



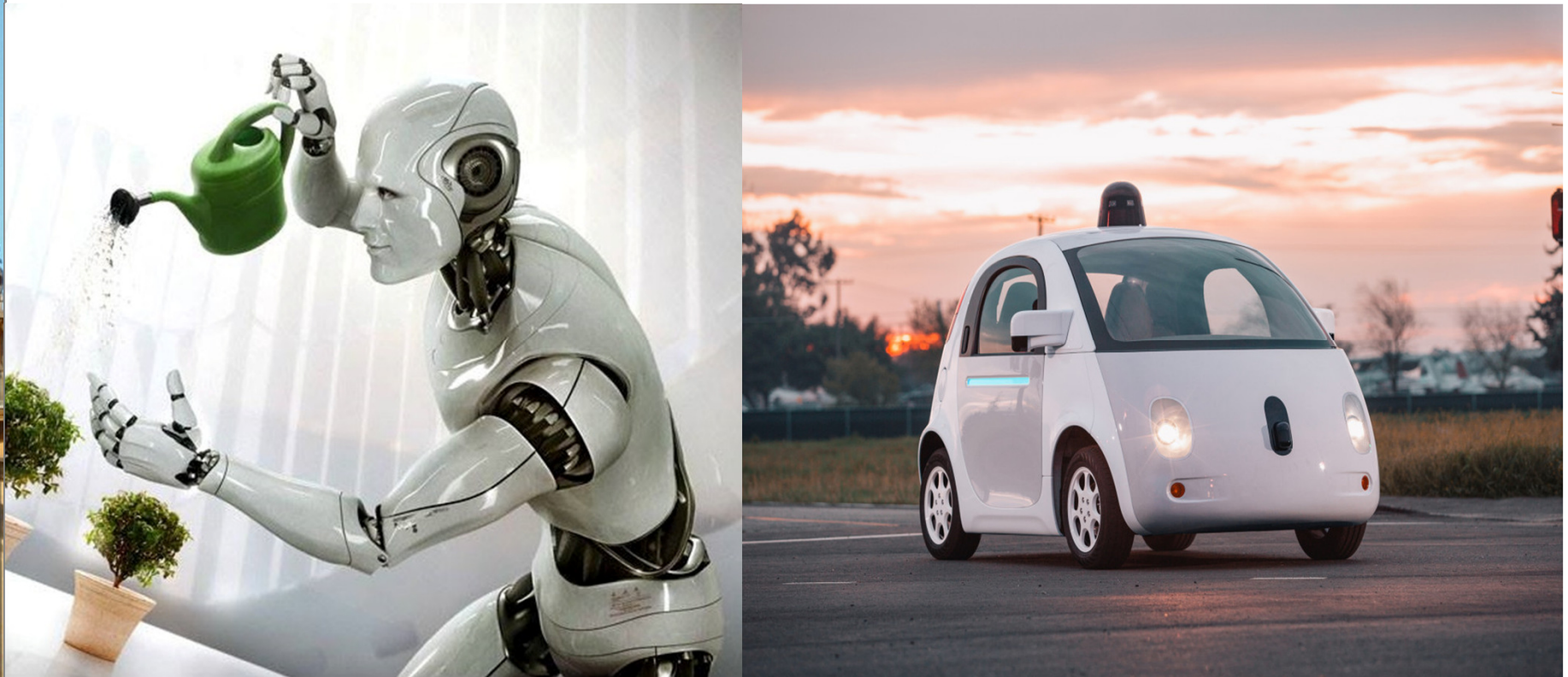
Model checking and strategy synthesis for mobile autonomy: from theory to practice

Marta Kwiatkowska

Department of Computer Science, University of Oxford

Uncertainty in Computation, Simons Institute, 4th October 2016


Mobile autonomy is here



Credits: That's Really Possible, Google

Are we safe?

- Embedded software at the heart of the device



THIS PROGRAM HAS PERFORMED AN ILLEGAL OPERATION
THIS PROGRAM WILL NOW CLOSE

- What if...
... something goes wrong in self-driving software?
- **Imagined or real?**

What happens when Tesla's AutoPilot goes wrong: owners post swerving videos

Early adopters share hair-raising videos of their Model S cars veering into traffic as new feature proves very beta



📷 Tesla's Model S may be one of the fastest, most advanced cars available, but reliability isn't quite up to scratch. Photograph: Christian Science Monitor/Getty

