Tutorial:
Planning in Formal Methods Land

Dave Parker

University of Birmingham

"Rigorous Automated Planning", Lorentz Centre, June 2022

Tutorial:

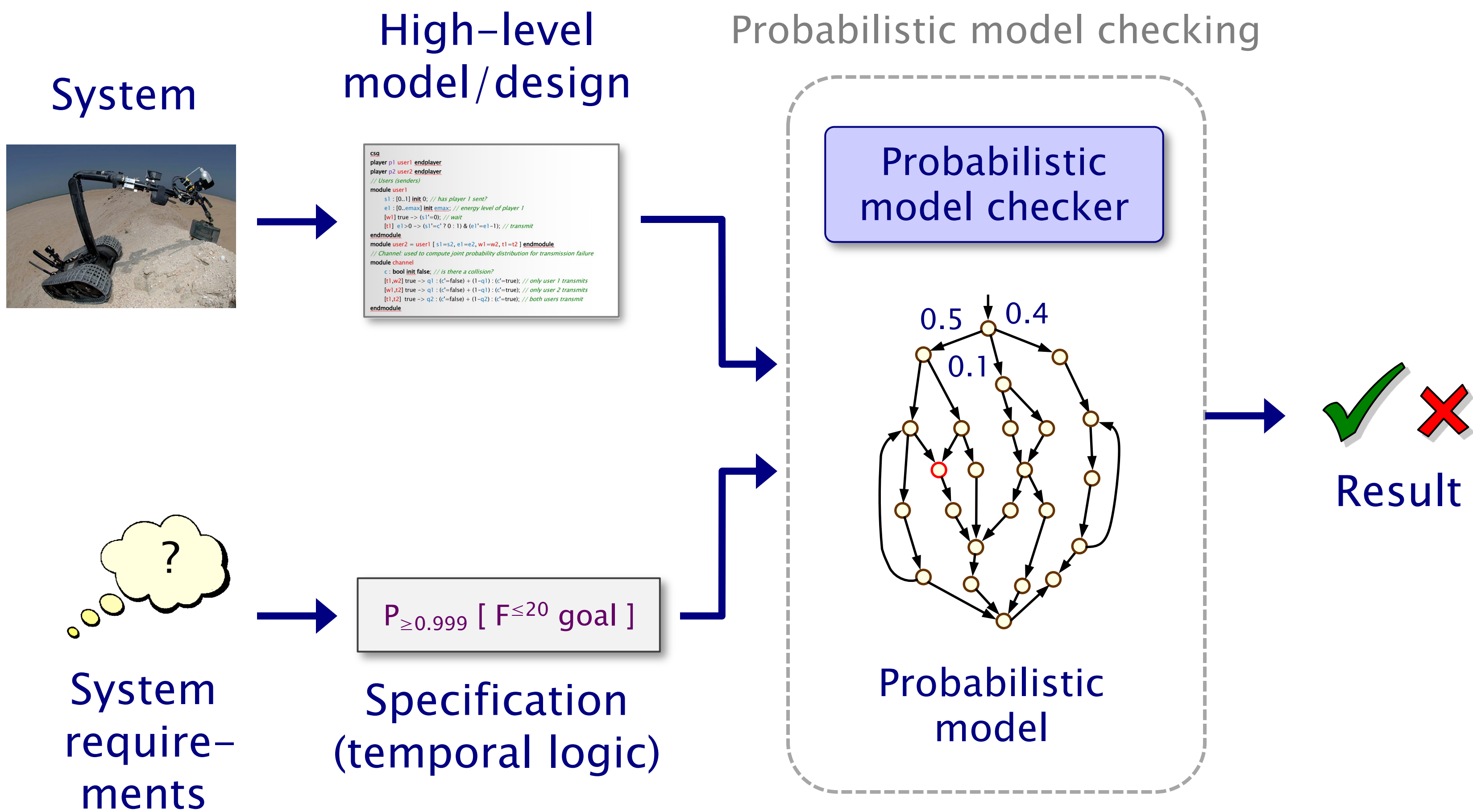
Planning with Probabilistic Model Checking

