Model checking and strategy synthesis for mobile autonomy: from theory to practice

Marta Kwiatkowska

Department of Computer Science, University of Oxford

ICALP 2016, Rome, 15th July 2016

Mobile autonomy is here

It’s not that simple…

Google self-driving car collides with bus in California, accident report says

If it is determined the Google vehicle caused the crash, it would be the first time one of its SUVS caused an accident while in autonomous mode

Software everywhere

- Users expect: predictability & high integrity in presence of
  - component failure, environmental uncertainty, …
  - can be quantified probabilistically

- Quantitative properties
  - safety, reliability, performance, …
  - “the probability of an airbag failing to deploy within 0.02s”
- Quantitative verification to the rescue
  - temporal logic specifications
  - formal verification

Quantitative verification

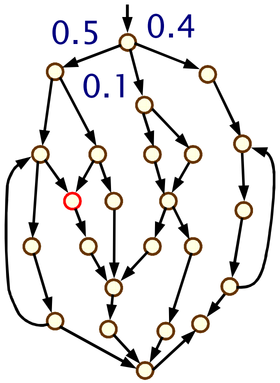
- Employ (quantitative) formal models
  - can be derived or extracted from code
  - can also be used at runtime
- Specify goals/objectives/properties in temporal logic:
  - reliability, energy efficiency, performance, resource usage, …
  - (reliability) “alert signal will be delivered with high probability in 10ms”, for in-car communication
  - (energy) “maximum expected energy consumption in 1 hr is at most 10mA”, for an autonomous robot
- Focus on automated, tool-supported methodologies
  - model-based design
  - automated verification via model checking
  - strategy synthesis from (temporal logic) specifications

Quantitative/probabilistic verification

Automatic verification and strategy synthesis from quantitative properties for probabilistic models

System

Probabilistic model
e.g. Markov chain



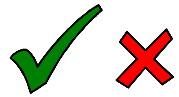
$P_{<0.01}$ [$F^{\leq t}$ crash]

System require-ments

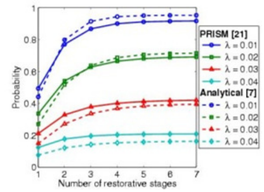
Probabilistic temporal logic specification
e.g. PCTL, CSL, LTL

Probabilistic model checker
e.g. PRISM

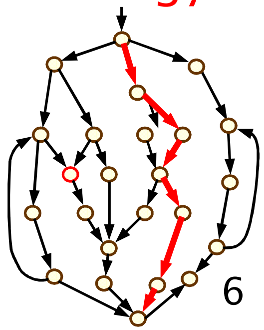
Result



Quantitative results



Strategy



Historical perspective

- First algorithms proposed in 1980s
  - algorithms [Vardi, Courcoubetis, Yannakakis, …]
  - [Hansson, Jonsson, de Alfaro] & first implementations
- 2000: general purpose tools released
  - PRISM: efficient extensions of symbolic model checking [Kwiatkowska, Norman, Parker, …]
  - ETMCC: 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
  - www.prismmodelchecker.org

But which modelling abstraction?

- Several probabilistic models supported…
- Markov chains (DTMCs and CTMCs)
  - discrete states + discrete or exponential probability
  - for: component failures, unreliable communication media, …
- Markov decision processes (MDPs)
  - probability + decisions (nondeterministic choices)
  - for: distributed coordination, motion planning in robotics, …
- Probabilistic timed automata (PTAs)
  - probability + decisions+ real-time passage
  - for: wireless comm. protocols, embedded control systems, …
- Towards stochastic cont.space/hybrid systems (LMPs, SHSs)
  - probability + decisions + continuous flows
  - for: control of physical processes, motion in space, …

The challenge of mobile autonomy

- Autonomous systems
  - are reactive, continuously interact with their environment
    - including other components or human users, adversarial
  - have goals/objectives
    - often quantitative, may conflict
  - take decisions based on current state and external events
- Natural to adopt a game-theoretic view
  - need to account for the uncontrollable behaviour of components, possibly with differing/opposing goals
  - in addition to controllable events
- Many occurrences in practice
  - e.g. decision making in economics, power distribution networks, controller synthesis, motion planning, security, distributed consensus, energy management, sensor network co-ordination, …

What makes a game?

- Players with moves (turn-based or concurrent)
- Strategy for each player
  - plans for how to choose moves, based on information available
- Value (or payoff) for each player
- Winning
  - corresponds to optimising the value no matter how the others play the game
- Main question: is there a winning strategy?

Playing games with Google car…

http://theoatmeal.com/blog/google_self_driving_car

Google's view

"This is a classic example of the negotiation that's a normal part of driving – we're all trying to predict each other's movements. In this case, we clearly bear some responsibility, because if our car hadn't moved there wouldn't have been a collision".

Google self-driving car collides with bus in California, accident report says

If it is determined the Google vehicle caused the crash, it would be the first time one of its SUVS caused an accident while in autonomous mode

This lecture…

- Introduce stochastic multi-player games (SMGs)
  - argue that games are an appropriate modelling abstraction for competitive behaviour, in adversarial environments
  - stochasticity to model e.g. failure, sensor uncertainty
- Property specification: rPATL
  - single-objective properties
  - verification
  - strategy synthesis
- Extensions
  - multiobjective properties, Pareto sets
  - compositional strategy synthesis
- Tool support: PRISM-games 2.0
- Case studies

Model Checking and Strategy Synthesis for Stochastic Games: From Theory to Practice.
In *Proc. 43rd ICALP*. To appear, 2016.

