Probabilistic Verification of Concurrent Autonomous Systems

Dave Parker

University of Birmingham

EXPRESS-SOS, Aug 2021

Probabilistic Verification of Concurrent Autonomous Systems

Dave Parker

University of Birmingham

Joint work with:

Gabriel Santos, Gethin Norman, Marta Kwiatkowska, …

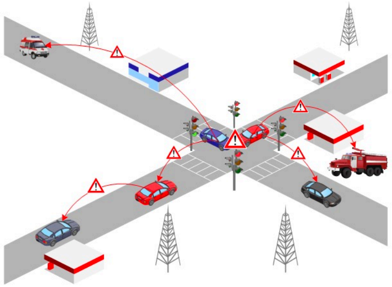
ERC Advanced Grant FUN2MODEL

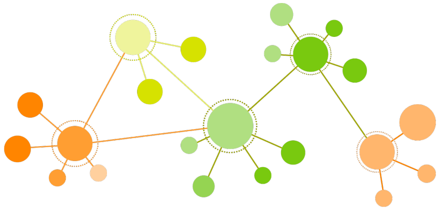
Verification of stochastic systems

- Formal verification needs stochastic modelling

faulty sensors/actuators

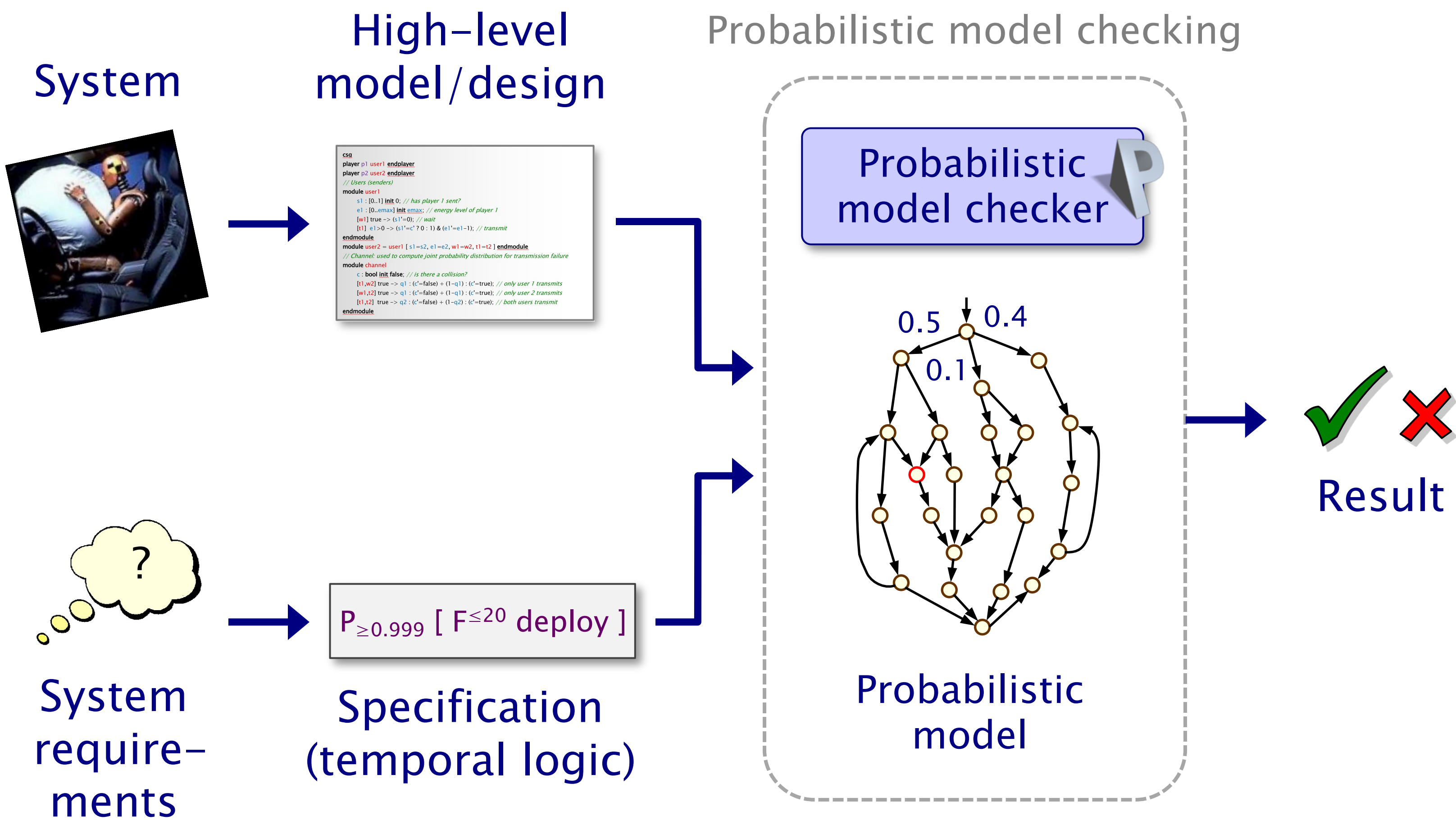
unpredictable/unknown environments





randomised protocols

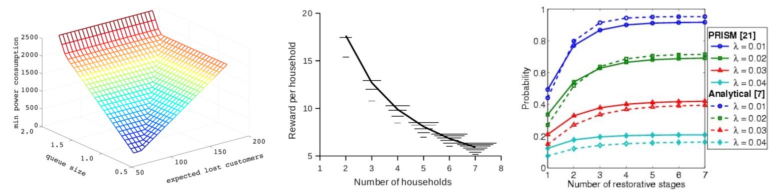
Probabilistic model checking



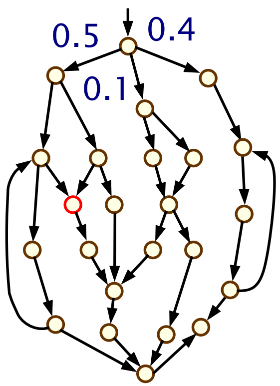
Probabilistic model checking

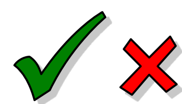
Probabilistic model checking

Numerical results/analysis