Dave Parker

University of Birmingham

“Rigorous Automated Planning”, Lorentz Centre, June 2022

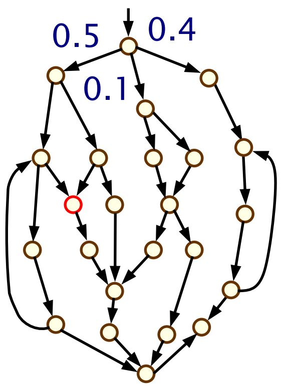
Probabilistic model checking



Probabilistic model checking

Probabilistic model checking

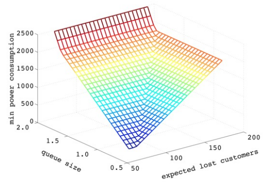
Probabilistic model checker

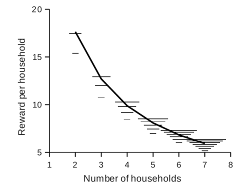


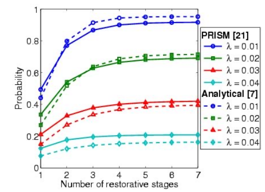
Probabilistic model

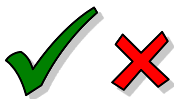
$P_{\geq 0.999}$ [$F^{\leq 20}$ goal]

Numerical results/analysis

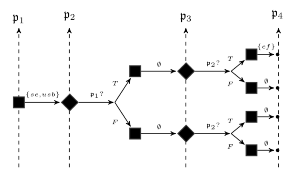


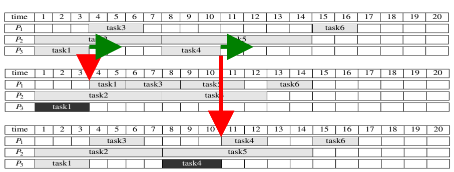






Result

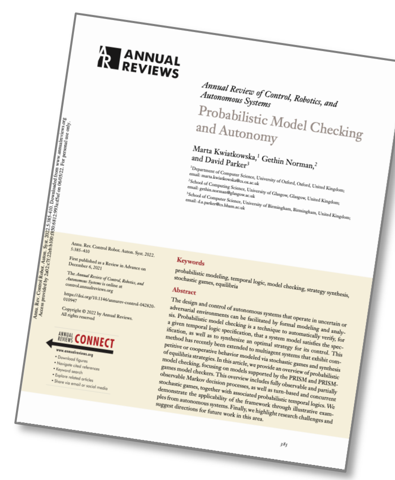




Strategies/policies/controllers

Overview

- Temporal logic
  - quantitative task specification/guarantees
- Techniques & tools
  - models, modelling languages
- Multi-agent planning
  - stochastic multi-player games


ANNUAL REVIEWS

Annual Review of Control, Robotics, and Autonomous Systems

Probabilistic Model Checking and Autonomy

Marta Kwiatkowska,[1] Gethin Norman,[2] and David Parker[3]

Keywords

probabilistic modeling, temporal logic, model checking, strategy synthesis, stochastic games, equilibria

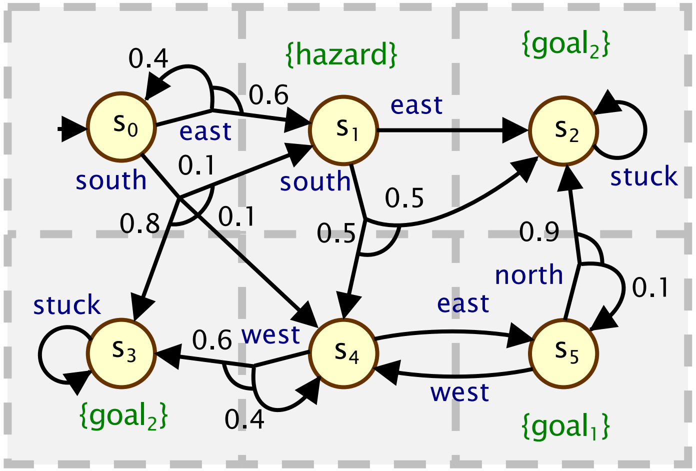
Temporal logic

Temporal logic

- Formal specification of desired behaviour
  - i.e., planning tasks/objectives
  - formal guarantees on resulting behaviour

- Simple examples (PCTL)
  - Probabilistic reachability
    $P_{\geq 0.7}$ [F goal$_1$]
    $P_{\geq 0.6}$ [$F^{\leq 10}$ goal$_1$]
  - Probabilistic safety/invariance
    $P_{\geq 0.99}$ [G¬hazard]
  - Numerical queries
    $P_{max=?}$ [F goal$_1$]

Example MDP (robot navigation)



- For planning with MDPs:
  - $P_{\sim p}[\psi]$ means: find a policy/strategy σ satisfying $Pr^{\sigma}(\psi) \sim p$

Linear temporal logic (LTL)

