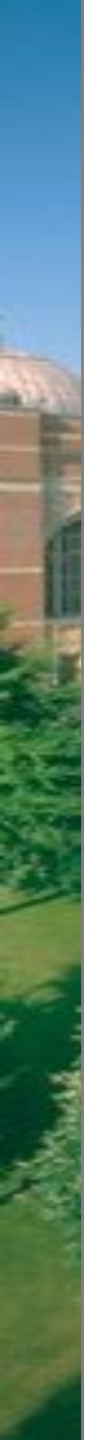


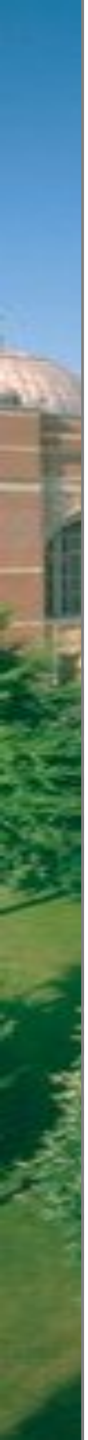
Multi-objective Reasoning with Probabilistic Model Checking

Dave Parker

University of Birmingham

2nd Intl. Workshop on Multi-objective Reasoning
in Verification and Synthesis (MoRe'19)

Vancouver, June 2019



Multi-objective Reasoning with Probabilistic Model Checking

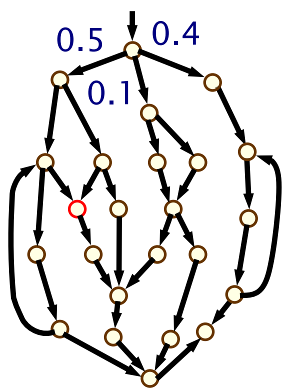
Dave Parker

University of Birmingham

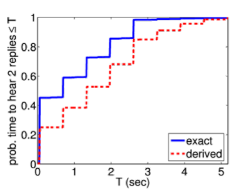
Joint work with:

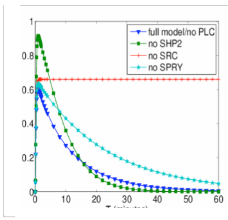
Gabriel Santos, Gethin Norman, Marta Kwiatkowska, …

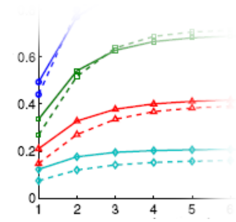
Probabilistic model checking

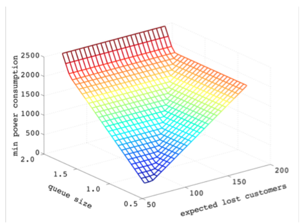


- Probabilistic model checking
  - formal construction/analysis of probabilistic models
  - "correctness" properties expressed in temporal logic
  - e.g. trigger → $P_{\geq 0.999}$ [$F^{\leq 20}$ deploy]
  - mix of exhaustive & numerical/quantitative reasoning









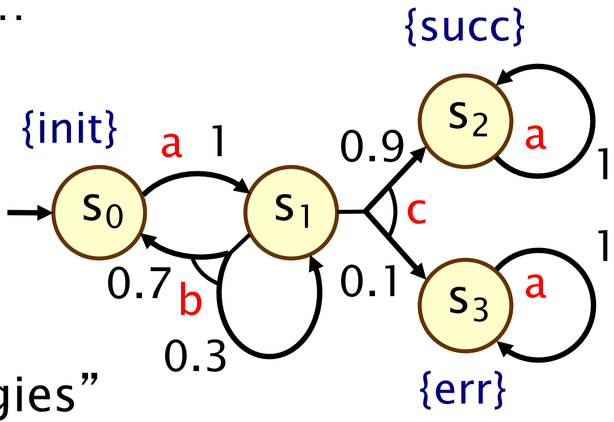
- Trends and advances
  - increasingly expressive/powerful model classes
  - from verification problems to control problems
  - ever widening range of application domains

Overview

- Multi-objective probabilistic model checking
  - Markov decision processes (MDPs)
    - examples: robot navigation, task scheduling
- Multiple players: competition/collaboration
  - rPATL model checking and strategy synthesis
  - stochastic multi-player games (SMGs)
    - example: energy management
  - concurrent stochastic games (CSGs)
    - example: investor models
- Multiple players and multiple objectives
  - (social welfare) Nash equilibria
    - example: communication protocols

Verification vs. Strategy synthesis

- Markov decision processes (MDPs)
  - models nondeterministic (actions, strategies) and probabilistic behaviour
  - strategies: randomisation, memory, …

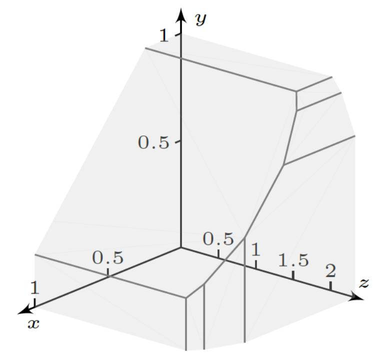


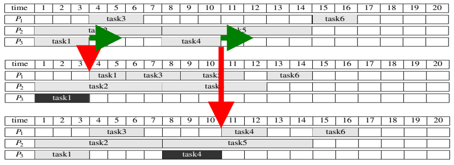
- 1. Verification
  - quantify over all possible strategies (i.e. best/worst-case)
  - $P_{\leq 0.1}$ [F *err*] : "the probability of an error occurring is $\leq$ 0.1 for all strategies"

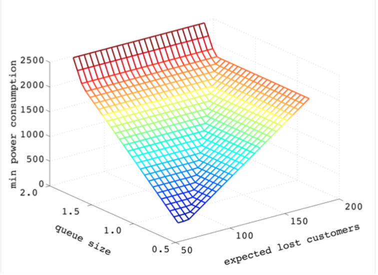
- 2. Strategy synthesis
  - generation of "correct-by-construction" controllers
  - $P_{\leq 0.1}$ [F *err*] : "does there exist a strategy for which the probability of an error occurring is $\leq$ 0.1?"