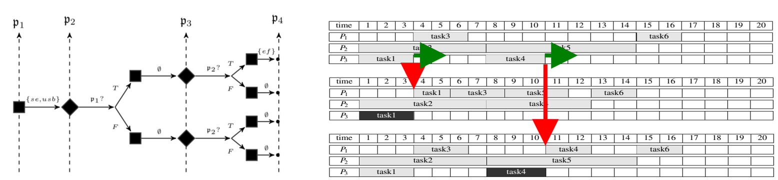
Probabilistic model checker





Result

Probabilistic model

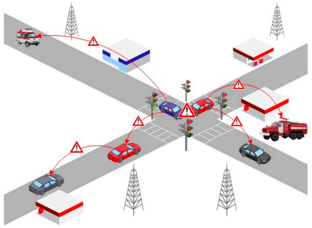
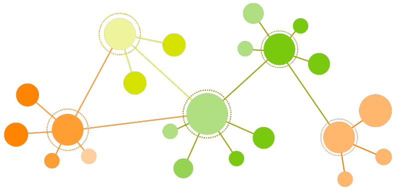


Strategies/policies/controllers

$P_{\geq 0.999}$ [$F^{\leq 20}$ deploy]

Verification with stochastic games

- How do we verify stochastic systems with…
  - multiple autonomous agents acting concurrently
  - competitive or collaborative behaviour between agents, possibly with differing/opposing goals
  - e.g. security protocols, algorithms for distributed consensus, energy management, autonomous robotics, auctions

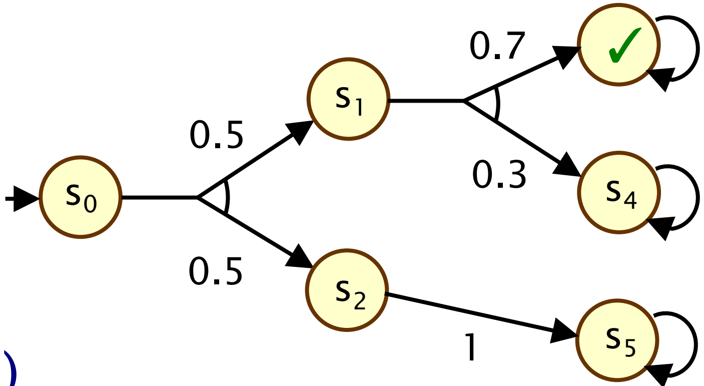
- This talk: verification with stochastic multi-player games
  - verification (and synthesis) of strategies that are robust in adversarial settings and stochastic environments
  - models, logics, algorithms, tools, examples