- Logic for describing properties of executions [Pnueli]
- LTL syntax:
  - $\psi ::= \text{true} \mid a \mid \psi \wedge \psi \mid \neg\psi \mid X\,\psi \mid \psi\,U\,\psi \mid F\,\psi \mid G\,\psi$
- Propositional logic + temporal operators:
  - a is an atomic proposition (labelling a state)
  - $X\,\psi$ means "ψ is true in the next state"
  - $F\,\psi$ means "ψ is eventually true"
  - $G\,\psi$ means "ψ always remains true"
  - $\psi_1\,U\,\psi_2$ means "ψ_2 is true eventually and ψ_1 is true until then"
- Common alternative notation:
  - $\bigcirc$ (next), $\diamond$ (eventually), $\Box$ (always) , U (until)

Linear temporal logic (LTL)

- LTL syntax:
  - $\psi ::= \text{true} \mid a \mid \psi \wedge \psi \mid \neg\psi \mid X\,\psi \mid \psi\, U\, \psi \mid F\,\psi \mid G\,\psi$
- Commonly used LTL formulae:
  - $G\,(a \rightarrow F\,b)$ – "b always eventually follows a"
  - $G\,(a \rightarrow X\,b)$ – "b always immediately follows a”
  - $G\,F\,a$ – "a is true infinitely often"
  - $F\,G\,a$ – "a becomes true and remains true forever"
- Robot task specifications in LTL (for MDPs)
  - e.g. $P_{>0.7}$ [(G¬hazard) ∧ (GF $goal_1$)] – ”the probability of avoiding hazard and visiting $goal_1$ infinitely often is > 0.7"
  - e.g. $P_{max=?}$ [¬$zone_3$ U ($zone_1$ ∧ (F $zone_4$))] – "max. probability of patrolling zones 1 then 4, without passing through 3?”

Temporal logic

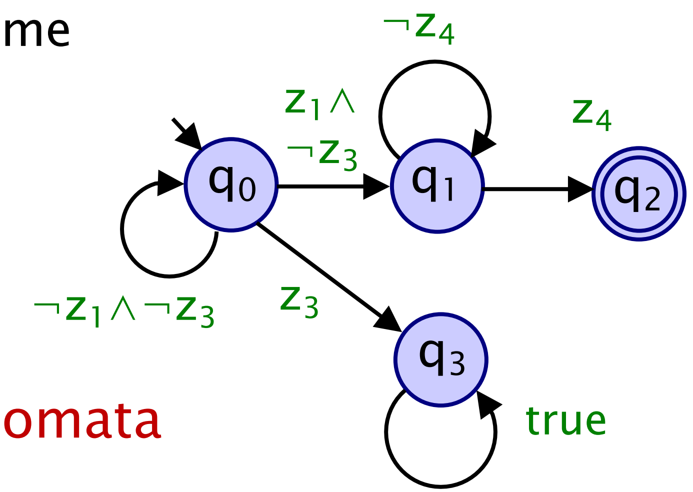
- Benefits of temporal logic
  - flexible, unambiguous behavioural specification
    - broad range of quantitative properties expressible
  - (probabilistic) guarantees on safety, performance, etc.
    - meaningful properties: event probabilities, time, energy,…

$$P_{>0.7} [(G\neg hazard) \wedge (GF\ goal_1)]$$

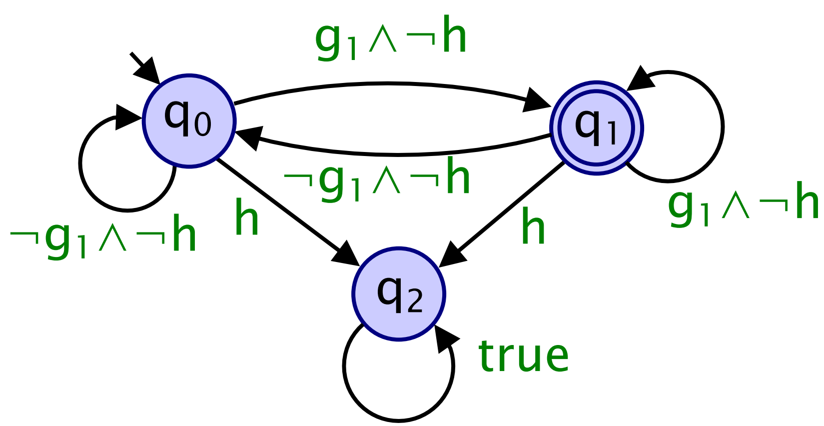
    - (c.f. ad-hoc reward structures, e.g. with discounting)
    - caveat: accuracy of model (and its solution)
  - efficient LTL-to-automata translation
    - optimal (finite-memory) policy synthesis (via product MDP)
    - correctness monitoring / shielding
    - task progress metrics