Software everywhere

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



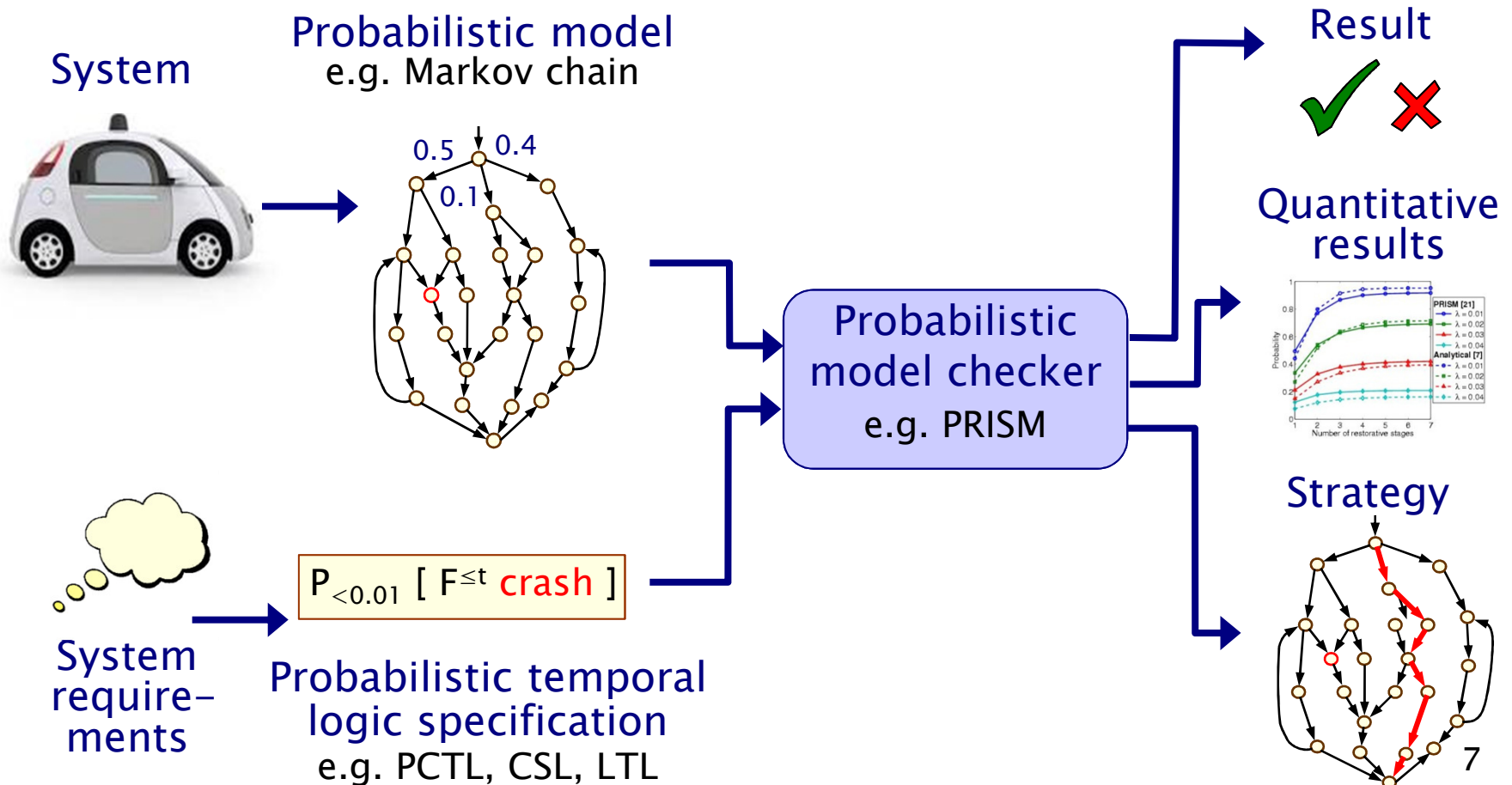
- **Quantitative properties**
 - safety, reliability, performance, efficiency, ...
 - “the probability of an airbag failing to deploy within 0.02s”
- **Quantitative verification to the rescue**
 - temporal logic specifications
 - formal verification

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, 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



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 the Google car...

“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...

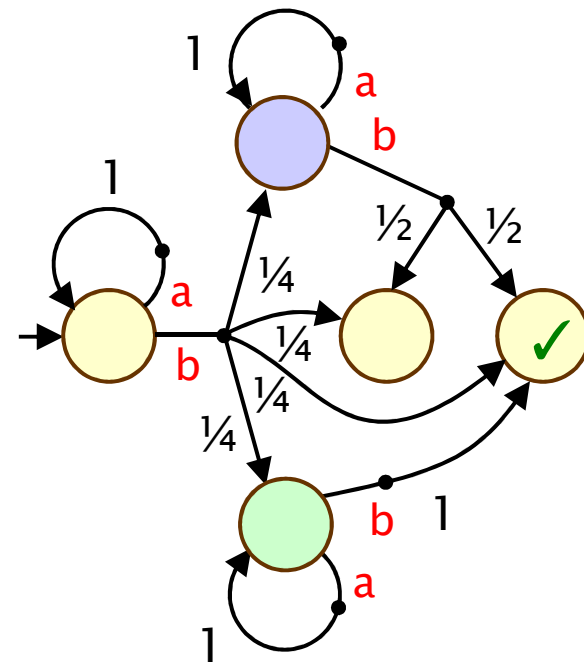
- Puts forward **stochastic multi-player games (SMGs)**
 - as an appropriate modelling abstraction for competitive behaviour, in adversarial environments
 - stochasticity to model e.g. failure, sensor uncertainty
- **Property specification: rPATL**
 - single-objective properties
 - verification
 - strategy synthesis
- **Extensions**
 - multi-objective properties, Pareto sets
 - compositional strategy synthesis
- **Tool support: PRISM-games 2.0**
- **Future challenges**

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
 - $\Delta : S \times A \rightarrow \text{Dist}(S)$ 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

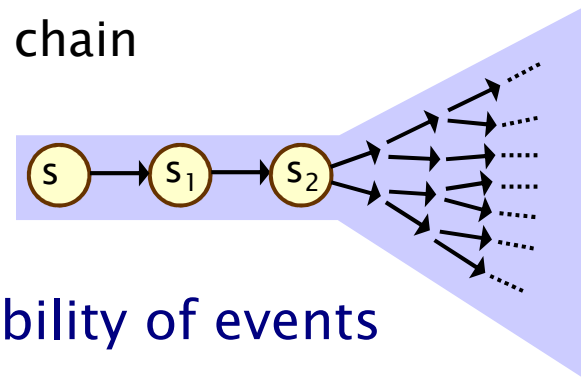
- Annotate SMGs with **rewards** (or costs)
 - real-valued quantities assigned to states and/or transitions
- Wide range of possible uses:
 - elapsed time, power consumption, number of messages successfully delivered, net profit, ...
- We work with:
 - state rewards: $r : S \rightarrow \mathbb{R}_{\geq 0}$
 - 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. $s_0 a_0 s_1 a_1 \dots$ 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 $\sigma_i : (SA)^* S_i \rightarrow \text{Dist}(A)$
 - Σ_i denotes the set of all strategies for player i
 - deterministic if σ_i always gives a Dirac distribution
 - memoryless if $\sigma_i(s_0 a_0 \dots s_k)$ depends only on s_k
 - also finite-memory, infinite memory, ...
 - history based or explicit memory representation
- A **strategy profile** is tuple $\sigma = (\sigma_1, \dots, \sigma_n)$
 - 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 $X : \text{Path}_s \rightarrow \mathbb{R}_{\geq 0}$
 - $E_s^\sigma [X] = \int_{\text{Path}_s} X \, d\Pr_s^\sigma$
 - used to define expected costs/rewards...