Overview

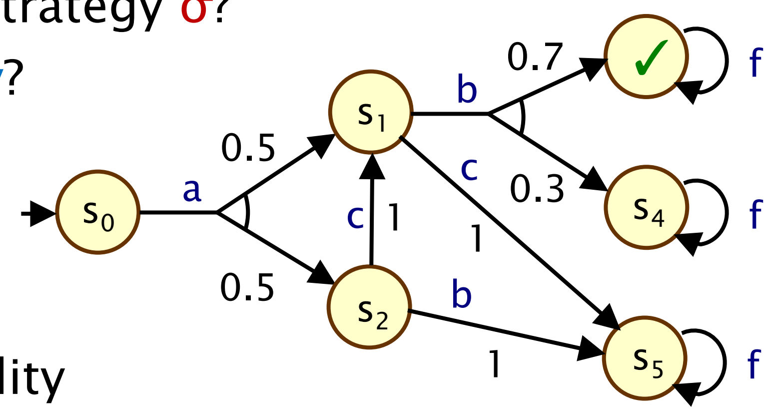
- Markov decision processes
- Stochastic multi-player games
- Concurrent stochastic games
- Equilibria-based properties

Probabilistic models

- Discrete-time Markov chains
  - e.g. what is the probability of reaching state ✓?



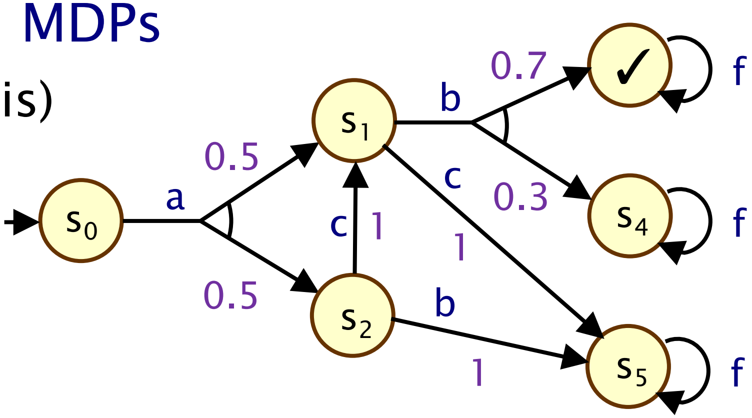
- Markov decision processes (MDPs)
  - strategies (or policies) resolve actions based on history
  - e.g. what is the maximum probability of reaching ✓ achievable by any strategy σ?
  - and what is an optimal strategy?



- Formally:
  - we write: $\sup_\sigma Pr_s^\sigma(F✓)$
  - where Pr_s^σ denotes the probability from state s under strategy σ

Solving MDPs

- Various techniques exist to solve MDPs
  - (and to perform strategy synthesis)



- Here, we focus on value iteration
  - dynamic programming approach
  - common for probabilistic model checking

transition probabilities:

δ : S x Act → Dist(S)

- For example:
  - maximum probability p(s) to reach ✓ from s
  - values p(s) are the least fixed point of:

$$p(s) = \begin{cases} 1 & \text{if } s \vDash \checkmark \\ \max_a \sum_{s'} \delta(s,a)(s') \cdot p(s') & \text{otherwise} \end{cases}$$

  - basis for iterative numerical computation

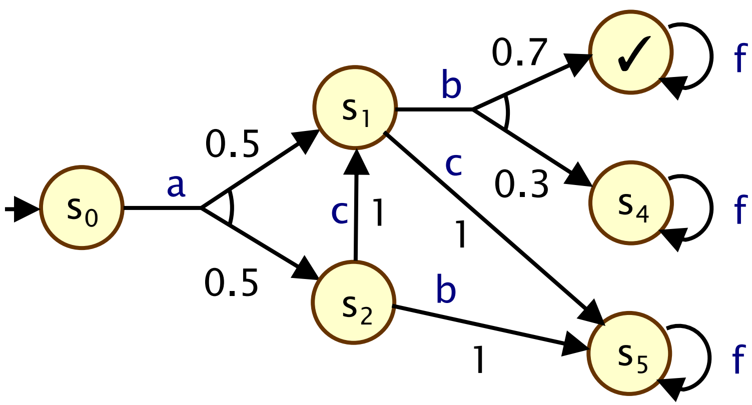
let p(s) = $\sup_\sigma Pr_s^\sigma(F\checkmark)$