Stochastic multi-player games (SMGs)

- A stochastic game involves
  - multiple players (competitive or collaborative behaviour)
  - nondeterminism (decisions, control, environment)
  - probability (failures, noisy sensors, randomisation)

- Here consider only games that are
  - turn-based, discrete time, zero sum, complete observation
  - timed/continuous extensions exist, but tool support lacking

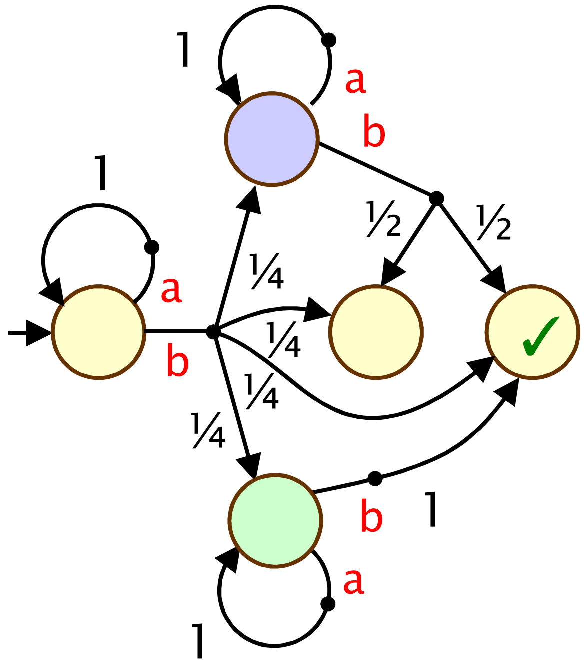
- Widely studied, esp. algorithmic complexity, many applications
  - autonomous traffic (risk averse vs risk taking)
  - distributed coordination (selfish agents vs unselfish)
  - controller synthesis (system vs. environment)
  - security (defender vs. attacker)

Stochastic multi-player games

- Stochastic multi-player game (SMGs)
  - multiple players + nondeterminism + probability
  - generalisation of MDPs: each state controlled by unique player

- 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
  - $\langle S_i \rangle_{i \in \Pi}$ is a partition of S
  - A is a set of action labels
  - $\Delta : S \times A \rightarrow Dist(S)$ is a (partial) transition probability function
  - $L : S \rightarrow 2^{AP}$ is a labelling with atomic propositions from AP

- NB tool does not support concurrent games



Rewards

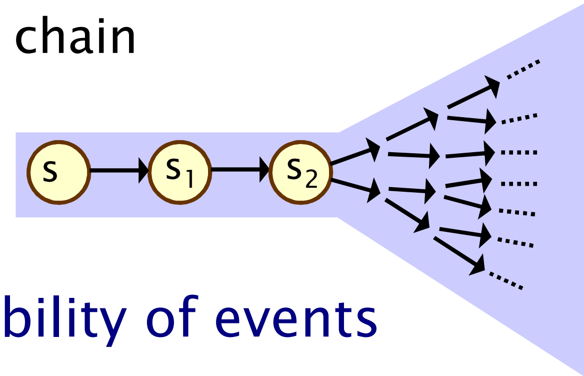
- Annotate SMGs with rewards (or costs)
  - real-valued quantities assigned to states and/or transitions
- Wide range of possible uses:
  - elapsed time, power consumption, number of messages successfully delivered, net profit, …
- We work with:
  - state rewards: $r : S \rightarrow \mathbb{R}_{\geq 0}$
  - action rewards: $r : A \rightarrow \mathbb{R}_{\geq 0}$
- Form basis for a variety of quantitative objectives
  - expected cumulative (total) reward (denoted C)
  - mean-payoff (limit-average) reward (denoted S)
  - ratio reward
  - (and many more not considered here)

Paths, strategies + probabilities

- A path is an (infinite) sequence of connected states in SMG
  - i.e. $s_0 a_0 s_1 a_1 \ldots$ such that $a_i \in A(s_i)$ and $\Delta(s_i,a_i)(s_{i+1})>0$ for all i
  - represents a system execution (i.e. one possible behaviour)
  - to reason formally, need a probability space over paths
- A strategy for player $i \in \Pi$ resolves choices in S_i states
  - based on history of execution so far
  - i.e. a function $\sigma_i : (SA)^*S_i \to Dist(A)$
  - Σ_i denotes the set of all strategies for player i
  - deterministic if σ_i always gives a Dirac distribution
  - memoryless if $\sigma_i(s_0 a_0 \ldots s_k)$ depends only on s_k
  - also finite-memory, infinite memory, …
  - history based or explicit memory representation
- A strategy profile is tuple $\sigma=(\sigma_1,\ldots,\sigma_n)$
  - combining strategies for all n players

Paths, strategies + probabilities…

- For a strategy profile σ:
  - the game's behaviour is fully probabilistic
  - essentially an (infinite-state) Markov chain
  - yields a probability measure Pr_s^σ over set of all paths $Path_s$ from s



- Allows us to reason about the probability of events
  - under a specific strategy profile σ
  - e.g. any (ω-)regular property over states/actions

- Also allows us to define expectation of random variables
  - i.e. measurable functions $X : Path_s \rightarrow \mathbb{R}_{\geq 0}$
  - $E_s^\sigma [X] = \int_{Path_s} X \, dPr_s^\sigma$
  - used to define expected costs/rewards…

Property specification: rPATL

- Temporal logic rPATL:
  - reward probabilistic alternating temporal logic
- CTL, extended with:
  - coalition operator $\langle\langle C \rangle\rangle$ of ATL (Alternating Temporal Logic)
  - probabilistic operator P of PCTL, where $P_{\bowtie q}[\psi]$ means "the probability of ensuring ψ satisfies $\bowtie q$"
  - reward operator R of PRISM, where $R_{\bowtie q}\ [\rho]$ means "the expected value of ρ satisfies $\bowtie q$"
- Example:
  - $\langle\langle \{1,2\} \rangle\rangle\ P_{<0.01}\ [\ F^{\leq 10}\ error\ ]$
  - "players 1 and 2 have a strategy to ensure that the probability of an error occurring within 10 steps is less than 0.1, regardless of the strategies of other players"

rPATL properties

- Syntax:

$$\phi ::= \langle\langle C\rangle\rangle P_{\bowtie q}[\psi] \mid \langle\langle C\rangle\rangle R^{r}_{\bowtie q}[\rho] \mid \langle\langle C\rangle\rangle R^{r/c}_{\bowtie q}[\rho]$$

$$\psi ::= F\ a$$

$$\rho ::= C \mid S$$

"ratio"

"reachability"

"longrun average"

"cumulative"

- where:
  - $a \in AP$ is an atomic proposition, $C \subseteq \Pi$ is a coalition of players, $\bowtie \in \{\leq,<,>,\geq\}$, $q \in \mathbb{R}_{\geq 0}$, r and c are reward structures

- $\langle\langle C\rangle\rangle P_{\geq 1}[F \text{ "end"}]$
  - "players in coalition C have a collective strategy to ensure that the game reaches an "end"-state almost surely, regardless of the strategies of other players"

rPATL reward properties

- Syntax:

"ratio"