Property specification: rPATL

- Temporal logic **rPATL**:
 - reward **p**robabilistic **a**lternating **t**emporal **l**ogic
- 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:
 - $\langle\langle\{1,2\}\rangle\rangle P_{<0.01} [F^{\leq 10} \text{ error}]$
 - “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:

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

$$\psi ::= F a$$

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

“ratio”

“reachability”

“longrun average”

“cumulative”

- $\langle\langle C \rangle\rangle R^{\text{fuel}}_{< q} [C]$

- “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”

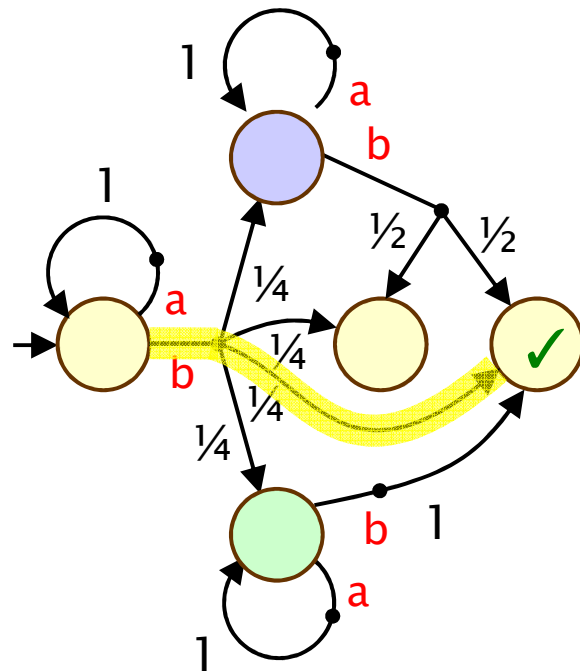
- $\langle\langle C \rangle\rangle R^{\text{fuel/time}}_{\leq q} [S]$

- “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 $\Pi \setminus C$ 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



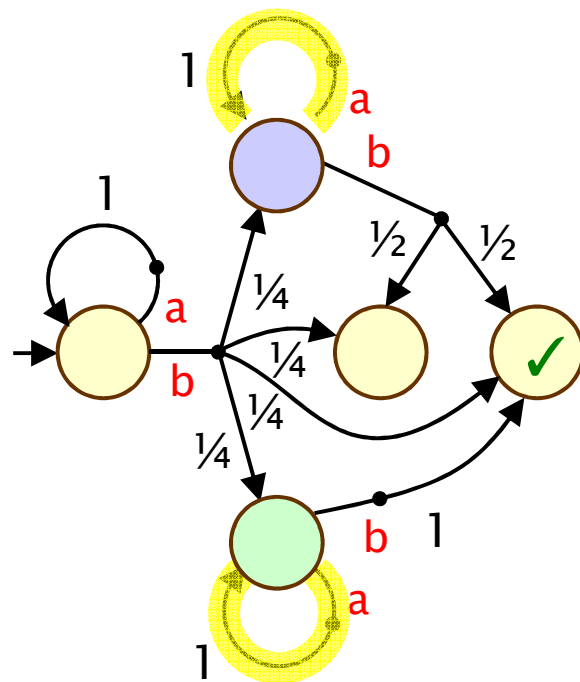
$$\langle\langle \bigcirc \rangle\rangle P_{\geq 1/4} [F \checkmark]$$

true in initial state

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

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

Examples

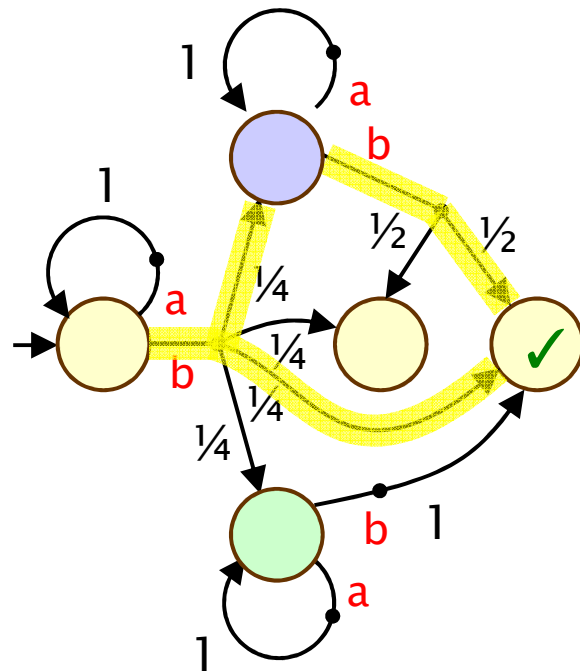


$\langle\langle \text{yellow} \rangle\rangle P_{\geq 1/4} [F \checkmark]$
true in initial state

$\langle\langle \text{yellow} \rangle\rangle P_{\geq 1/3} [F \checkmark]$
false in initial state

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

Examples



$\langle\langle \text{yellow} \rangle\rangle P_{\geq 1/4} [F \checkmark]$
true in initial state

$\langle\langle \text{yellow} \rangle\rangle P_{\geq 1/3} [F \checkmark]$
false in initial state

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

Value iteration

- E.g. $\langle\langle C \rangle\rangle P_{\geq q} [F \phi]$: max/min reachability probabilities
 - compute $\sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} \Pr_s^{\sigma_1, \sigma_2} (F \phi)$ for all states s
 - deterministic memoryless strategies suffice

- Value is:

- 1 if $s \in \text{Sat}(\phi)$, and otherwise **least** fixed point of:

$$f(s) = \begin{cases} \max_{a \in A(s)} \left(\sum_{s' \in S} \Delta(s, a)(s') \cdot f(s') \right) & \text{if } s \in S_1 \\ \min_{a \in A(s)} \left(\sum_{s' \in S} \Delta(s, a)(s') \cdot f(s') \right) & \text{if } s \in S_2 \end{cases}$$

- Computation:

- start from zero, propagate probabilities backwards
- guaranteed to converge

- Expected rewards: $\langle\langle C \rangle\rangle R_{\geq q}^r [F \phi]$ and $\langle\langle C \rangle\rangle R_{\geq q}^{r/c} [F \phi]$ are computed similarly adding the state-rewards at each step

Multi-objective properties

- May need to explore trade-offs, if conflicting objectives
 - 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 **v1** and **v2**, respectively ”