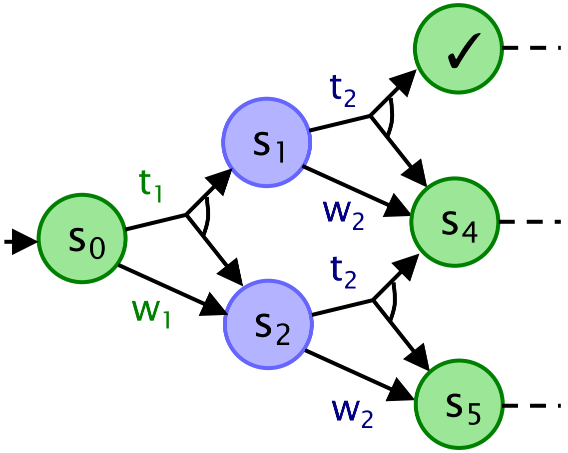
Stochastic games

Stochastic multi-player games

- Stochastic multi-player games
  - strategies + probability + multiple players
  - for now: turn-based (player i controls states S_i)

Markov decision processes (MDPs) ⇒ Turn-based stochastic games (TSGs)





Property specification: rPATL

- rPATL (reward probabilistic alternating temporal logic)
  - branching-time temporal logic for stochastic games
- CTL, extended with:
  - coalition operator $\langle\langle C \rangle\rangle$ of ATL
  - probabilistic operator P of PCTL
  - generalised (expected) reward operator R from PRISM
- In short:
  - zero-sum, probabilistic reachability + expected total reward
- Example:
  - $\langle\langle \{robot_1, robot_3\} \rangle\rangle \; P_{>0.99} \, [\, F^{\leq 10} \, (goal_1 \vee goal_3) \,]$
  - "robots 1 and 3 have a strategy to ensure that the probability of reaching the goal location within 10 steps is >0.99, regardless of the strategies of other players"

rPATL syntax/semantics

- Syntax:

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

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

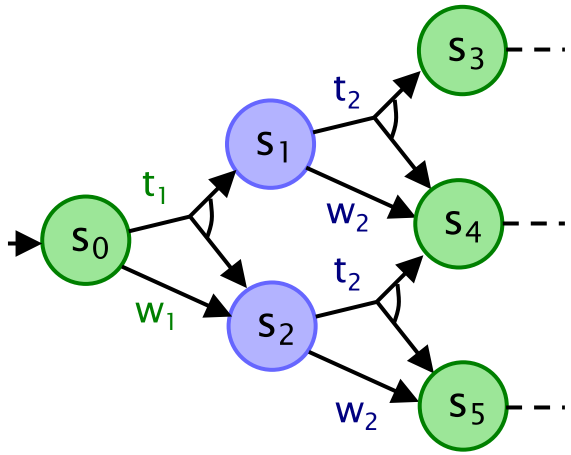
$$\rho ::= I^{=k} \mid C^{\leq k} \mid F\ \phi$$

- 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: $s \models \langle\langle C\rangle\rangle P_{\bowtie q}[\psi]$ 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"

Model checking rPATL

- Main task: checking individual P and R operators
  - reduces to solving a (zero-sum) stochastic 2-player game
  - e.g. max/min reachability probability: $\sup_{\sigma_1} \inf_{\sigma_2} \Pr_s^{\sigma_1,\sigma_2}(F\checkmark)$
  - complexity: NP ∩ coNP (if we omit some reward operators)



- We again use value iteration
  - values p(s) are the least fixed point of:

