2 \models \phi^A \Rightarrow \phi^B$ compose to a global strategy for $G_1 \| G_2 \models \phi^B$
- Need to extend synthesis methods:
  - multi-objective properties to use in local and global properties
  - long-run properties (e.g. ratios of rewards) to be compatible with fairness requirements for assume-guarantee rules

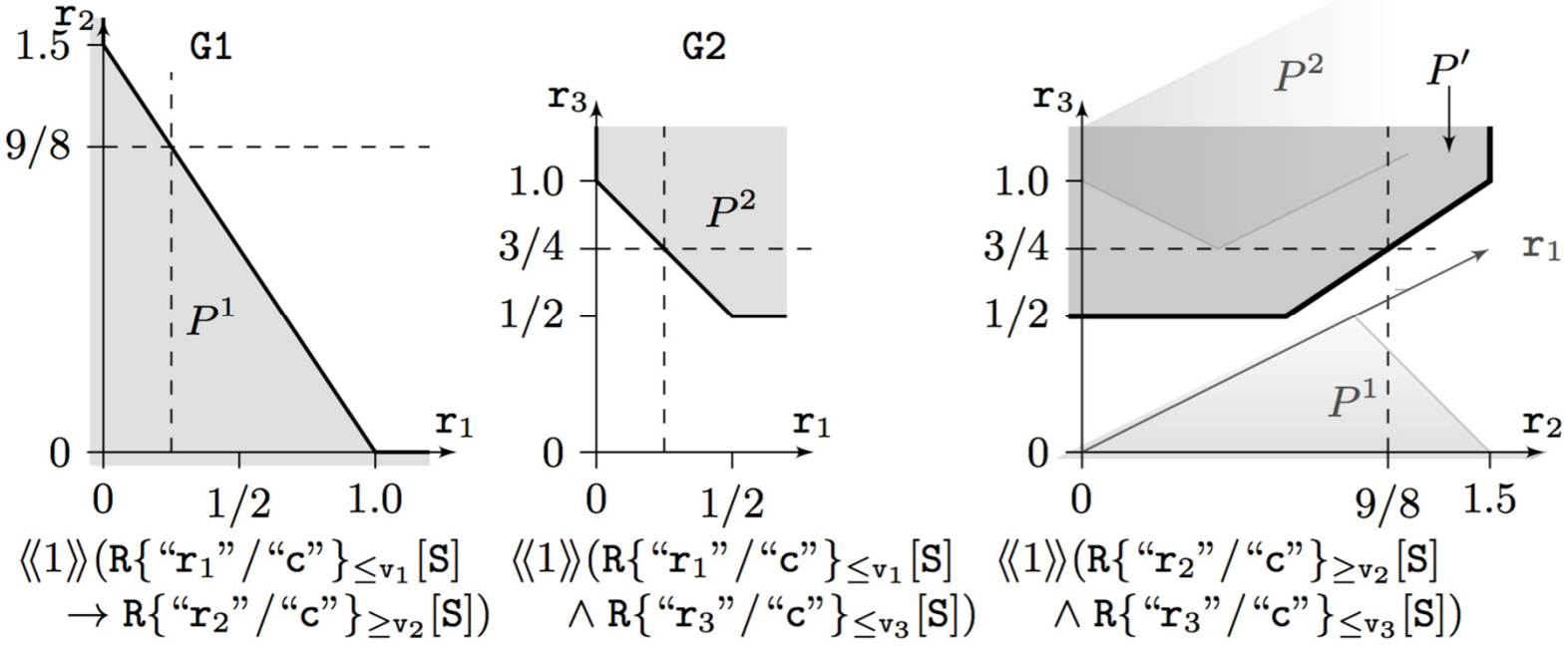
Computation of Pareto sets

- Multi-objective strategy synthesis
  - epsilon-optimal strategies, randomised
  - value iteration over polytopic sets
  - stochastic memory update representation
- Pareto sets
  - optimal achievable trade-offs between objectives
- Visualisation of high-dimensional Pareto sets
  - projection
  - slicing



Compositional strategy synthesis

- Based on assume-guarantee contracts over component interfaces
- Synthesise local strategies for components, then compose into a global strategy using assume-guarantee rules
- Under-approximation of Pareto sets



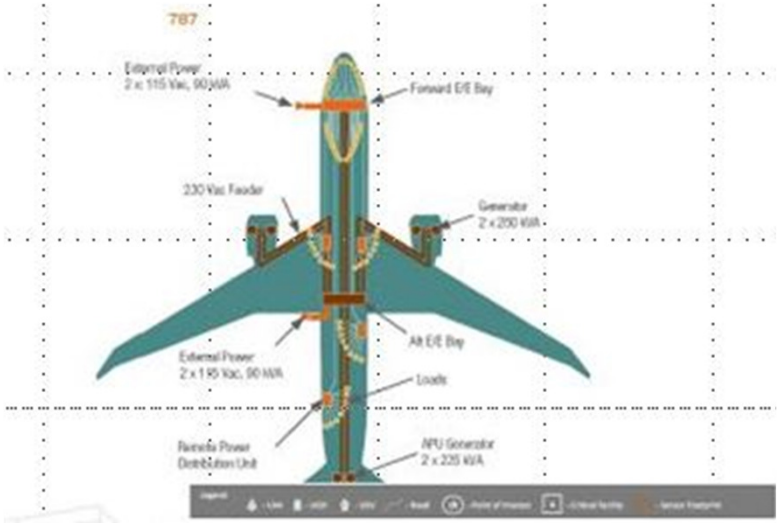
$\langle\langle 1 \rangle\rangle (\mathrm{R}\{\text{“}\mathtt{r}_1\text{”}/\text{“}\mathtt{c}\text{”}\}_{\leq \mathtt{v}_1}[\mathrm{S}] \rightarrow \mathrm{R}\{\text{“}\mathtt{r}_2\text{”}/\text{“}\mathtt{c}\text{”}\}_{\geq \mathtt{v}_2}[\mathrm{S}])$