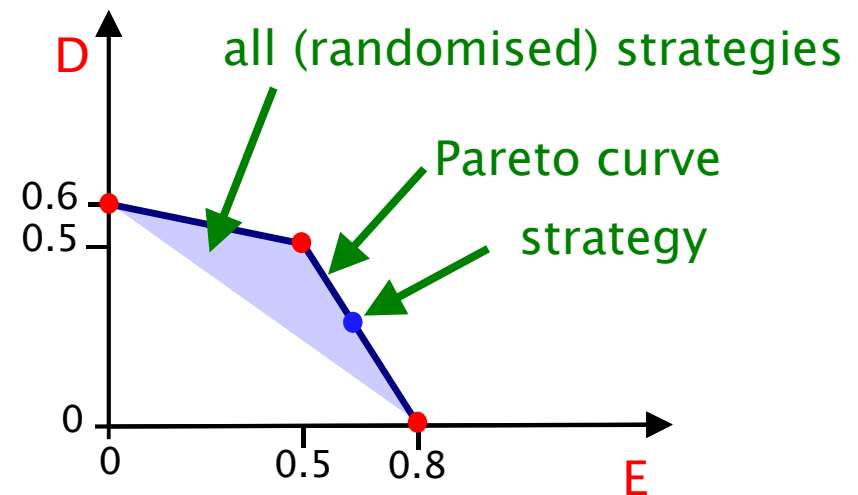
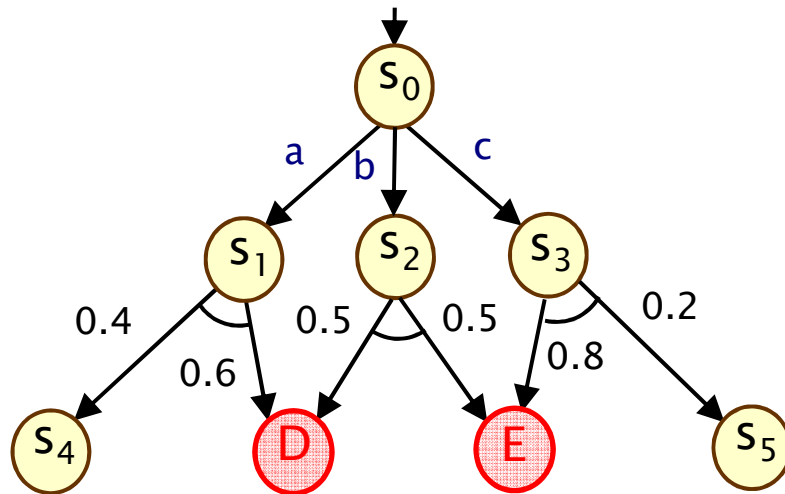
$$\langle\langle C \rangle\rangle (R^{\text{fuel}}_{\geq v1} [S] \ \& \ R^{\text{profit}}_{\geq v2} [S])$$

- NB Boolean combinations may be needed for implication

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

Example of Pareto optimality

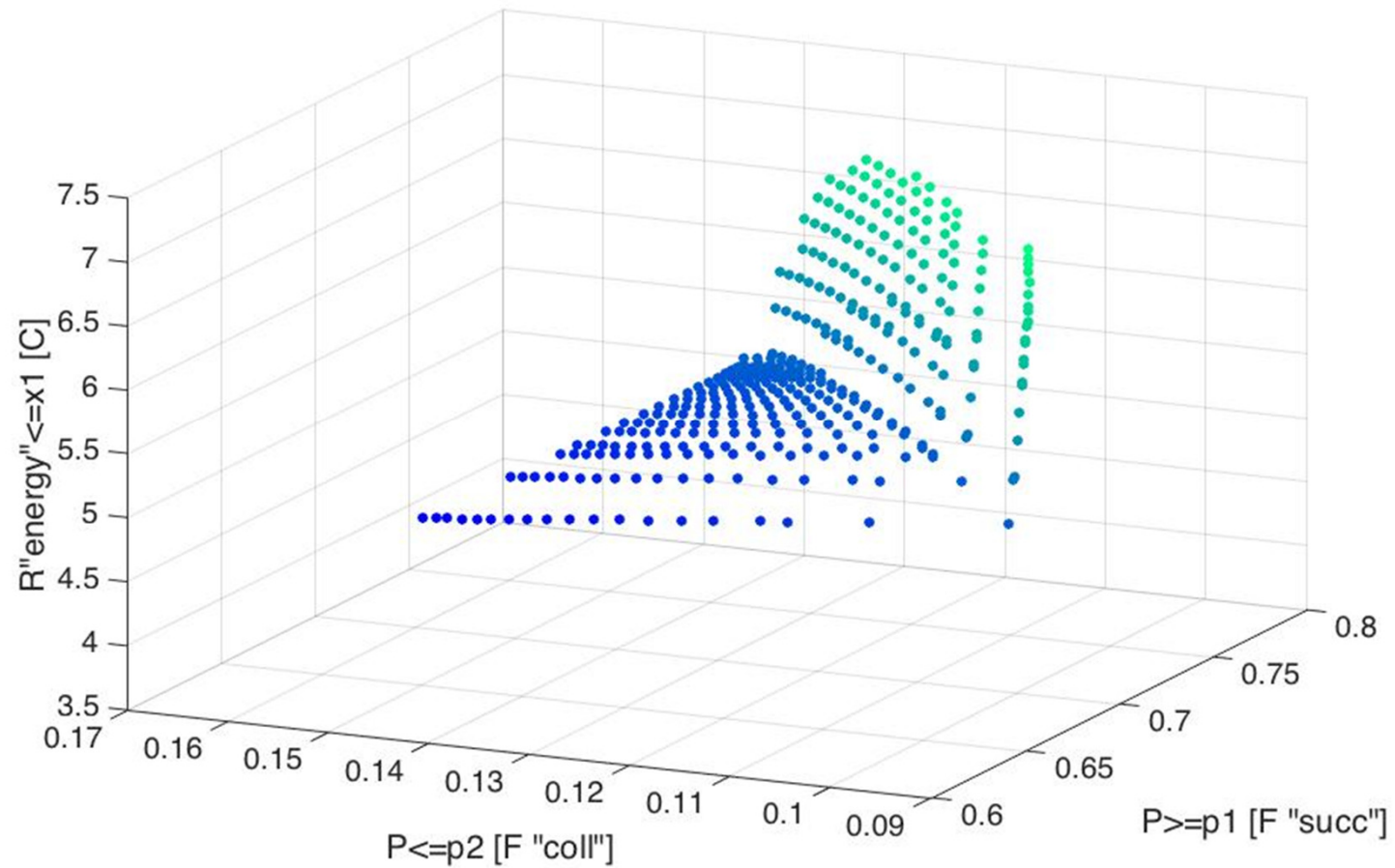
- Consider the simpler scenario of MDPs (1½ player games)
- Pareto optimum for conjunction of two objectives
 - probability of reaching **D** is greater than 0.2 and
 - probability of reaching **E** is greater than 0.6
- Randomised strategies may be needed...