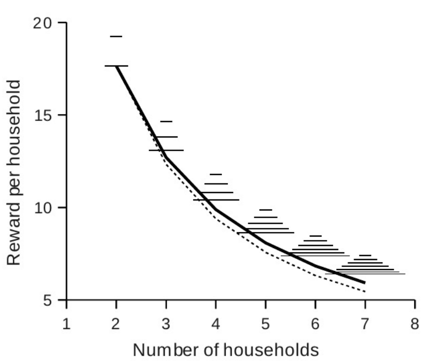
$$p(s) = \begin{cases} 1 & \text{if } s \models \checkmark \\ \max_a \Sigma_{s'} \delta(s,a)(s') \cdot p(s') & \text{if } s \not\models \checkmark \text{ and } s \in S_1 \\ \min_a \Sigma_{s'} \delta(s,a)(s') \cdot p(s') & \text{if } s \not\models \checkmark \text{ and } s \in S_2 \end{cases}$$

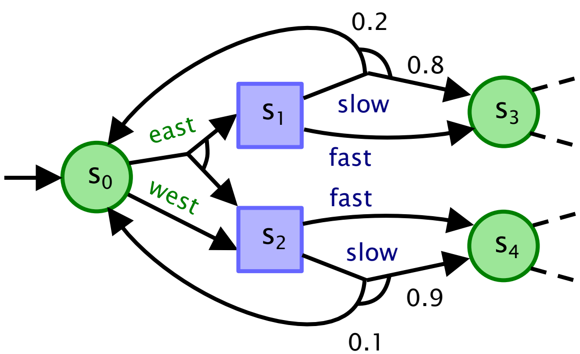
  - and more: graph-algorithms, sequences of fixed points, …

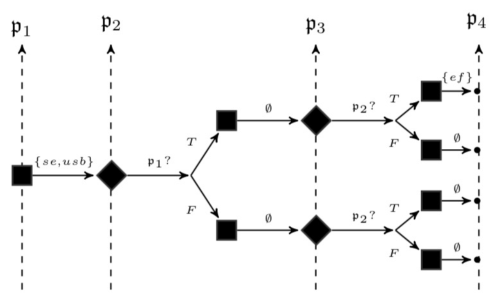
PRISM-games

- PRISM-games: prismmodelchecker.org/games
  - extension of PRISM modelling language
  - explicit state (and prototype symbolic) implementation

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







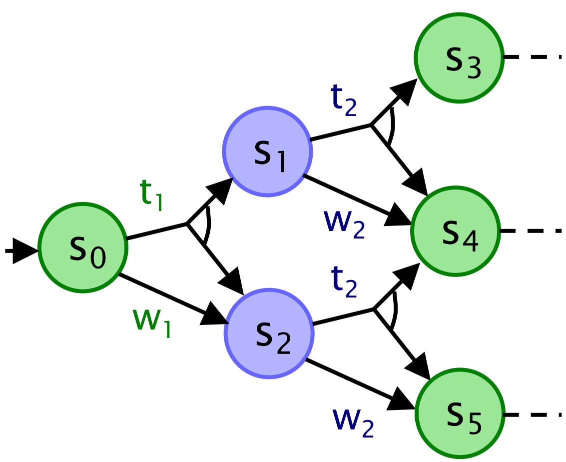
Concurrent stochastic games

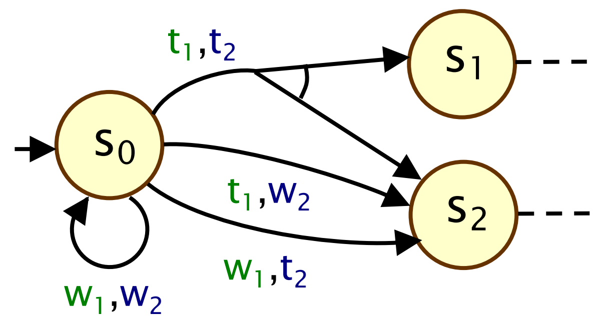
Concurrent stochastic games

- Motivation:
  - more realistic model of components operating concurrently, making action choices <u>without</u> knowledge of others

Turn-based stochastic games (TSGs)

Concurrent stochastic games (CSGs)





Concurrent stochastic games

- Concurrent stochastic games (CSGs)
  - players choose actions concurrently & independently
  - jointly determines (probabilistic) successor state
  - $\delta : S \times (A_1 \cup \{\bot\}) \times \ldots \times (A_n \cup \{\bot\}) \rightarrow Dist(S)$
  - generalises turn-based stochastic games

- We again use the logic rPATL for properties

- Same overall rPATL model checking algorithm [QEST'18]
  - key ingredient is now solving (zero-sum) 2-player CSGs
  - this problem is in PSPACE
  - note that optimal strategies are now randomised