$\langle\langle 1 \rangle\rangle (\mathrm{R}\{\text{“}\mathtt{r}_1\text{”}/\text{“}\mathtt{c}\text{”}\}_{\leq \mathtt{v}_1}[\mathrm{S}] \wedge \mathrm{R}\{\text{“}\mathtt{r}_3\text{”}/\text{“}\mathtt{c}\text{”}\}_{\leq \mathtt{v}_3}[\mathrm{S}])$

$\langle\langle 1 \rangle\rangle (\mathrm{R}\{\text{“}\mathtt{r}_2\text{”}/\text{“}\mathtt{c}\text{”}\}_{\geq \mathtt{v}_2}[\mathrm{S}] \wedge \mathrm{R}\{\text{“}\mathtt{r}_3\text{”}/\text{“}\mathtt{c}\text{”}\}_{\leq \mathtt{v}_3}[\mathrm{S}])$

Case study: Aircraft power distribution

- Consider Honeywell high-voltage AC (HVAC) subsystem
  - power routed from generators to buses through switches
  - represent as a stochastic game, modelling competition for buses, with stochasticity used to model failures
  - specify control objectives in LTL using longrun average
  - e.g. “maximise uptime of the buses and minimise failure rate”



- Solution (PRISM-games 2.0)
  - compositional strategy synthesis
  - enable the exploration of trade-offs between uptime of buses and failure rate

Compositional Controller Synthesis for Stochastic Games, Basset et al., In *Proc* CONCUR 2014

Tool support: PRISM-games 2.0

- Model checker for stochastic games
  - integrated into PRISM model checker
  - using new explicit-state model checking engine
- SMGs added to PRISM modelling language
  - guarded command language, based on reactive modules
  - finite data types, parallel composition, proc. algebra op.s, …
- rPATL added to PRISM property specification language
  - implemented value iteration based model checking
- Supports strategy synthesis
  - single and multiple objectives, Pareto curve
  - total expected reward, longrun average, ratio rewards
  - compositional strategy synthesis
- Available now:
  - http://www.prismmodelchecker.org/games/

Case studies

- Evaluated on several case studies:
  - team formation protocol [CLIMA'11]
  - futures market investor model [McIver & Morgan]
  - collective decision making for sensor networks [TACAS'12]
  - energy management in microgrids [TACAS'12]
  - reputation protocol for user-centric networks [SR'13]
  - DNS bandwidth amplification attack [Deshpande et a]
  - self-adaptive software architectures [Camara, Garlan et al]
- Case studies using PRISM-games 2.0 functionality:
  - autonomous urban driving (multi-objective) [QEST'13]
  - UAV path planning with operator (multi-objective) [ICCPS'15]
  - aircraft electric power control (compositional) [TACAS'15]
  - temperature control (compositional) [Wiltsche PhD]

PRISM-games 2.0: A Tool for Multi-Objective Strategy Synthesis for Stochastic Games.
Kwiatkowska et al., In *Proc* TACAS 2016

Summing up…

- CPSs take up is fast increasing
  - implantable, closed-loop and wearable devices
  - software integrated and critical component
  - 24/7 health performance expectation
  - safety-critical context, software failures on the increase

- Future work
  - huge models!
  - more general models, more sensor types
  - need approximation, not discretisation
  - symbolic implementation
  - automatically inferring game decomposition and rules
  - synthesis for concurrent and non-zero sum games

Challenges

- Model expressiveness
  - full class of stochastic hybrid models
- Partial information
  - currently largely limited to complete information
- Model learning and adaptation from data
  - self-aware, self-adaptive systems
- Model synthesis from specifications
  - not just parameter synthesis
- Scalability and efficiency
- Modelling social interactions
  - e.g. trust
- Certification
  - incl. tools and standards

Announcing…

SOCIAL TRUST IN AUTONOMOUS ROBOTS

http://qav.cs.ox.ac.uk/trust_in_autonomy/

How to understand,
formalise and express trust?

Open for poster submissions

Workshop of Robotics: Science and Systems 2016

JUNE 19 2016
University of Michigan

sociology
philosophy
cognitive science
ethics logic
computation

EPSRC
Engineering and Physical Sciences Research Council

Acknowledgements

- My group and collaborators in this work
- Project funding
  - ERC Advanced Grant, ERC Proof of Concept
  - Oxford Martin School, Institute for the Future of Computing

- See also
  - VERIWARE www.veriware.org
  - PRISM www.prismmodelchecker.org