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

Multidimensional Pareto set



Pareto set approximation for a mixed multi-objective property

Computation of Pareto sets

- Multi-objective strategy synthesis

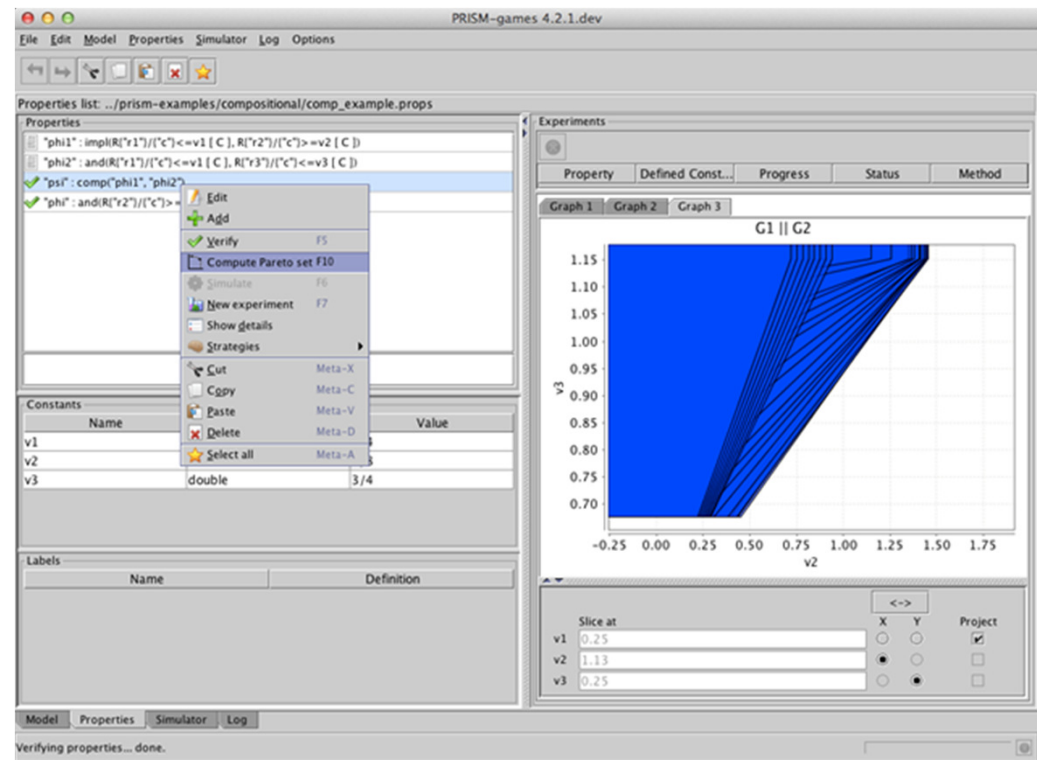
- value iteration over **polytopic** sets
- maintains a **vector** of such sets for each state, one for each dimension

- Pareto sets

- optimal achievable trade-offs between objectives

- Visualisation of high-dimensional Pareto sets

- projection
- slicing



Compositional strategy synthesis

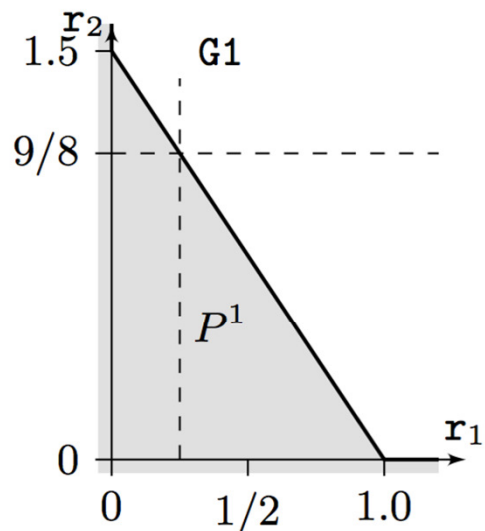
- **Componentised games**
 - improve scalability of analysis, avoid product state space
 - devise a composition operator for SMGs
- **Assume–guarantee synthesis:**
 - need a strategy for the full system satisfying a **global property**
 - synthesise one strategy per component, for **local properties**
 - 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

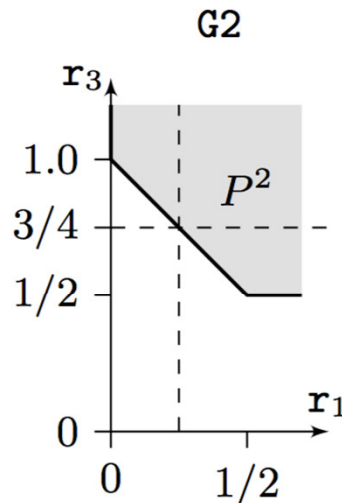
- Extension of rPATL: **Boolean** combinations of objectives
 - expected total rewards (for stopping games)
 - expected mean-payoffs or ratios (controllable multi-chain)
 - conjunctions of almost sure mean-payoffs/ratios (all games)
- **Example**
 - “**Player 1** can guarantee that, whenever the expected ratio of longrun average values for "**r1**" and "**c**" is at most **v1**, then the ratio for "**r2**" and "**c**" is at least **v2**
 - $\langle\langle 1 \rangle\rangle (R\{r1/c\} \leq v1 [S] \Rightarrow R\{r2/c\} \geq v2 [S])$
- **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