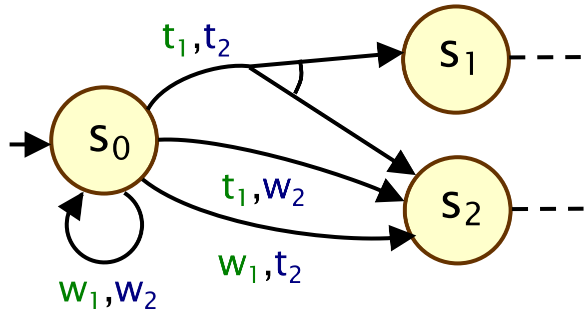
rPATL model checking for CSGs

- We again use a value iteration based approach
  - e.g. max/min reachability probabilities
  - $\sup_{\sigma_1} \inf_{\sigma_2} \Pr_s^{\sigma_1,\sigma_2}(F\ ✓)$ for all states s
  - values p(s) are the least fixed point of:

$$p(s) = \begin{cases} 1 & \text{if } s \vDash ✓ \\ val(Z) & \text{if } s \nvDash ✓ \end{cases}$$

  - where Z is the matrix game with $z_{ij} = \Sigma_{s'}\ \delta(s,(a_i,b_j))(s') \cdot p(s')$

- So each iteration solves a matrix game for each state
  - LP problem of size |A|, where A = action set

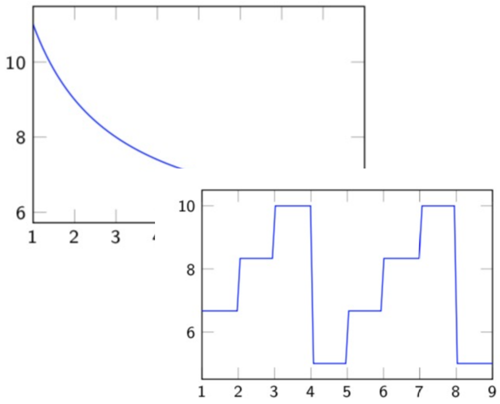


CSGs in PRISM-games

- CSG model checking implemented in PRISM-games 3.0
- Further extension of PRISM modelling language
- Explicit engine implementation
  - plus LP solvers for matrix game solution
  - this is the main bottleneck
  - experiments with CSGs up to ~3 million states
- Case studies:
  - future markets investor,
    trust models for user-centric networks,
    intrusion detection policies,
    multi-robot planning, …
    jamming radio systems

Example: 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



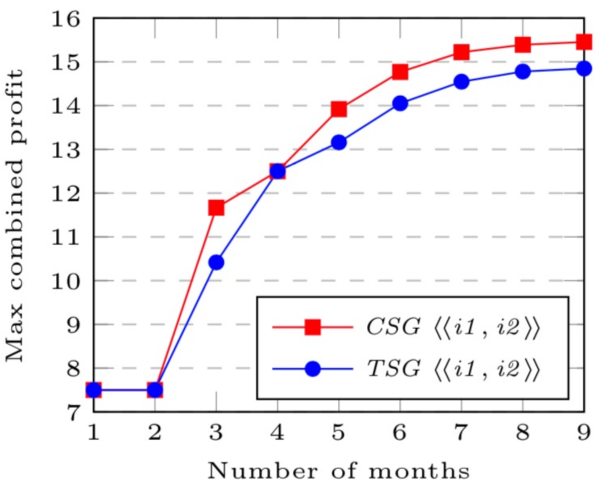
- Modelled as a 3-player CSG
  - extends simpler model originally from [McIver/Morgan'07]
  - investing/barring decisions are simultaneous
  - profit reduced for simultaneous investments
  - market cannot observe investors' decisions