Strategy synthesis for MDPs

- Core property: probabilistic reachability
  - solvable with value iteration, policy iteration, linear programming, interval iteration, …
- Wide range of useful extensions
  - expected costs/rewards
  - linear temporal logic (LTL)
  - multi-objective model checking
  - real-time (PTAs)
  - partial observability (POMDPs)
- Applications
  - dynamic power management, robot navigation, UUV mission planning, task scheduling

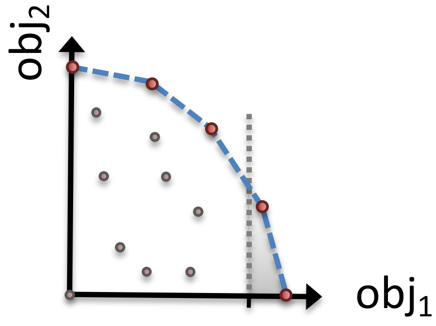






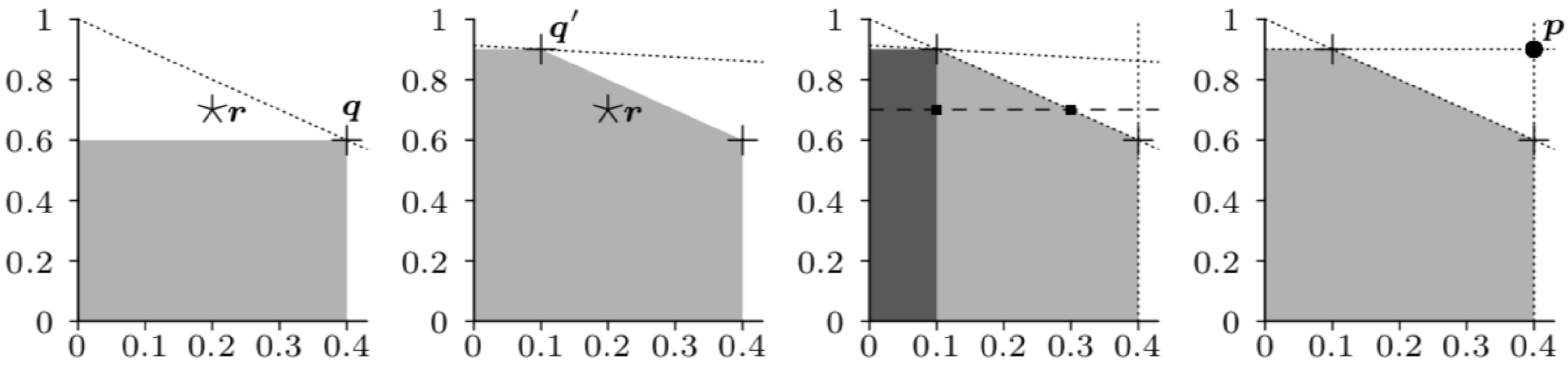
Multi-objective model checking

- Multi-objective probabilistic model checking
  - investigate trade-offs between conflicting objectives
  - in PRISM, objectives are probabilistic LTL or expected rewards
- Achievability queries: multi($P_{\geq 0.95}$ [F *send*], $R^{time}_{\geq 10}$ [C])
  - e.g. "is there a strategy such that the probability of message transmission is $\geq$ 0.95 and expected battery life $\geq$ 10 hrs?"
- Numerical queries: multi($P_{max=?}$ [F *send*], $R^{time}_{\geq 10}$ [C])
  - e.g. "maximum probability of message transmission, assuming expected battery life-time is $\geq$ 10 hrs?"
- Pareto queries:
  - multi($P_{max=?}$ [F *send*], $R^{time}_{max=?}$ [C])
  - e.g. "Pareto curve for maximising probability of transmission and expected battery life-time"



Multi-objective model checking

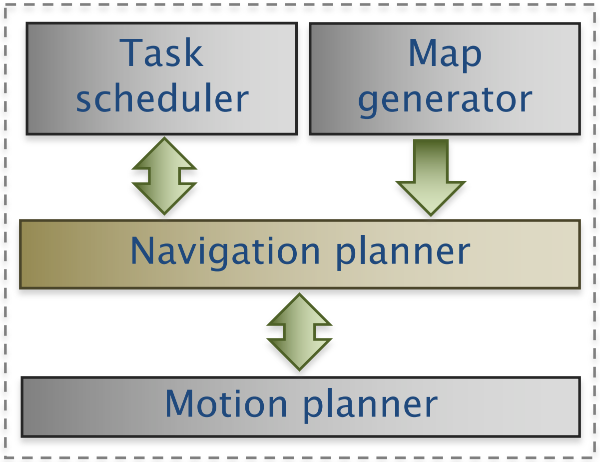
- PRISM implements two distinct approaches
- 1. Linear programming
  - solve dual problem to classical LP formulation
- 2. Value iteration based weighted sweep
  - approximate exploration/construction of Pareto curve
  - e.g. $P_{\geq r1}$ […] ∧ $P_{\geq r2}$ […] for $\mathbf{r}=(r_1,r_2)=(0.2,0.7)$

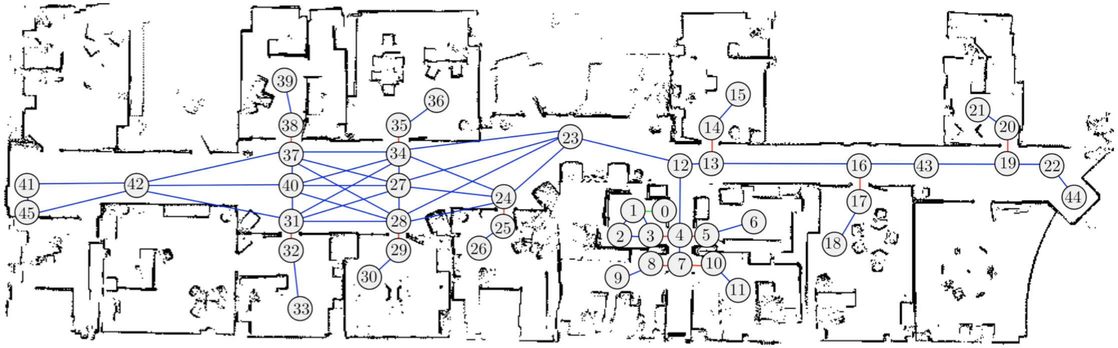


  - method 2 extends to step-bounded objectives

Application: Robot navigation

- Robot navigation planning: [IROS'14,IJCAI'15,ICAPS'17,IJRR'19]
  - learnt MDP models navigation through uncertain environment
  - co-safe LTL used to formally specify tasks to be executed by robot
  - finite-memory strategy synthesis to construct plans/controllers
  - ROS module based on PRISM
  - 100s of hrs of autonomous deployment





G4S Technology, Tewkesbury (STRANDS)