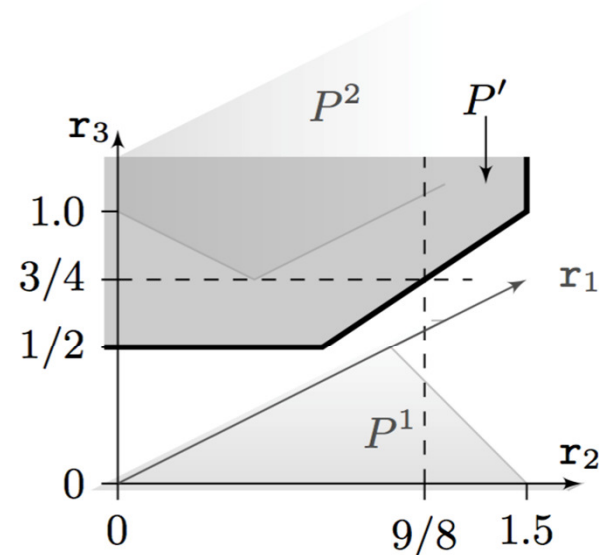
- Based on **assume-guarantee contracts** over component interfaces
- Synthesise **local** strategies for components, then compose into a **global** strategy using assume-guarantee rules
- Under-approximation of Pareto sets



$$\langle\langle 1 \rangle\rangle (\mathbb{R}\{r_1 / c\} \leq_{v_1} [S] \rightarrow \mathbb{R}\{r_2 / c\} \geq_{v_2} [S])$$



$$\langle\langle 1 \rangle\rangle (\mathbb{R}\{r_1 / c\} \leq_{v_1} [S] \wedge \mathbb{R}\{r_3 / c\} \leq_{v_3} [S])$$



$$\langle\langle 1 \rangle\rangle (\mathbb{R}\{r_2 / c\} \geq_{v_2} [S] \wedge \mathbb{R}\{r_3 / c\} \leq_{v_3} [S])$$

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*

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 games sufficient?
 - i.e. is the modelling abstraction satisfactory for the problem at hand?
 - can quantitative verification help?

It's not that simple...

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

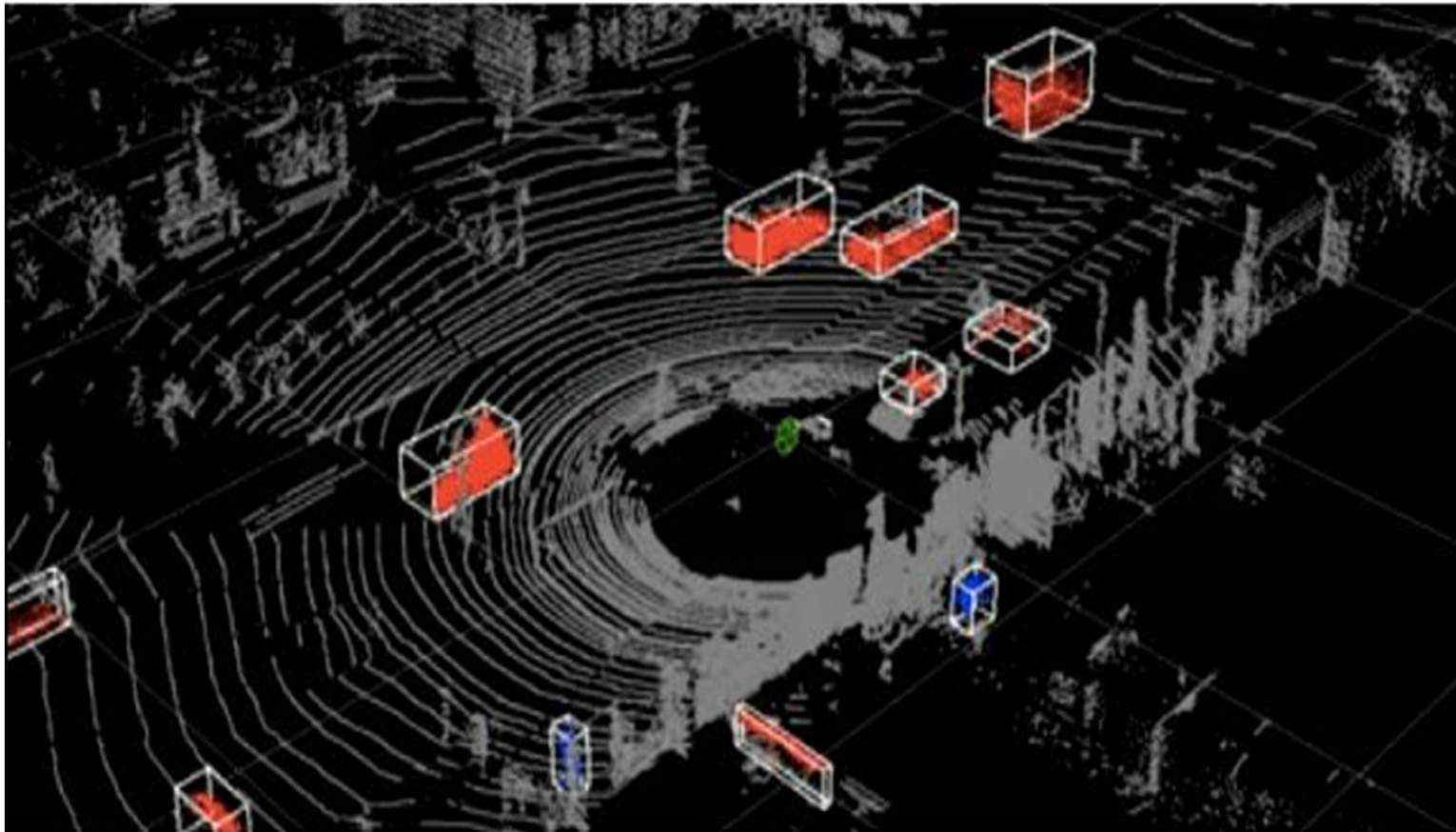


By Samuel English Anthony



- **Games will not suffice** – need multi-modal communication, cognitive reasoning, social norms, trust, ethics, ...