$\phi ::= \langle\langle C \rangle\rangle P_{\bowtie q}[\psi] \mid \langle\langle C \rangle\rangle R^{r}_{\bowtie q}[\rho] \mid \langle\langle C \rangle\rangle R^{r/c}_{\bowtie q}[\rho]$

$\psi ::= F\ a$ "reachability"

$\rho ::= C \mid S$ "longrun average"

"cumulative"

- $\langle\langle C \rangle\rangle R^{fuel}_{<q}\ [C]$
  - "players in coalition C have a strategy to ensure that the expected total fuel consumption is less than q, regardless of the strategies of other players"
- $\langle\langle C \rangle\rangle R^{fuel/time}_{\leq q}\ [S]$
  - "players in coalition C have a strategy to ensure that the expected longrun fuel consumption per time unit is at most q, regardless of the strategies of other players"

rPATL semantics

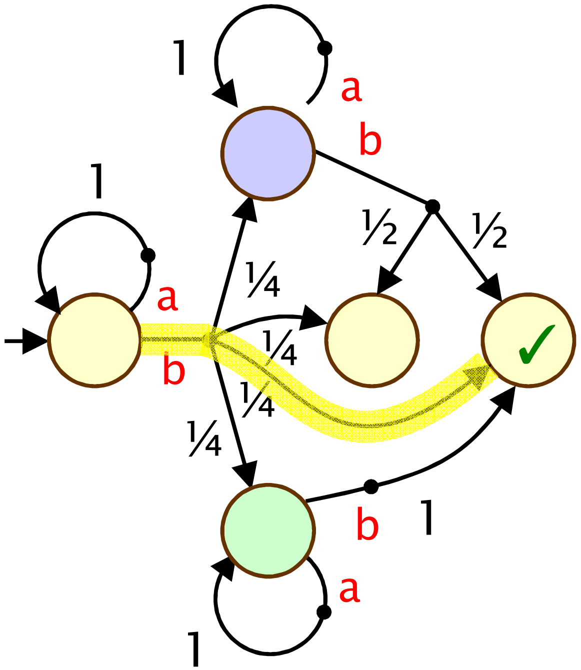
- Semantics for most operators is standard
- Just focus on P and R operators…
  - use reduction to a stochastic 2-player game

- Coalition game G_C for SMG G and coalition $C \subseteq \Pi$
  - 2-player SMG where C and $\Pi \backslash C$ collapse to players 1 and 2

- $\langle\langle C \rangle\rangle P_{\bowtie q}[\psi]$ is true in state s of G iff:
  - in coalition game G_C:
  - $\exists \sigma_1 \in \Sigma_1$ such that $\forall \sigma_2 \in \Sigma_2$. $Pr_s^{\sigma_1, \sigma_2}(\psi) \bowtie q$

- Semantics for R operator defined similarly…

Examples



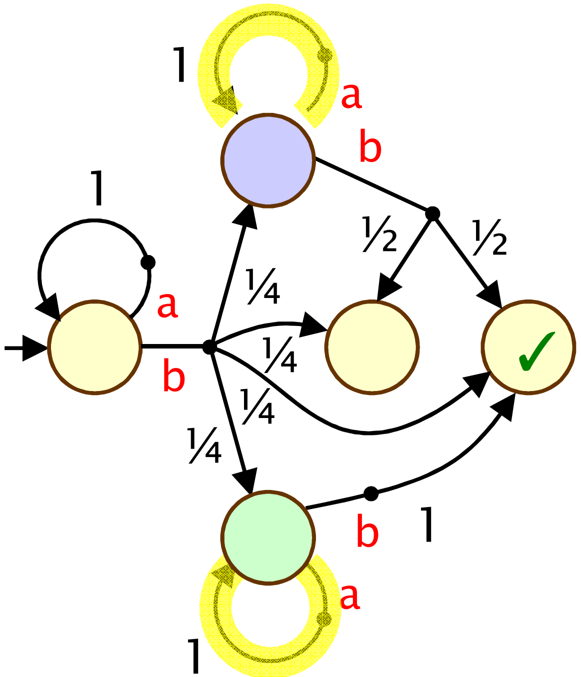
⟨⟨◯⟩⟩$P_{\geq 1/4}$[F ✓]

true in initial state

⟨⟨◯⟩⟩$P_{\geq 1/3}$ [F ✓]

⟨⟨◯,◯⟩⟩$P_{\geq 1/3}$ [F ✓]

Examples



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

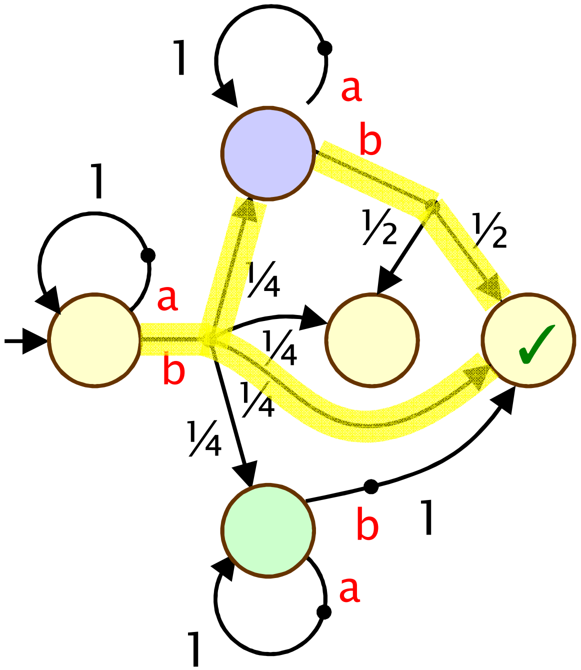
true in initial state

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

false in initial state

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

Examples



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

true in initial state

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

false in initial state

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

true in initial state

Verification and strategy synthesis

- The verification problem is:
  - Given a game G and rPATL property ϕ, does G satisfy ϕ?
- e.g. $\langle\langle C\rangle\rangle P_{\bowtie q}[\psi]$ is true in state s of G iff:
  - in coalition game G_C:
  - $\exists \sigma_1 \in \Sigma_1$ such that $\forall \sigma_2 \in \Sigma_2$. $Pr_s^{\sigma_1,\sigma_2}(\psi) \bowtie q$
- The synthesis problem is:
  - Given a game G and a coalition property ϕ, find, if it exists, a coalition strategy σ that is a witness to G satisfying ϕ
- Reduce to computing optimal values and winning strategies in 2-player games
  - e.g. $\langle\langle C\rangle\rangle P_{\geq q}[\psi] \;\Leftrightarrow\; \sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} Pr_s^{\sigma_1,\sigma_2}(\psi) \geq q$
  - complexity NP $\cap$ coNP (this fragment), cf P for MDPs