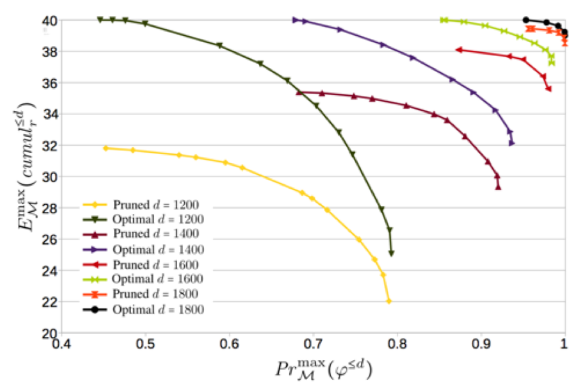
Multi-objective: Partial satisfiability

- Partially satisfiable task specifications
  - e.g. $P_{max=?}$ [$\neg zone_3$ U ($room_1 \wedge$ (F $room_4 \wedge$ F $room_5$)] < 1
- Synthesise strategies that, in decreasing order of priority:
  - maximise the probability of finishing the task;
  - maximise progress towards completion, if this is not possible;
  - minimise the expected time (or cost) required
- Progress function constructed from DFA
  - (distance to accepting states, reward for decreasing distance)
- Encode prioritisation using multi-objective queries:
  - $p = P_{max=?}$ [task]
  - r = multi($R^{prog}_{max=?}$ [C], $P_{>=p}$ [task])
  - multi($R^{time}_{min=?}$ [task], $P_{>=p}$ [task] $\wedge$ $R^{prog}_{>=r}$ [C])
- Or alternatively, using nested value iteration

Multi-obj: Time-bounded guarantees

- Often need probabilistic time-bounded guarantees
  - e.g. "probability of completing tasks within 5 mins is >0.99"
  - but verification techniques for these are less efficient/scalable
  - and often needed in conjunction with secondary objectives

- Efficient generation of time-bounded guarantees [ICAPS'17]
  - implemented in the PRISM model checker



- Key ideas:
  - optimize secondary goal wrt. guarantee
  - two phase verification: initial exploration of Pareto front on coarser untimed model
  - then generate guarantee from pruned model
  - significant gains in scalability

Overview

- Multi-objective probabilistic model checking
  - Markov decision processes (MDPs)
    - examples: robot navigation, task scheduling
- Multiple players: competition/collaboration
  - rPATL model checking and strategy synthesis
  - stochastic multi-player games (SMGs)
    - example: energy management
  - concurrent stochastic games (CSGs)
    - example: investor models
- Multiple players and multiple objectives
  - (social welfare) Nash equilibria
    - example: communication protocols

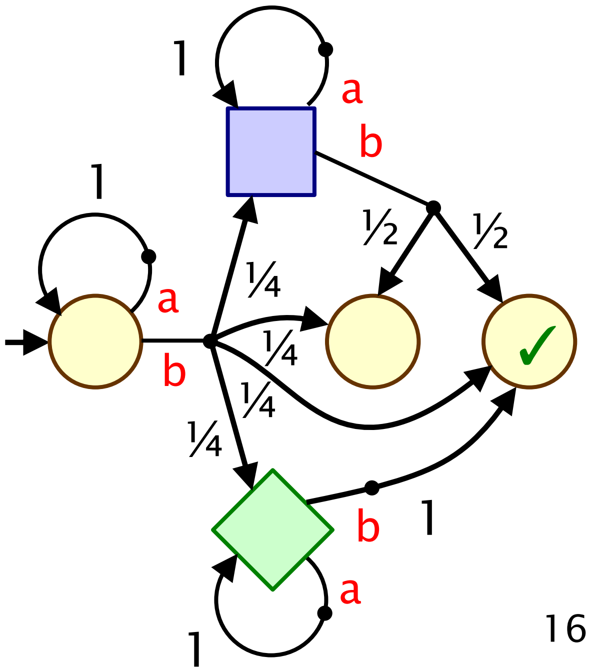
Competitive/collaborative behaviour

- Open systems
  - multiple system components, not all under our control
  - possibly with differing/opposing goals
  - giving rise to competitive/collaborative behaviour
- Many occurrences in practice
  - e.g. security protocols, algorithms for distributed consensus, energy management or sensor network co-ordination
- Natural to adopt a game-theoretic view
  - here: stochastic multi-player games
  - key ingredients: temporal logic, probabilistic model checking, tool support (PRISM-games), case studies

Stochastic multi-player games

- Stochastic multi-player game (SMGs)
  - nondeterminism + probability + multiple players
  - for now: turn-based (players control states)
  - applications: e.g. security (system vs. attacker), controller synthesis (controller vs. environment)

- A (turn-based) SMG is a tuple $(N, S, \langle S_i \rangle_{i \in N}, A, \delta, L)$ where:
  - N is a set of n players
  - S is a (finite) set of states
  - $\langle S_i \rangle_{i \in N}$ is a partition of S
  - A is a set of action labels
  - $\delta : S \times A \to Dist(S)$ is a (partial) transition probability function
  - $L : S \to 2^{AP}$ is a labelling function



Strategies, probabilities & rewards

- Strategy for player i: resolves choices in S_i states
  - based on execution history, i.e. $\sigma_i : (SA)^*S_i \rightarrow Dist(A)$
  - can be: deterministic (pure), randomised, memoryless, finite-memory, …
  - Σ_i denotes the set of all strategies for player i

- Strategy profile: strategies for all players: $\sigma=(\sigma_1,\ldots,\sigma_n)$
  - induces a set of (infinite) paths from some start state s
  - a probability measure Pr_s^σ over these paths
  - expectation $E_s^\sigma(X)$ of random variable X over Pr_s^σ

- Rewards (or costs)
  - non-negative values assigned to states/transitions
  - e.g. elapsed time, energy consumption, number of packets lost, net profit, …

Property specification: rPATL

- rPATL (reward probabilistic alternating temporal logic)
  - branching-time temporal logic for SMGs
- CTL, extended with:
  - coalition operator ⟨⟨C⟩⟩ of ATL
  - probabilistic operator P of PCTL
  - generalised (expected) reward operator R from PRISM
- In short:
  - zero-sum, probabilistic reachability + expected (total) reward
- Example:
  - $\langle\langle\{1,3\}\rangle\rangle \, P_{<0.01} [\, F^{\leq 10} \, \mathrm{error} \,]$
  - "players 1 and 3 have a strategy to ensure that the probability of an error occurring within 10 steps is less than 0.01, regardless of the strategies of other players"

rPATL syntax/semantics

- Syntax:

$$\phi ::= \text{true} \mid a \mid \neg\phi \mid \phi \wedge \phi \mid \langle\langle C\rangle\rangle P_{\bowtie q}[\psi] \mid \langle\langle C\rangle\rangle R^{r}_{\bowtie x}\,[\rho]$$

$$\psi ::= X\,\phi \mid \phi\ U^{\leq k}\ \phi \mid \phi\ U\ \phi$$

$$\rho ::= I^{=k} \mid C^{\leq k} \mid F\,\phi$$

- where:
  - $a \in AP$ is an atomic proposition, $C \subseteq N$ is a coalition of players, $\bowtie\ \in \{\leq,<,>,\geq\}$, $q \in [0,1] \cap \mathbb{Q}$, $x \in \mathbb{Q}_{\geq 0}$, $k \in \mathbb{N}$
    r is a reward structure