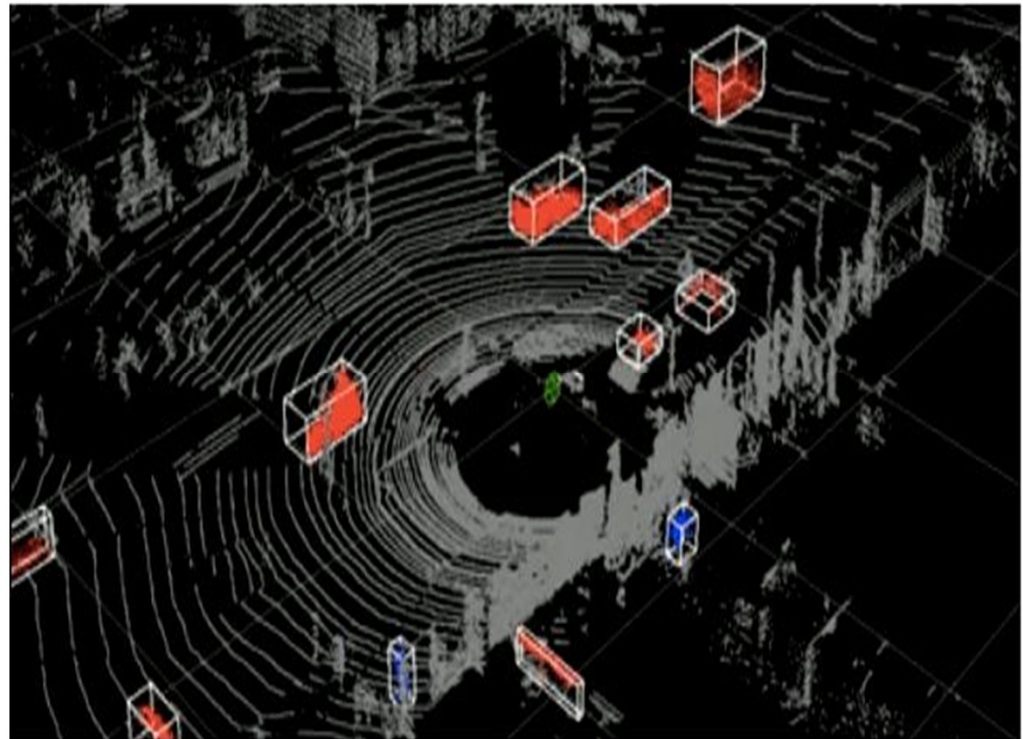
Perception software



Credits: Oxford Robotics Institute

Things that can go wrong...

- ...in **perception** software
 - sensor failure
 - object detection failure
- **Machine learning** software
 - not clear how it works
 - does **not** offer guarantees
- Yet end-to-end solutions are being considered...



Conclusion

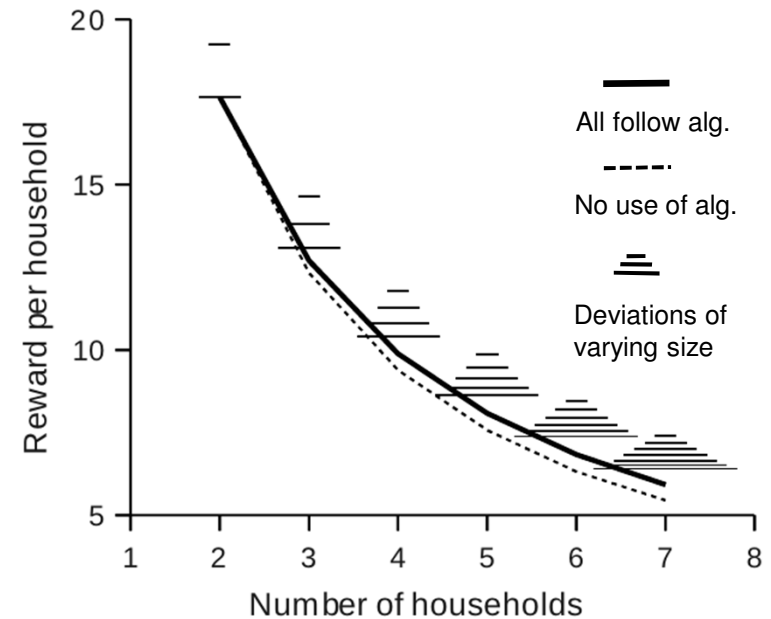
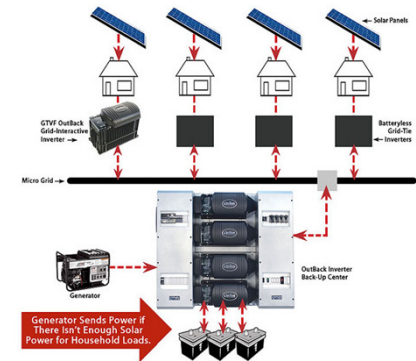
- Demonstrated progress towards ensuring **safety** and **efficiency** for mobile autonomy based on the games abstraction, with tool support
- But many challenges remain
 - how to handle **partial** observability?
 - can we verify **perception** software?
 - combine with **Nash equilibria**?
 - integrate with **mechanism design**?
- Need social aspects and ethics
 - extend with **cognitive** reasoning?
 - model human-like **trust**?
- Can we develop quantitative verification and synthesis?

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

Case study: Energy management

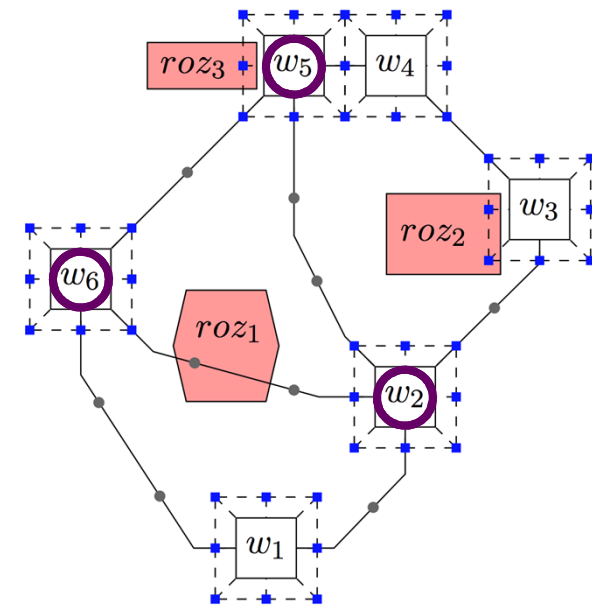
- Energy management protocol for Microgrid
 - Microgrid: local energy management
 - randomised demand management protocol [Hildmann/Saffre'11]
 - probability: randomisation, demand model, ...
- Existing analysis
 - simulation-based
 - assumes all clients are unselfish
- Our analysis
 - stochastic multi-player 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* 45