Verification and strategy synthesis

- The verification problem is:
  - Given a game G and rPATL property ϕ, does G satisfy ϕ?
- The synthesis problem is:
  - Given a game G and a coalition property ϕ, find, if it exists, a coalition strategy σ that is a witness to G satisfying ϕ
- Reduce to computing optimal values and winning strategies in 2-player games
  - typically employ value iteration to specified convergence
  - both players have optimal strategies
  - memoryless deterministic strategies suffice
  - (epsilon-optimal) strategies can be typically extracted from optimal values in linear time

Multi-objective properties

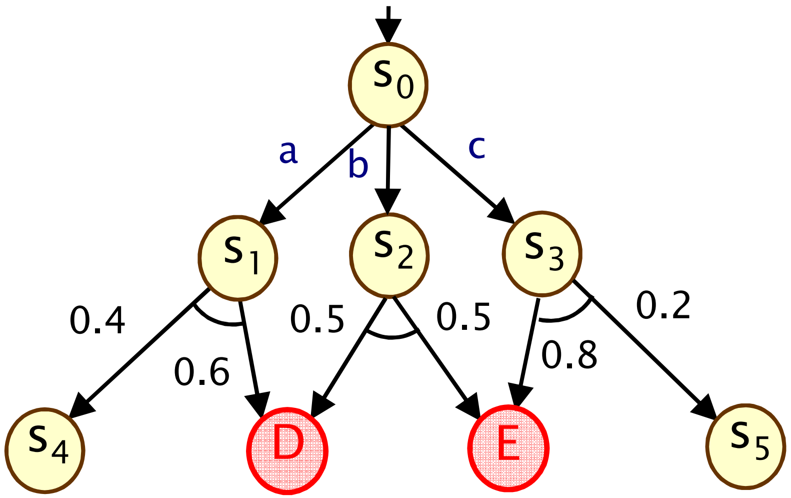
- May need to explore trade-offs
  - e.g. between performance and resource usage: maximise probability of success and minimise energy usage
- Consider conjunctions of objectives (for stopping games), also known as multidimensional
  - expected total rewards, mean-payoffs or ratios
  - almost sure mean-payoffs/ratios
- Example
  - "the expected longrun average fuel consumption and profit are simultaneously at least v1and v2, respectively "

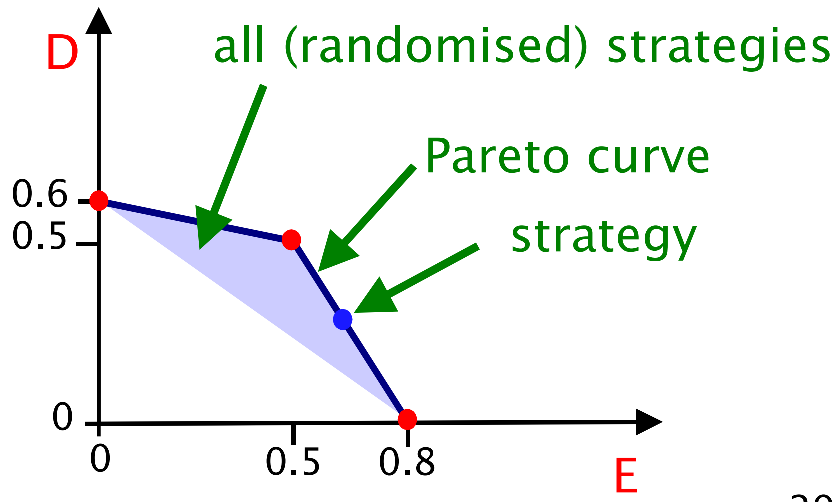
    $\langle\langle C\rangle\rangle$ ($R^{fuel}_{\geq v1}$ [S] & $R^{profit}_{\geq v2}$ [S])
- NB Boolean combinations may be needed for implication

  $\langle\langle C\rangle\rangle$ ($R^{fuel/time}_{\geq v1}$ [S] $\Rightarrow$ $R^{profit}_{\geq v2}$ [S])

Example of Pareto optimality

- Consider the simpler scenario of MDPs (1½ player games)
- Pareto optimum for conjunction of two objectives
  - probability of reaching D is greater than 0.2 <u>and</u>
  - probability of reaching E is greater than 0.6
- Randomised strategies may be needed…