- Semantics:
- e.g. P operator: $s \vDash \langle\langle C\rangle\rangle P_{\bowtie q}[\psi]$ iff:
  - "there exist strategies for players in coalition C such that, for all strategies of the other players, the probability of path formula ψ being true from state s satisfies ⋈ q"

rPATL and beyond

- Generalised reward operators [TACAS'12, FMSD'13]
  - $\langle\langle C \rangle\rangle R^{r}_{\bowtie x}$ [$F^{\star}\phi$] where $\star \in \{\infty, c, 0\}$
  - F^0 is tricky: needs finite-memory strategies
- Quantitative (numerical) properties:
  - $\langle\langle \{1\} \rangle\rangle\, P_{max=?}$ [F error], i.e. $\sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} Pr_s^{\sigma_1,\sigma_2}$ (F error)
  - "what is the maximum probability of reaching an error state that player 1 can guarantee?" (against player 2)
- Nesting (and n>2 players)
  - players: $sensor_1$, $sensor_2$, repairer
  - $\langle\langle sensor_1 \rangle\rangle\, P_{<0.01}$[F ($\neg\langle\langle repairer \rangle\rangle\, P_{\geq 0.95}$[F "operational"])]
- And more…
  - rPATL*, reward-bounded [FMSD], exact bounds [CONCUR'12]
  - multi-objective model checking [QEST'13,TACAS15,I&C'17]

rPATL model checking for SMGs

- Reduces to solving zero-sum stochastic 2-player games
  - complexity: NP $\cap$ coNP (without any $R[F^0]$ operators)
  - complexity for full logic: NEXP $\cap$ coNEXP (due to $R[F^0]$)
- In practice, we use value iteration (numerical fixed points)
  - and more: graph-algorithms, sequences of fixed points, …
- E.g. probabilistic reachability: $\langle\langle C \rangle\rangle\, P_{\geq q}[\, F\, \phi\,]$
  - 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 $p(s)$ for state s is least fixed point of:

$$p(s) = \begin{cases} 1 & \text{if } s \in Sat(\phi) \\ \max_{a \in A(s)} \Sigma_{s' \in S}\, \delta(s,a)(s') \cdot p(s') & \text{if } s \in S_1 \setminus Sat(\phi) \\ \min_{a \in A(s)} \Sigma_{s' \in S}\, \delta(s,a)(s') \cdot p(s') & \text{if } s \in S_2 \setminus Sat(\phi) \end{cases}$$

  - convergence criteria need to be selected carefully

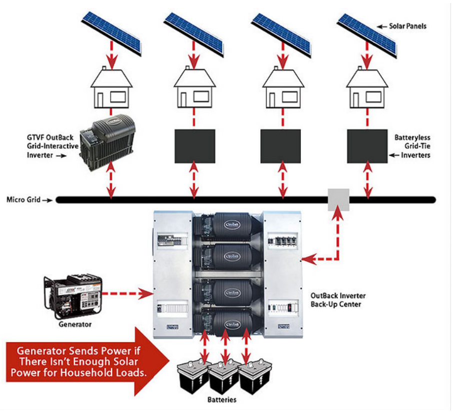
PRISM-games

- PRISM-games: www.prismmodelchecker.org/games
  - extension of PRISM modelling language (see later)
  - implementation in explicit engine
  - prototype MTBDD version also available

- Example application domains
  - security: attack-defence trees; DNS bandwidth amplification
  - self-adaptive software architectures
  - autonomous urban driving
  - human-in-the-loop UAV mission planning
  - collective decision making and team formation protocols
  - energy management protocols

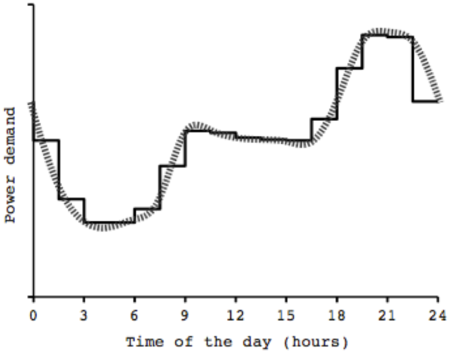
Application: Energy management

- Energy management protocol for Microgrid
  - randomised demand management protocol
  - random back-off when demand is high



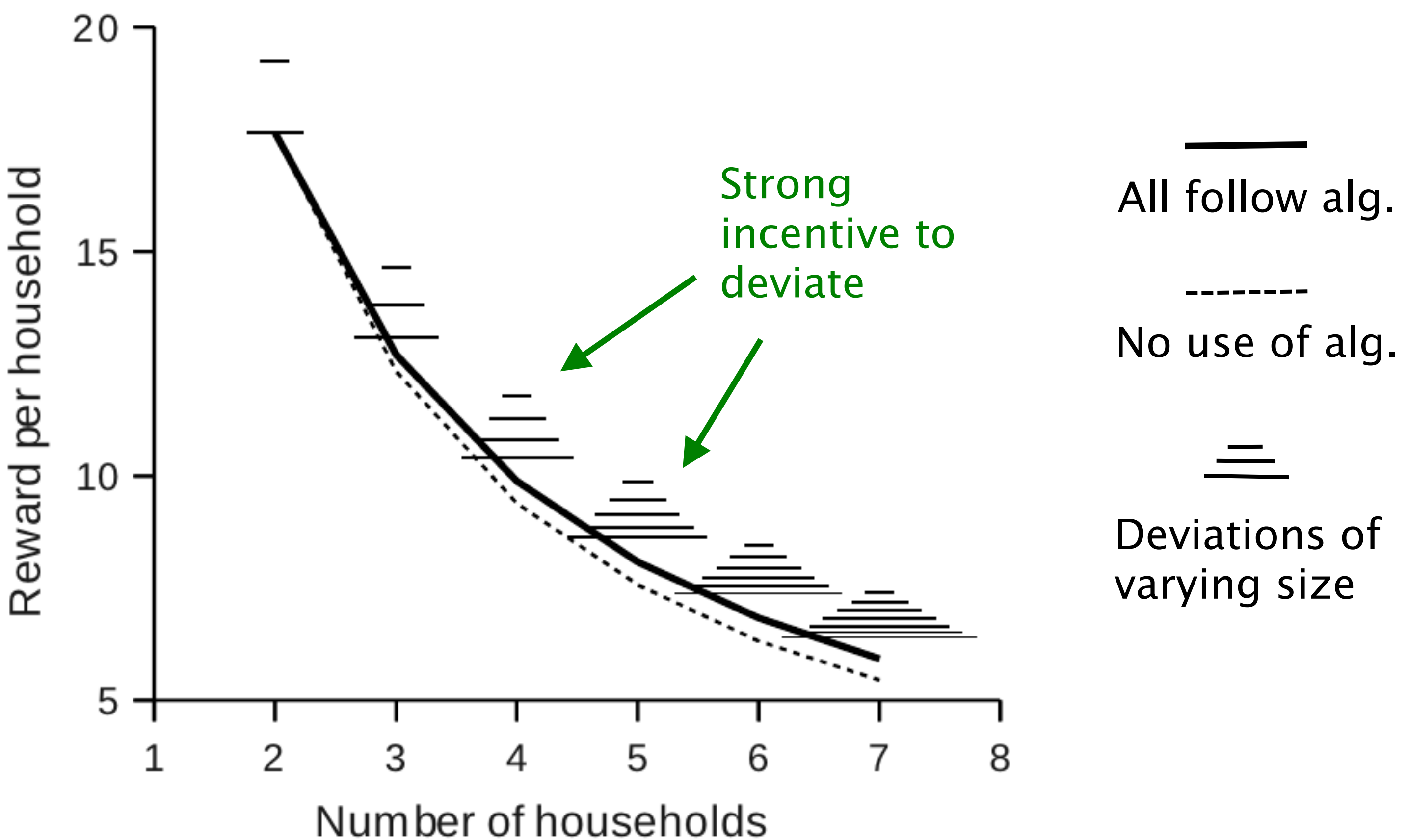
- Original analysis [Hildmann/Saffre'11]
  - protocol increases "value" for clients
  - simulation-based, clients are honest