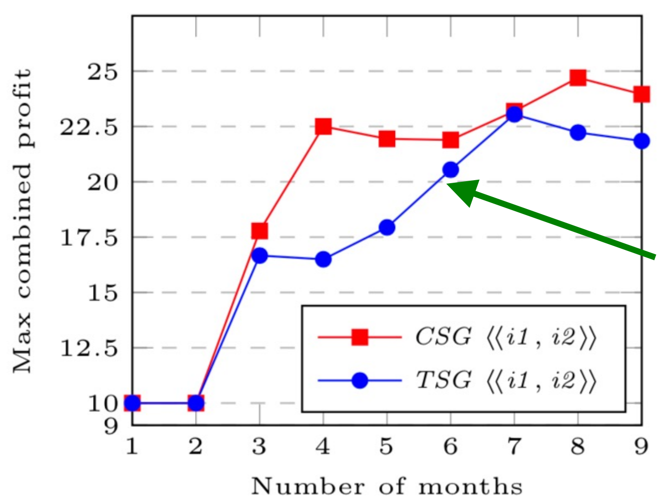
- Analysed with rPATL model checking & strategy synthesis
  - distinct profit models considered: 'normal market', 'later cash-ins' and 'later cash-ins with fluctuation'
  - comparison between TSG and CSG models

Example: Future markets investor

- Example rPATL query:
  - $\langle\langle investor_1, investor_2 \rangle\rangle \, R^{profit_{1,2}}_{max=?} \, [\, F \; finished_{1,2} \,]$
  - i.e. maximising joint profit

- Results: with (left) and without (right) fluctuations
  - optimal (randomised) investment strategies synthesised
  - CSG yields more realistic results (market has less power due to limited observation of investor strategies)





Too pessimistic: unrealistic strategy for adversary

Equilibria-based properties

Equilibria-based properties

- Motivation:
  - players/components may have distinct objectives but which are not directly opposing (non zero-sum)

Zero-sum properties → Equilibria-based properties

$\langle\langle robot_1 \rangle\rangle_{max=?}$ P [$F^{\leq k}$ $goal_1$]

$\langle\langle robot_1 : robot_2 \rangle\rangle_{max=?}$ (P [$F^{\leq k}$ $goal_1$]+P [F $^{\leq k}$ $goal_2$])

- We use Nash equilibria (NE)
  - no incentive for any player to unilaterally change strategy
  - actually, we use ϵ-NE, which always exist for CSGs
  - a strategy profile $\sigma=(\sigma_{1,\ldots,}\sigma_n)$ for a CSG is an ϵ-NE for state s and objectives $X_1,\ldots,X_n$ iff:
  - $Pr_s^{\sigma}(X_i) \geq \sup \{ Pr_s^{\sigma'}(X_i) \mid \sigma'=\sigma_{-i}[\sigma_i'] \text{ and } \sigma_i' \in \Sigma_i \} - \epsilon$ for all i

Social-welfare Nash equilibria

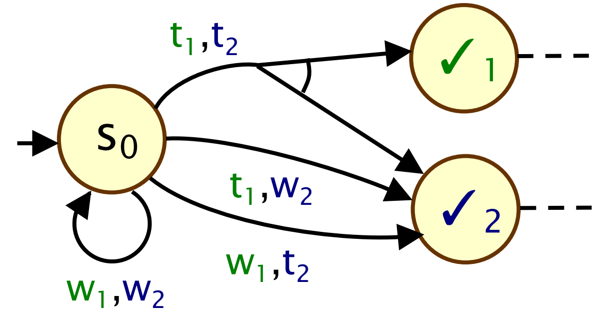
- Key idea: formulate model checking (strategy synthesis) in terms of social-welfare Nash equilibria (SWNE)
  - these are NE which maximise the sum $E_s^\sigma(X_1) + \ldots E_s^\sigma(X_n)$
  - i.e., optimise the players combined goal

- We extend rPATL accordingly

Zero-sum properties	→	Equilibria-based properties
$\langle\langle robot_1 \rangle\rangle_{max=?}\ P\ [\ F^{\leq k}\ goal_1\ ]$		$\langle\langle robot_1{:}robot_2 \rangle\rangle_{max=?}$ $(P\ [\ F^{\leq k}\ goal_1\ ]+P\ [F^{\leq k}\ goal_2])$
find a robot 1 strategy which maximises the probability of it reaching its goal, regardless of robot 2		find (SWNE) strategies for robots 1 and 2 where there is no incentive to change actions and which maximise joint goal probability

Model checking for extended rPATL

- Model checking for CSGs with equilibria
  - first: 2-coalition case [FM'19]
  - needs solution of bimatrix games
  - (basic problem is EXPTIME)
  - we adapt a known approach using labelled polytopes, and implement with an SMT encoding