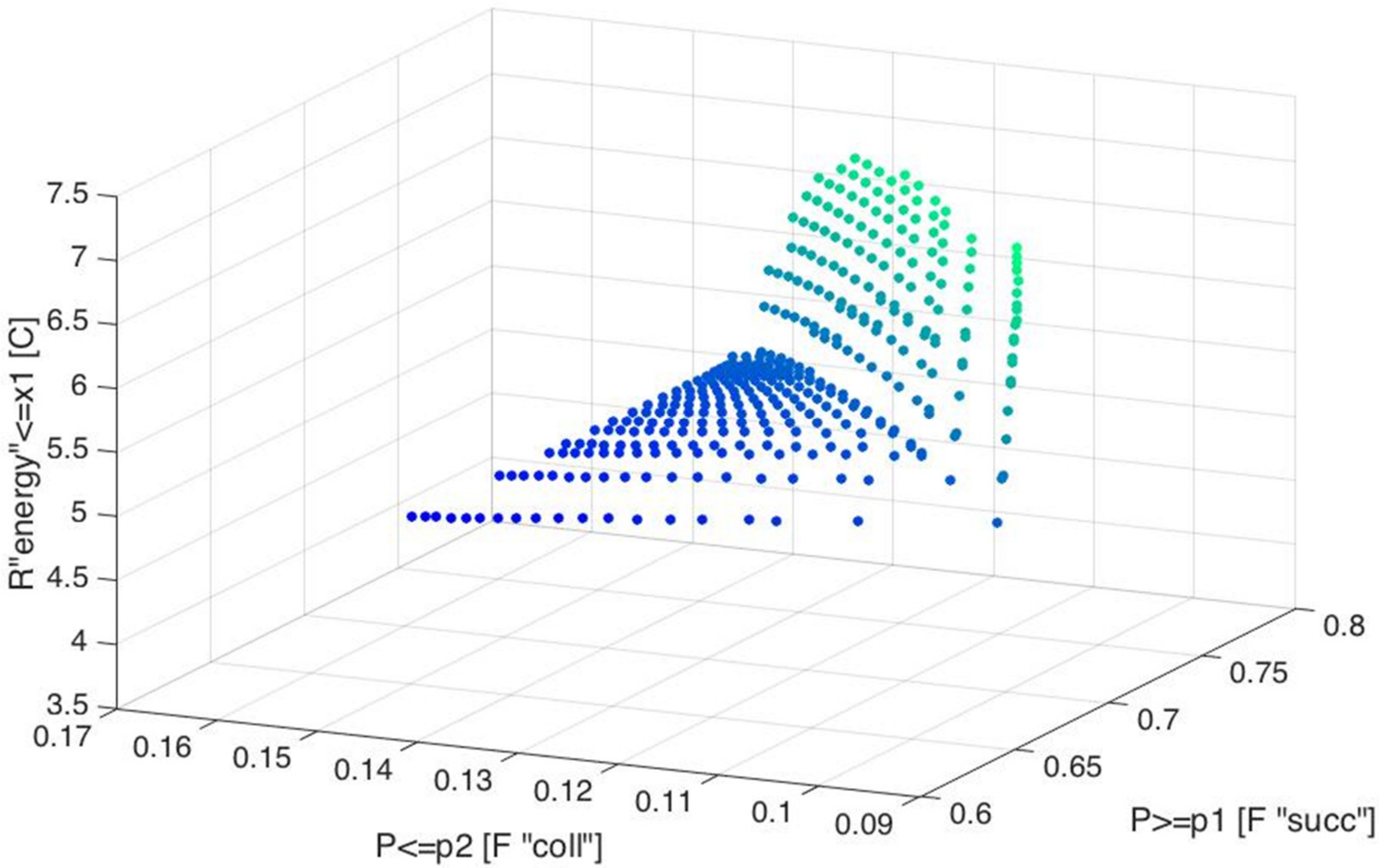




Multi-objective properties

- For MDPs, optimal strategies exist but randomised strategies may be needed
- For stochastic games:
  - optimal strategies may not exist
  - infinite memory may be required
- Therefore
  - work with restricted games (e.g. stopping)
  - use stochastic memory update representation [Brazdil et al, 2014]
    - exponentially more succinct than deterministic update
    - equivalent power if infinite memory allowed
- Decision procedure
  - complexity is NP $\cap$ coNP
  - compute epsilon-approximations of Pareto sets and epsilon-optimal strategies, fixed point reached in finitely many steps

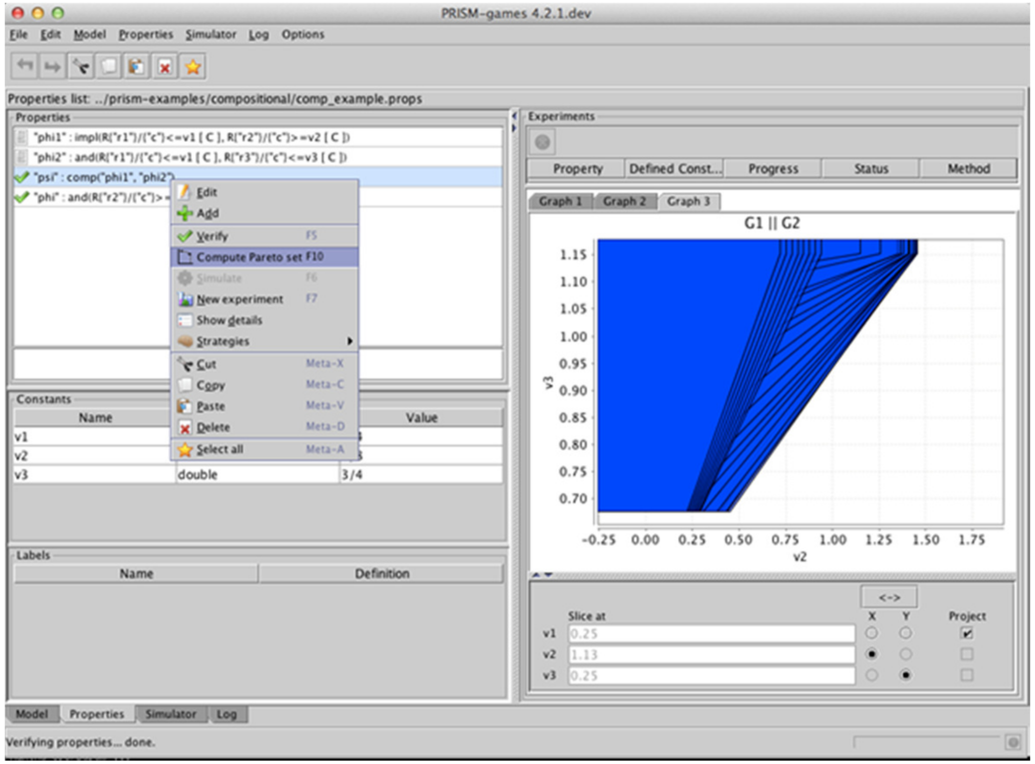
Multidimentional Pareto set



Pareto set approximation for a mixed multi-objective property

Computation of Pareto sets

- Multi-objective strategy synthesis
  - value iteration over polytopic sets
  - maintains a vector of such sets for each state, one for each dimension
- Pareto sets
  - optimal achievable trade-offs between objectives
- Visualisation of high-dimensional Pareto sets
  - projection
  - slicing