- Our analysis
  - stochastic multi-player game model
  - clients can cheat (and cooperate)
  - model checking: PRISM-games
  - exposes protocol weakness (incentive for clients to act selfishly
  - propose/verify simple fix using penalties



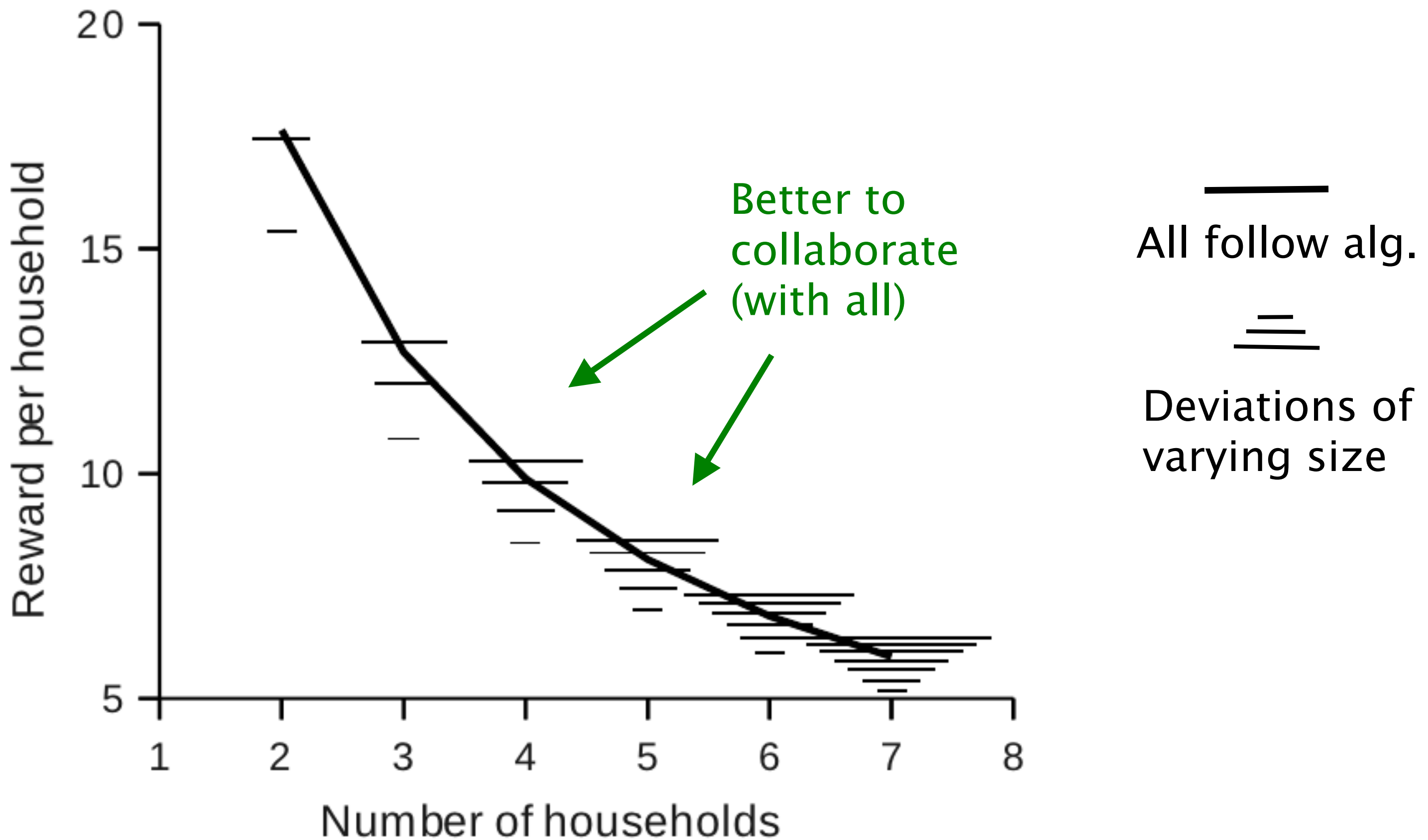
Results: Competitive behaviour

- Expected total value V per household
  - in rPATL: $\langle\langle C\rangle\rangle R^{r_C}_{max=?}\,[F^0\ time=max\ time]\ /\ |C|$
  - where r_C is combined rewards for coalition C



Results: Competitive behaviour

- Algorithm fix: simple punishment mechanism
  - distribution manager can cancel some loads exceeding c_{lim}



Overview

- Multi-objective probabilistic model checking
  - Markov decision processes (MDPs)
    - examples: robot navigation, task scheduling
- Multiple players: competition/collaboration
  - rPATL model checking and strategy synthesis
  - stochastic multi-player games (SMGs)
    - example: energy management
  - concurrent stochastic games (CSGs)
    - example: investor models
- Multiple players and multiple objectives
  - (social welfare) Nash equilibria
    - example: communication protocols