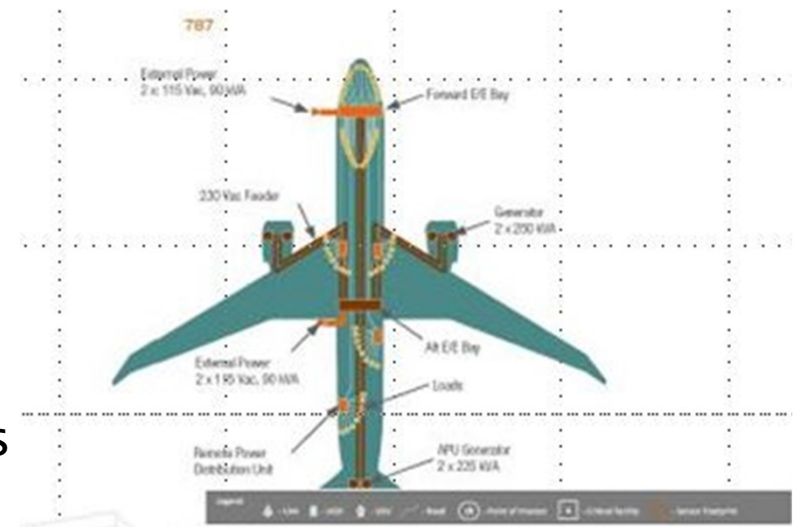
Case study: UAV path planning

- **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



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

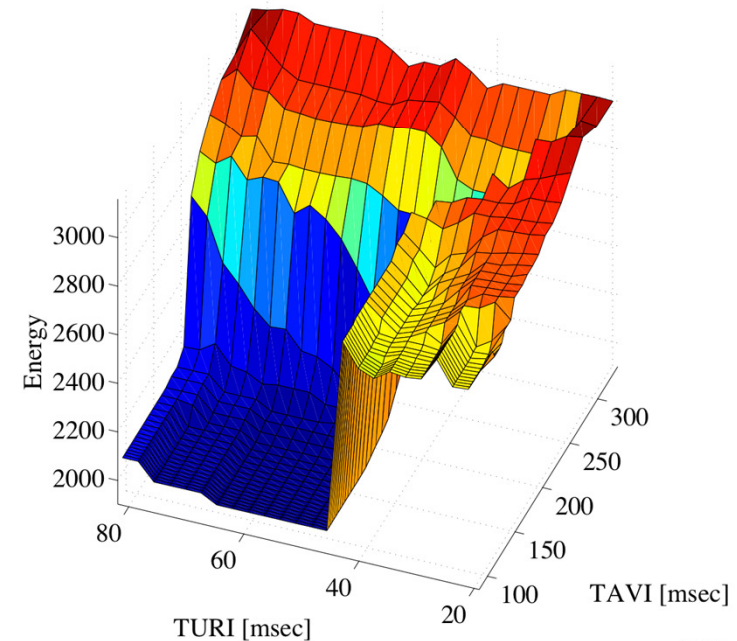


Personalised wearable/implantable devices

- 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



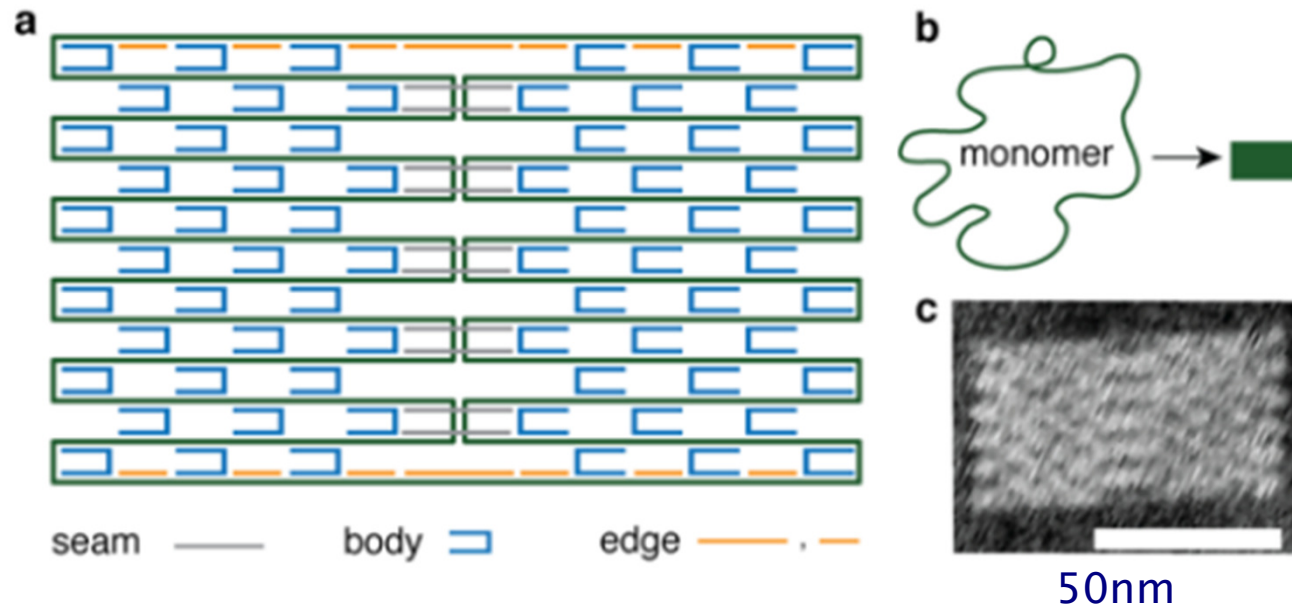
Copyright ©2008 Boston Scientific Corporation All rights reserved.



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

DNA origami tiles

- DNA origami tiles: **molecular breadboard** [Turberfield lab]



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 predictions

[Guiding the folding pathway of DNA origami.](#) Dunne, Dannenberg, Ouldrige, Kwiatkowska^{5,1}
Turberfield & Bath, Nature 525, pages 82–86, 2015.