Compositional strategy synthesis

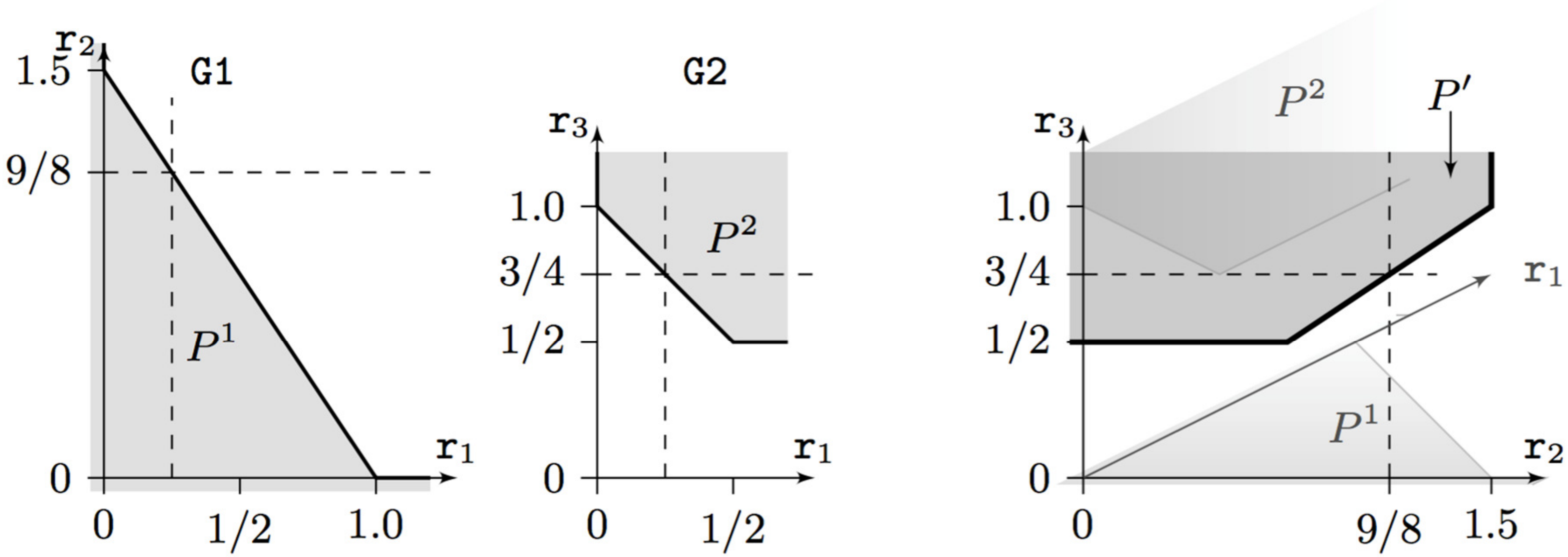
- Componentised games
  - improve scalability of analysis, avoid product state space
  - devise a composition operator for SMGs
- Assume-guarantee synthesis:
  - need a strategy for the full system satisfying a global property
  - synthesise one strategy per component, for local properties
  - use assume-guarantee 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 $G_1 \parallel G_2 \models \phi^B$
- For any player 1 strategy, each game G_i becomes MDP M_i
  - can leverage matching compositional assume-guarantee rules for MDPs, e.g. [Etessami et al 2017][Kwiatkowska et al, 2013]

Compositional strategy synthesis

- 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)
- Example
  - "Player 1 can guarantee that, whenever the expected ratio of longrun 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])
- Employ strategy synthesis on component games:
  - multi-objective properties to use in local and global properties
  - admit also longrun properties (e.g. ratios of rewards)
  - need to consider fairness requirements

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



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

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

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

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/

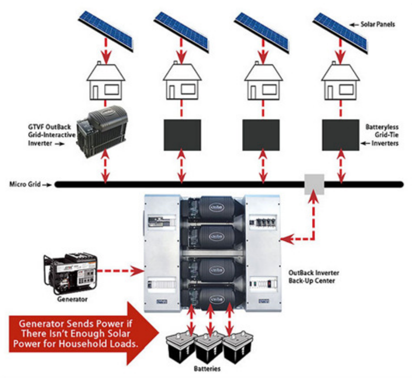
Case studies

- 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 al]
  - self-adaptive software architectures [Camara, Garlan et al]
  - attack-defence scenarios in RFID goods man. [Aslanyan 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]

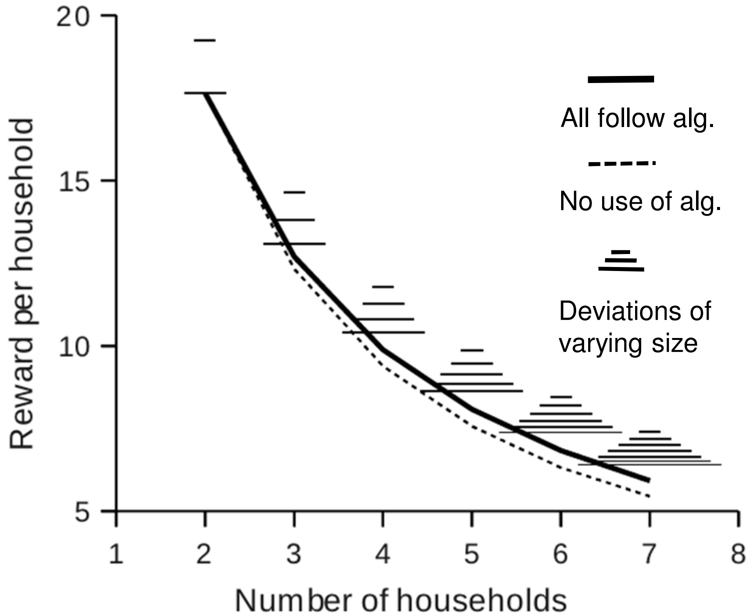
PRISM-games 2.0: A Tool for Multi-Objective Strategy Synthesis for Stochastic Games.
Kwiatkowska et al., In *Proc* TACAS 2016

Case study: Energy management

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



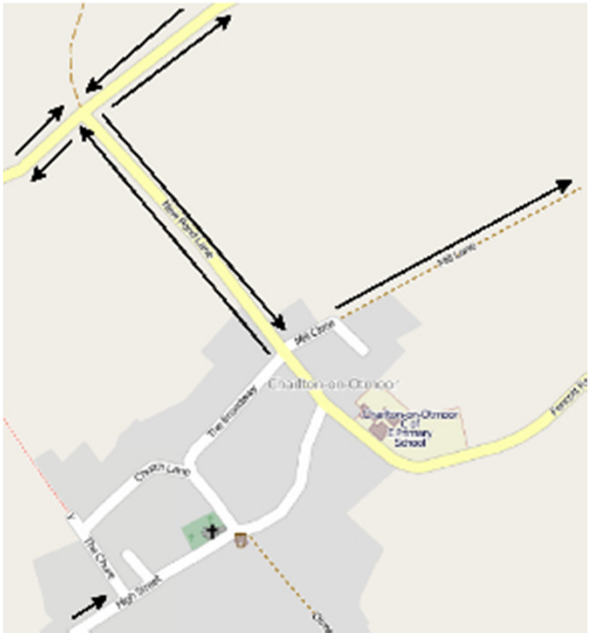
- Existing analysis
  - simulation-based
  - assumes all clients are unselfish
- Our analysis
  - stochastic multi-player game
  - clients can cheat (and cooperate)
  - exposes protocol weakness
  - propose/verify simple fix