- We further extend the value iteration approach:

$$p(s) = \begin{cases} (1,1) & \text{if } s \models \checkmark_1 \wedge \checkmark_2 \\ (p_{max}(s,\checkmark_2),1) & \text{if } s \models \checkmark_1 \wedge \neg\checkmark_2 \\ (1,p_{max}(s,\checkmark_1)) & \text{if } s \models \neg\checkmark_1 \wedge \checkmark_2 \\ val(Z_1,Z_2) & \text{if } s \models \neg\checkmark_1 \wedge \neg\checkmark_2 \end{cases}$$

standard MDP analysis (cases 2 and 3)

bimatrix game (case 4)

  - where Z_1 and Z_2 encode matrix games similar to before

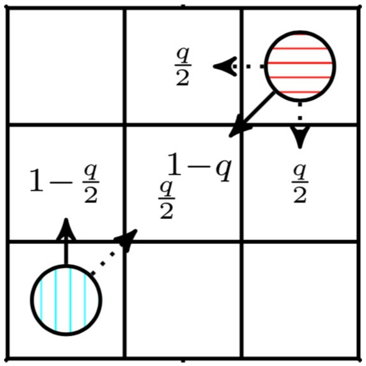
PRISM-games support

- Implementation in PRISM-games 3.0
  - bimatrix games solved using Z3/Yices encoding
  - optimised filtering of dominated strategies
  - scales up to CSGs with ~2 million states
  - extended to n-coalition case in [QEST'20]

- Applications & results
  - 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

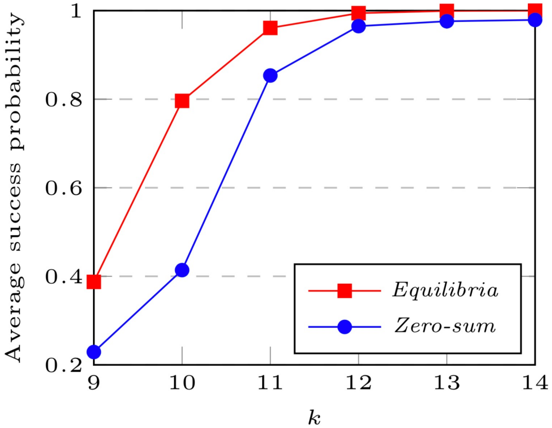
Example: multi-robot coordination

- 2 robots navigating an l x l grid
  - start at opposite corners, goals are to navigate to opposite corners
  - obstacles modelled stochastically: navigation in chosen direction fails with probability q



- We synthesise SWNEs to maximise the average probability of robots reaching their goals within time k
  - $\langle\langle robot1{:}robot2 \rangle\rangle_{max=?} (P [F^{\leq k} goal_1] + P [F^{\leq k} goal_2])$

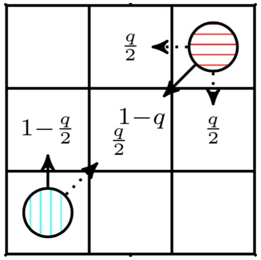
- Results (10 x 10 grid)
  - better performance obtained than using zero-sum methods, i.e., optimising for robot 1, then robot 2



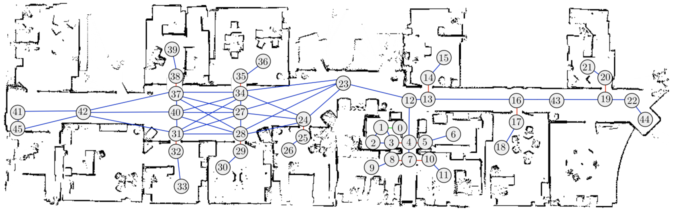
Future challenges

Challenges

- Partial information/observability
  - we need realisable strategies
  - leverage progress on POMDPs?



- Managing model uncertainty
  - integration with learning
  - robust verification



- Accuracy of model checking results
  - value iteration improvements; exact methods
- Scalability & efficiency
  - e.g. symbolic methods, abstraction, symmetry reduction
  - sampling-based strategy synthesis methods

PRISM-games

- See the PRISM-games website for more info
  - prismmodelchecker.org/games/
  - documentation, examples, case studies, papers
  - downloads: + CAV'20 artefact VM
  - open source (GPLV2): GitHub