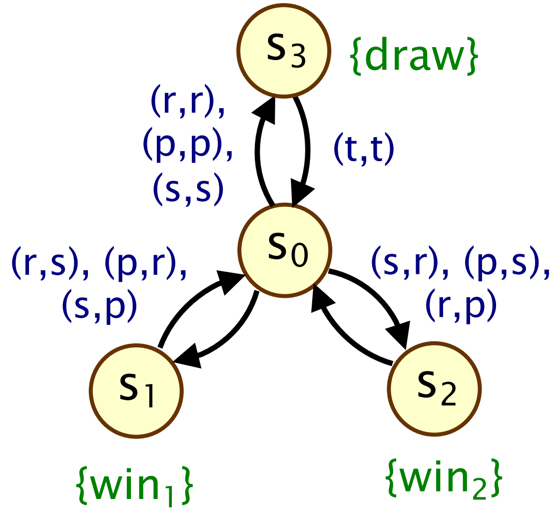
Concurrent stochastic games

- Concurrent stochastic games (CSGs)
  - players choose actions concurrently
  - jointly determines (probabilistic) successor state
  - generalises turn-based stochastic games
- Key motivation:
  - more realistic model of components operating concurrently, making action choices without knowledge of others
- Formally
  - set of n players N, state space S, actions A_i for player i
  - transition probability function $\delta : S \times A \rightarrow Dist(S)$
  - where $A = (A_1 \cup \{\bot\}) \times \ldots \times (A_n \cup \{\bot\})$
  - strategies $\sigma_i : FPath \rightarrow Dist(A_i)$, strategy profiles $\sigma=(\sigma_1,\ldots,\sigma_n)$
  - probability measure Pr_s^σ, expectations $E_s^\sigma(X)$

Example CSG: rock-paper-scissors

- Rock-paper-scissors game
  - 2 players repeated draw rock (r), paper (p), scissors (s), then restart the game (t)
  - rock > scissors, paper > rock, scissors > paper, otherwise draw

- Example CSG
  - 2 players: N={1,2}
  - $A_1 = A_2 = \{r,p,s,t\}$
  - NB: no probabilities here



Matrix games

- Matrix games
  - finite, one-shot, 2-player, zero-sum games
  - utility function $u_i : A_1 \times A_2 \rightarrow \mathbb{R}$ for each player i
  - represented by matrix Z where $z_{ij} = u_1(a_i,b_j) = -u_2(a_i,b_j)$
- Example:
  - one round of rock-paper-scissors

$$Z = \begin{array}{c} \\ r \\ p \\ s \end{array} \begin{array}{c} \begin{array}{ccc} r & p & s \end{array} \\ \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix} \end{array}$$

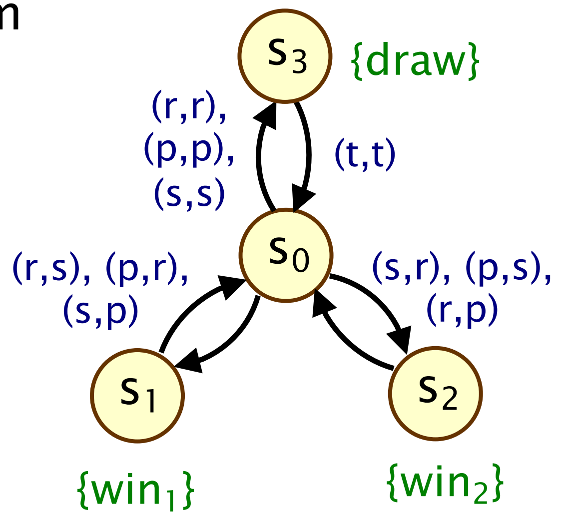
- Optimal (player 1) strategy via LP solution (minimax):
  - compute value val(Z): maximise value v subject to:
  - $v \leq x_p - x_s$
    $v \leq x_s - x_{r,}$
    $v \leq x_s - x_p$
    $x_r + x_p + x_s = 1$
    $x_r \geq 0,\ x_p \geq 0,\ x_s \geq 0$

Optimal strategy (randomised):
$(x_r,x_p,x_s) = (⅓,⅓,⅓)$

rPATL for CSGs

- We use the same logic rPATL as for SMGs

- Examples for rock-paper-scissors game:
  - $\langle\langle 1 \rangle\rangle\, P_{\geq 1}$ [F win_1] – player 1 can ensure it eventually wins a round of the game with probability 1
  - $\langle\langle 2 \rangle\rangle\, P_{max=?}$ [$\neg win_1$ U win_2] – the maximum probability with which player 2 can ensure it wins before player 1
  - $\langle\langle 1 \rangle\rangle\, R^{utility_1}_{max=?}$ [$C^{\leq 2K}$] – the maximum expected utility player 1 can ensure over K rounds (utility = 1/0/−1 for win/draw/lose)



rPATL model checking for CSGs

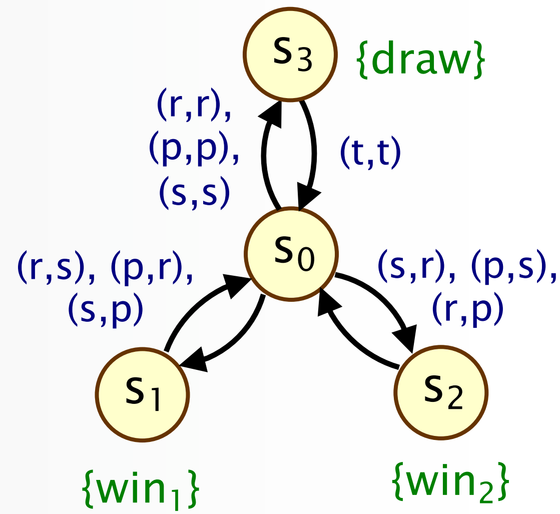
- Extends model checking algorithm for SMGs [QEST'18]
  - key ingredients are solution of (zero-sum) 2-player CSGs
- E.g. $\langle\langle C \rangle\rangle P_{\geq q}[\ F\ \phi\ ]$: max/min reachability probabilities
  - compute $\sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} Pr_s^{\sigma_1,\sigma_2}(F\ \phi)$ for all states s
  - note that optimal strategies are now randomised
  - solution of the 2-player CSG is in PSPACE
  - we use a value iteration based approach
- Value p(s) for state s is least fixed point of:
  - $p(s) = 1$ if $s \in Sat(\phi)$ and otherwise $p(s) = val(Z)$ where:
  - Z is the matrix game with $z_{ij} = \Sigma_{s' \in S}\ \delta(s,(a_i,b_j))(s') \cdot p(s')$
  - so each iteration requires solution of a matrix game for each state (LP problem of size |A|, where A = action set)