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

Case study: Autonomous urban driving

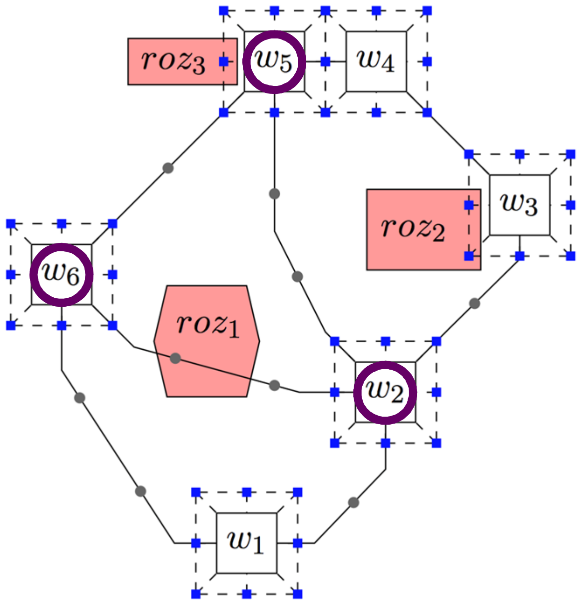
- Inspired by DARPA challenge
  - represent map data as a stochastic game, with environment active, able to select hazards
  - express goals as conjunctions of probabilistic and reward properties
  - e.g. "maximise probability of avoiding hazards and minimise time to reach destination"
- Solution (PRISM-games 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

Case study: UAV path planning

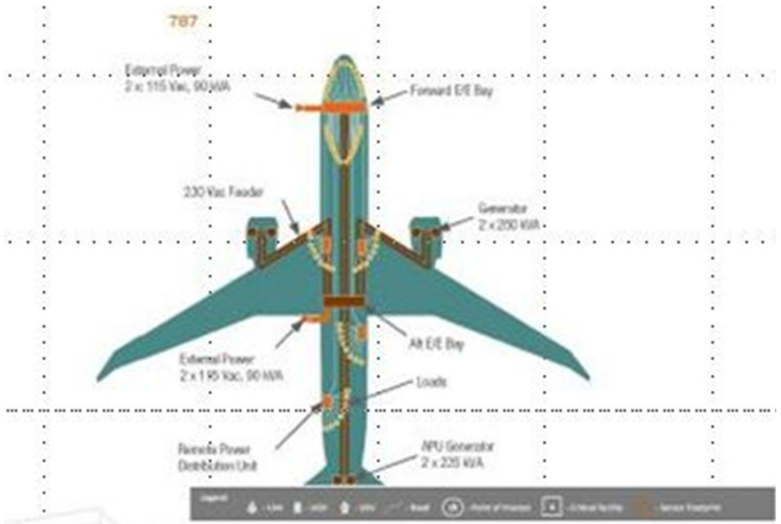
- Human operator
  - sensor tasks
  - high-level commands for piloting
- UAV autonomy
  - low-level piloting function
- Quantitative mission objectives
  - road network surveillance with the minimal time, fuel, or restricted operating zone visits
- Analysis of trade-offs
  - consider operator fatigue and workload
  - multi-objective, MDP and SMG models



Controller Synthesis for Autonomous Systems Interacting with Human Operators. L. Feng et al, In *Proc*. ICCPS 2015, ACM

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

Summary so far…

- What we have shown
  - games can model a wide range of competitive and cooperative scenarios relevant for mobile autonomy
  - variety of quantitative objectives
  - multi-objective properties
  - compositional synthesis via assume-guarantee rules
  - implementation: explicit engine, Parma polyhedra library, value iteration
  - many applications
- But are we doing the right thing?
  - i.e. is the modelling abstraction satisfactory for the problem at hand?
- Beyond games…

Playing games with Google car…

– http://theoatmeal.com/blog/google_self_driving_car

Need cognitive models!

Humans are pretty good at guessing what others on the road will do. Driverless cars are not—and that can be exploited.

770 234

By Samuel English Anthony

- Games will not suffice – need to make mobile autonomy more like humans, with goals and intentions…

Conclusion

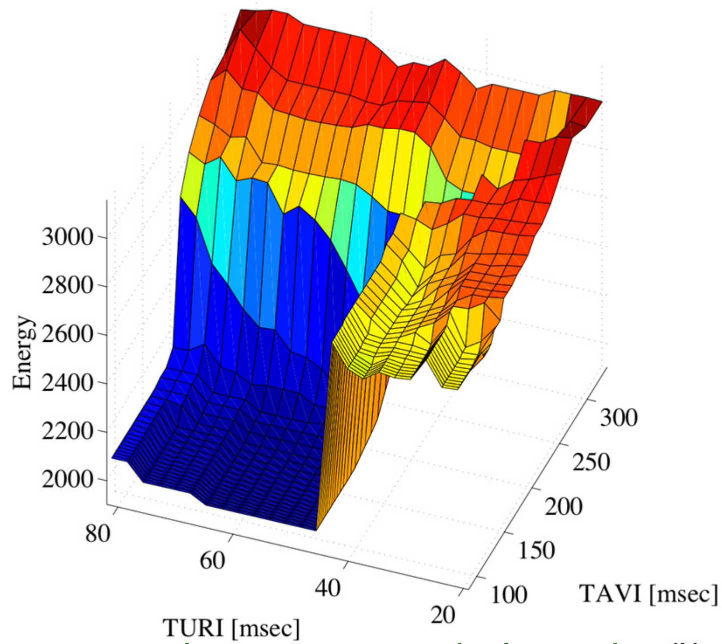
- Appraisal
  - some progress towards model checking and strategy synthesis for autonomous scenarios based on the games abstraction
  - high complexity, performance sluggish
  - model still too limited, particularly regarding trust
    - e.g. Tesla crash example of ‘overtrust’

- Future work
  - mobility?
  - consider social aspects?
  - allow partial observability?
  - combine with Nash equilibria?