CSGs in PRISM-games

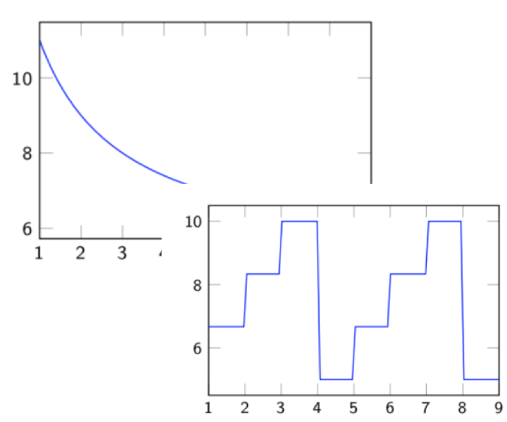
- CSG model checking implemented in PRISM-games
- Extension of PRISM modelling language
  - player specification via partition of modules
  - unlike SMGs, all modules move simultaneously
  - concurrent updates modelled with multi-action commands, e.g. [r1,r2] m1=0 → … and chained updates, e.g. (m2'=m1')
- Explicit engine implementation
  - plus LPsolve library for minimax LP solution
  - experiments with CSGs up to ~3 million states
- Case studies:
  - future markets investor, trust models for user-centric networks, intrusion detection policies, jamming radio systems

CSGs in PRISM (rock–paper–scissors)

```
csg

player player1 M1 endplayer
player player2 M2 endplayer

module M1
    m1 : [0..3];
    [r1] m1=0 → (m1'=1); // rock
    [p1] m1=0 → (m1'=2); // paper
    [s1] m1=0 → (m1'=3); // scissors
    [t1] m1>0 → (m1'=0); // restart
endmodule
module M2 = M1 [ m1=m2, r1=r2 , p1=p2, s1=s2, t1=t2 ] endmodule

label "win1" = (m1=1&m2=3) | (m1=2&m2=1) | (m1=3&m2=2); // player 1 wins round
rewards "utility1" // utility for player 1
    [t1] (m1=1 & m2=3) | (m1=2 & m2=1) | (m1=3 & m2=2) : 1; // player 1 wins
    [t1] (m1=1 & m2=2) | (m1=2 & m2=3) | (m1=3 & m2=1) : -1; // player 2 wins
endrewards
```



Application: Future markets investor



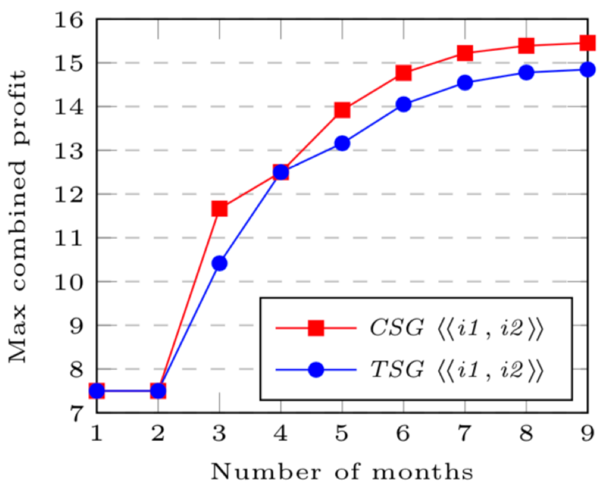
- Model of interactions between:
  - stock market, evolves stochastically
  - two investors i_1, i_2 decide when to invest
  - market decides whether to bar investors

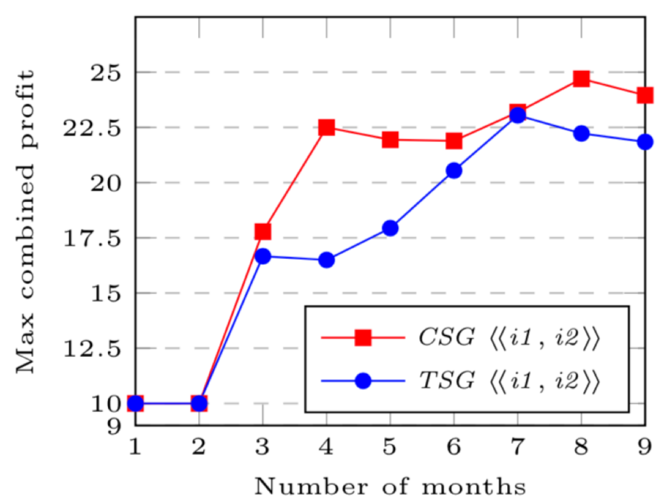
- Modelled as a 3-player CSG
  - extends simpler model originally from [McIver/Morgan'07]
  - investing/barring decisions are simultaneous
  - profit reduced for simultaneous investments
  - market cannot observe investors' decisions

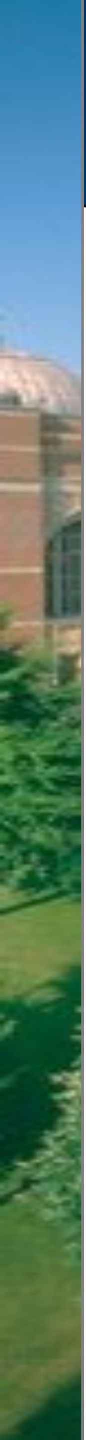
- Analysed with rPATL model checking & strategy synthesis
  - distinct profit models considered: 'normal market', 'later cash-ins' and 'later cash-ins with fluctuation'
  - comparison between SMG and CSG models

Application: Future markets investor

- Example rPATL queries:
  - $\langle\langle \text{investor}_1 \rangle\rangle \, R^{\text{profit}_1}_{\text{max}=?} [\, F \text{ finished}_1 \,]$
  - $\langle\langle \text{investor}_1, \text{investor}_2 \rangle\rangle \, R^{\text{profit}_{1,2}}_{\text{max}=?} [\, F \text{ finished}_{1,2} \,]$
  - i.e. maximising individual/joint profit
- Results (joint profit) – limited power of market shown
  - with (left) and without (right) fluctuations
  - optimal (randomised) investment strategies synthesised







Overview

- Multi-objective probabilistic model checking
  - Markov decision processes (MDPs)
    - examples: robot navigation, task scheduling
- Multiple players: competition/collaboration
  - rPATL model checking and strategy synthesis
  - stochastic multi-player games (SMGs)
    - example: energy management
  - concurrent stochastic games (CSGs)
    - example: investor models
- Multiple players and multiple objectives
  - (social welfare) Nash equilibria
    - example: communication protocols

Multiple objectives: Nash equilibria

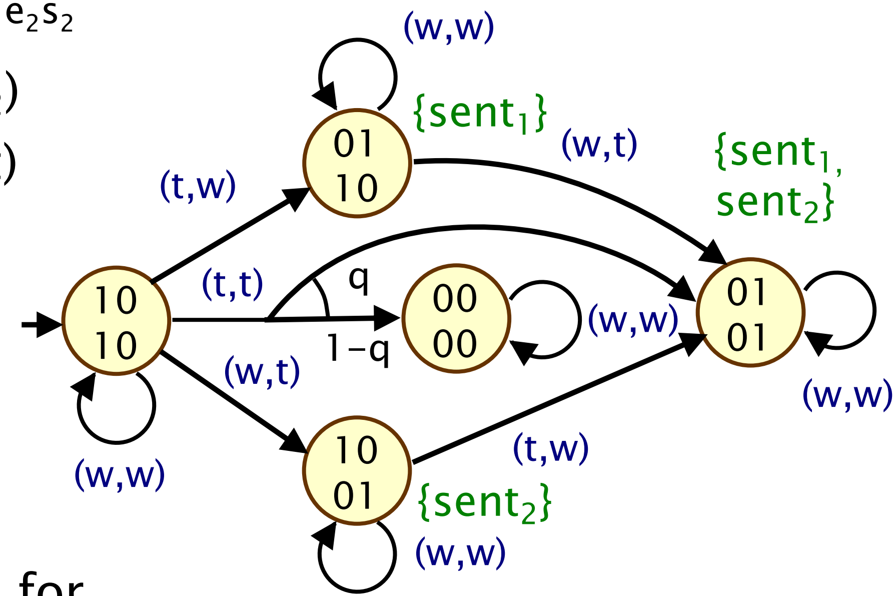
- Now consider distinct objectives X_i for each player i
  - i.e., no longer restricted to zero-sum goals

- We use Nash equilibria (NE)
  - no incentive for any player to unilaterally change strategy
  - more precisely subgame-perfect ε-Nash equilibrium
  - a strategy profile $\sigma=(\sigma_{1,\ldots,}\sigma_n)$ for a CSG is a subgame-perfect ε-Nash equilibrium for objectives $X_1,\ldots,X_n$ iff:
  - $E_s^{\sigma}(X_i) \geq \sup \{ E_s^{\sigma'}(X_i) \mid \sigma'=\sigma_{-i}[\sigma_i'] \text{ and } \sigma_i' \in \Sigma_i \} - \epsilon$ for all i, s
  - ε-NE (but not 0-NE) guaranteed to exist for CSGs

- In particular: social welfare Nash equilibria (SWNE)
  - NE which maximise sum $E_s^{\sigma}(X_1) + \ldots E_s^{\sigma}(X_n)$

Example

- CSG example: Medium access control protocol
  - 2 players (senders); states = $\begin{matrix} e_1 s_1 \\ e_2 s_2 \end{matrix}$ (energy$_1$/sent$_1$, energy$_2$/sent$_2$)
  - actions = t (transmit), w (wait)
  - q = probability of success if messages collide



(probabilistic extension of [Brenguier'13])

- If objectives X_i = probability to send successfully:
  - 2 SWNEs when one user waits for the other to transmit and then transmits

- If the objectives X_i =probability of being *first* to transmit their packet:
  - only 1 SWNE: both immediately try to transmit

rPATL + Nash operator

- Extension of rPATL for Nash equilibria [FM'19]

  $\phi ::= \text{true} \mid a \mid \neg\phi \mid \phi \wedge \phi \mid$
  $\langle\langle C\rangle\rangle P_{\bowtie q}[\psi] \mid \langle\langle C\rangle\rangle R^{r}_{\bowtie x}[\rho] \mid \langle\langle C,C'\rangle\rangle_{\max\bowtie x}[\theta]$

  $\theta ::= P[\psi]+P[\psi] \mid R^{r}[\rho]+R^{r}[\rho]$

  $\psi ::= X\ \phi \mid \phi\ U^{\leq k}\ \phi \mid \phi\ U\ \phi$

  $\rho ::= I^{=k} \mid C^{\leq k} \mid F\ \phi$

- where:
  - $a \in AP$ is an atomic proposition, $C \subseteq N$ is a coalition of players and $C'=N\backslash C$, $\bowtie \in \{\leq,<,>,\geq\}$, $q \in [0,1]\cap\mathbb{Q}$, $x \in \mathbb{Q}_{\geq 0}$, $k \in \mathbb{N}$

    r is a reward structure

- Semantics:
  - $\langle\langle C,C'\rangle\rangle_{\max\bowtie x}[\theta]$ is satisfied if there exist strategies for all players that form a SWNE between coalitions C and C'(=N\C), and under which the *sum* of the two objectives in θ is $\bowtie x$

Model checking for extended rPATL

- Key ingredient is now:
  - solution of SWNEs for bimatrix games
  - (basic problem is EXPTIME)
  - we adapt known approach using labelled polytopes, and implement using an encoding to SMT

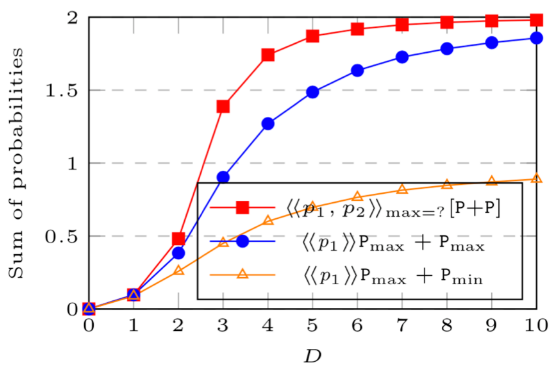
- Two types of model checking operator
  - bounded: backwards induction
  - unbounded: value iteration, e.g.:

$$
\mathtt{V}_{\mathsf{G}^C}(s,\theta,n) = \begin{cases} (1,1) & \text{if } s \in Sat(\phi^1) \cap Sat(\phi^2) \\ (1, \mathrm{P}^{\max}_{\mathsf{G},s}(\mathtt{F}\ \phi^2)) & \text{else if } s \in Sat(\phi^1) \\ (\mathrm{P}^{\max}_{\mathsf{G},s}(\mathtt{F}\ \phi^1), 1) & \text{else if } s \in Sat(\phi^2) \\ (0,0) & \text{else if } n{=}0 \\ val(\mathsf{Z}_1, \mathsf{Z}_2) & \text{otherwise} \end{cases}
$$

  - where Z_1 and Z_2 encode matrix games similar to before

PRISM-games support

- Implementation in PRISM-games
  - needed further extensions to modelling language
  - extends CSG rPATL model checking implementation
  - bimatrix games solved using Z3 encoding
  - optimised filtering of dominated strategies
  - scales up to CSGs with ~2 million states



- Applications
  - robot navigation in a grid, medium access control, Aloha communication protocol, power control
  - SWNE strategies outperform those found with rPATL
  - ϵ-Nash equilibria found typically have ϵ=0

Conclusions

- Probabilistic model checking: PRISM & PRISM-games
  - multi-objective techniques for MDPs
  - rPATL model checking for
    - stochastic multi-player games (SMGs)
    - concurrent stochastic games (CSGs)
  - CSGs + (social welfare) Nash equilibria
  - wide variety of case studies studied

- Challenges & directions
  - extending to >2 players
  - scalability, e.g. symbolic methods, abstraction
  - partial information/observability & greater efficiency
  - further applications and case studies