- Beyond games…

Personalised wearable/implantable devices

- Hybrid model-based framework
  - timed automata model for pacemaker software
  - hybrid heart models in Simulink, adopt synthetic ECG model (non-linear ODE)
- 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

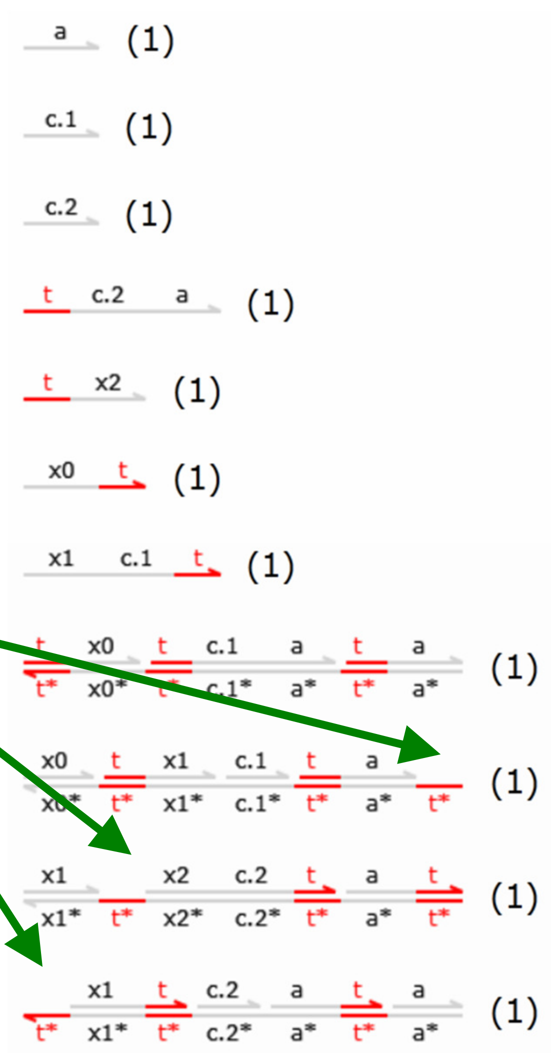


Synthesising robust and optimal parameters for cardiac pacemakers using symbolic and evolutionary computation techniques. Kwiatkowska, Mereacre, Paoletti and Patane, HSB'16

DNA computation

- Cardelli's DNA transducer gate
  - inputs/outputs single strands
  - two transducers connected
- PRISM identifies a bug: 5-step trace to a "bad" deadlock state
  - previously found manually [Cardelli'10]
  - detection now fully automated
- Bug is easily fixed
  - (and verified)

reactive gates

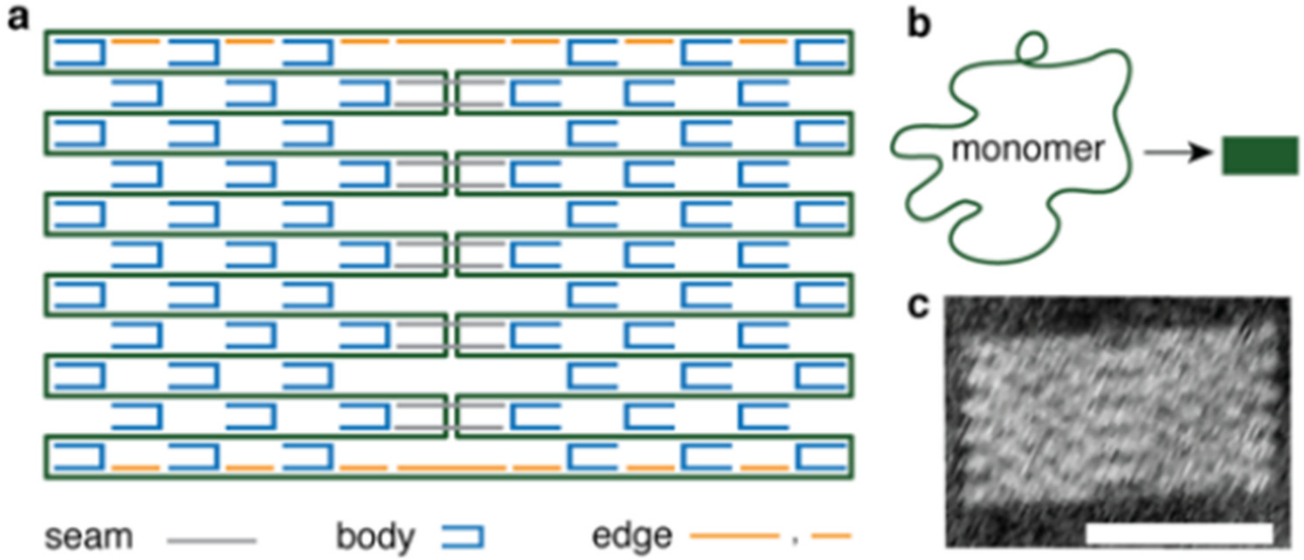


Counterexample:
(1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
(0,1,1,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
(0,0,1,0,1,1,1,1,1,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
(0,0,1,0,1,1,1,1,0,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
(0,0,1,0,1,1,0,1,0,0,1,1,1,0,0,0,1,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0)
(0,0,1,0,1,1,0,1,0,0,1,0,1,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0)

Design and Analysis of DNA Strand Displacement Devices using Probabilistic Model Checking, Lakin *et al*, Journal of the Royal Society Interface, 9(72), 1470-1485, 2012

DNA origami tiles

- DNA origami tiles: molecular breadboard [Turberfield lab]



50nm

Aim to understand how to control the folding pathways

- formulate an abstract Markov chain model
- obtain model predictions using Gillespie simulation
- perform a range of experiments, consistent with preditions

Guiding the folding pathway of DNA origami. Dunne, Dannenberg, Ouldridge, Kwiatkowska, Turberfield & Bath, Nature 525, pages 82-86, 2015.

Acknowledgements

- My group and collaborators in this work
- Project funding
  - ERC Advanced Grant
  - EPSRC Mobile Autonomy Programme Grant
  - Oxford Martin School, Institute for the Future of Computing

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