LTL & automata

- Safe/co-safe LTL: (deterministic) finite automata
  - (non-)satisfaction occurs in finite time
  - $\neg zone_3 \text{ U } (zone_1 \wedge (\text{F } zone_4))$

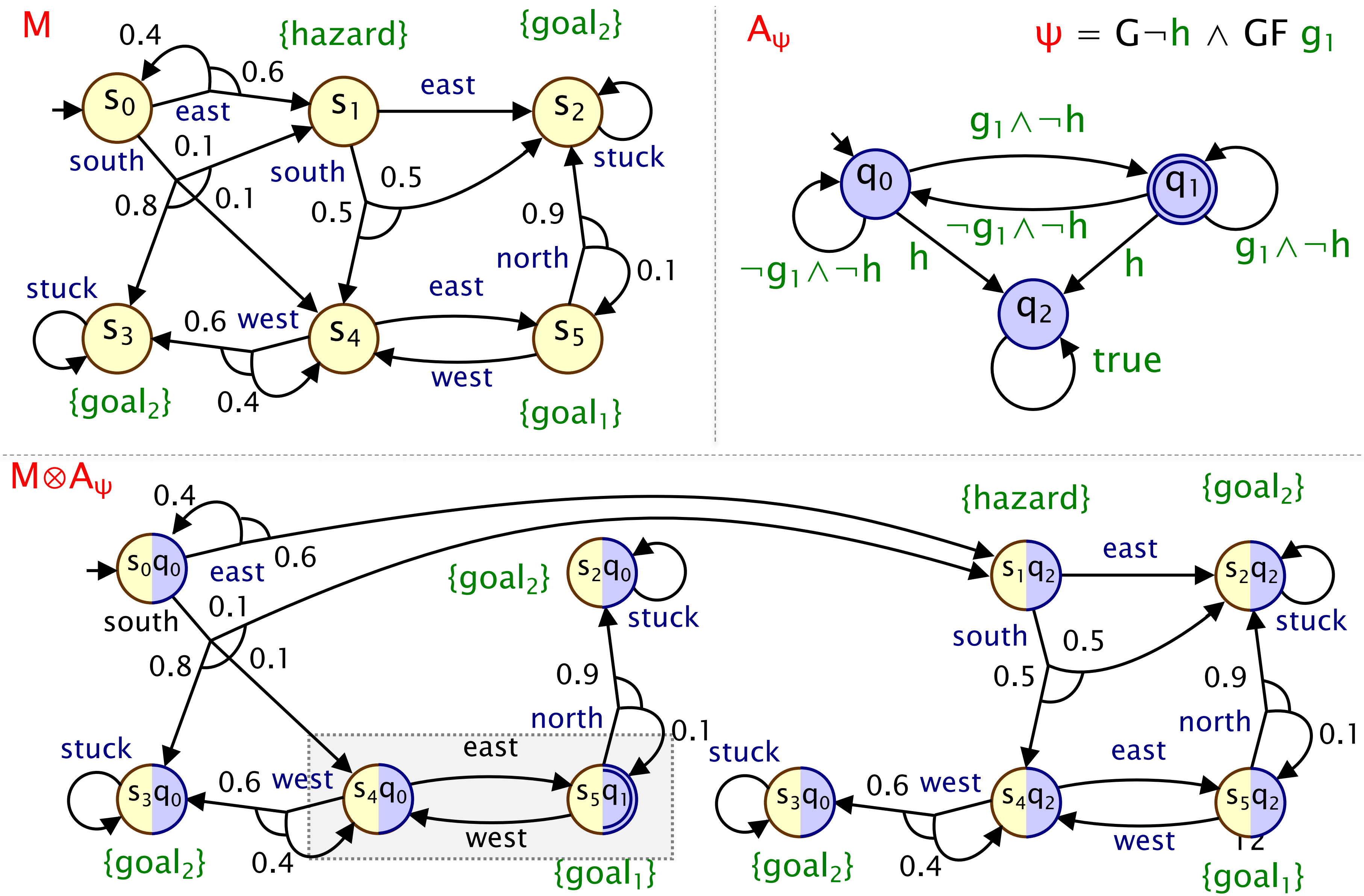


- Full LTL: e.g. (det.) Rabin/Buchi automata
  - $\text{G}\neg hazard \wedge \text{GF } goal_1$

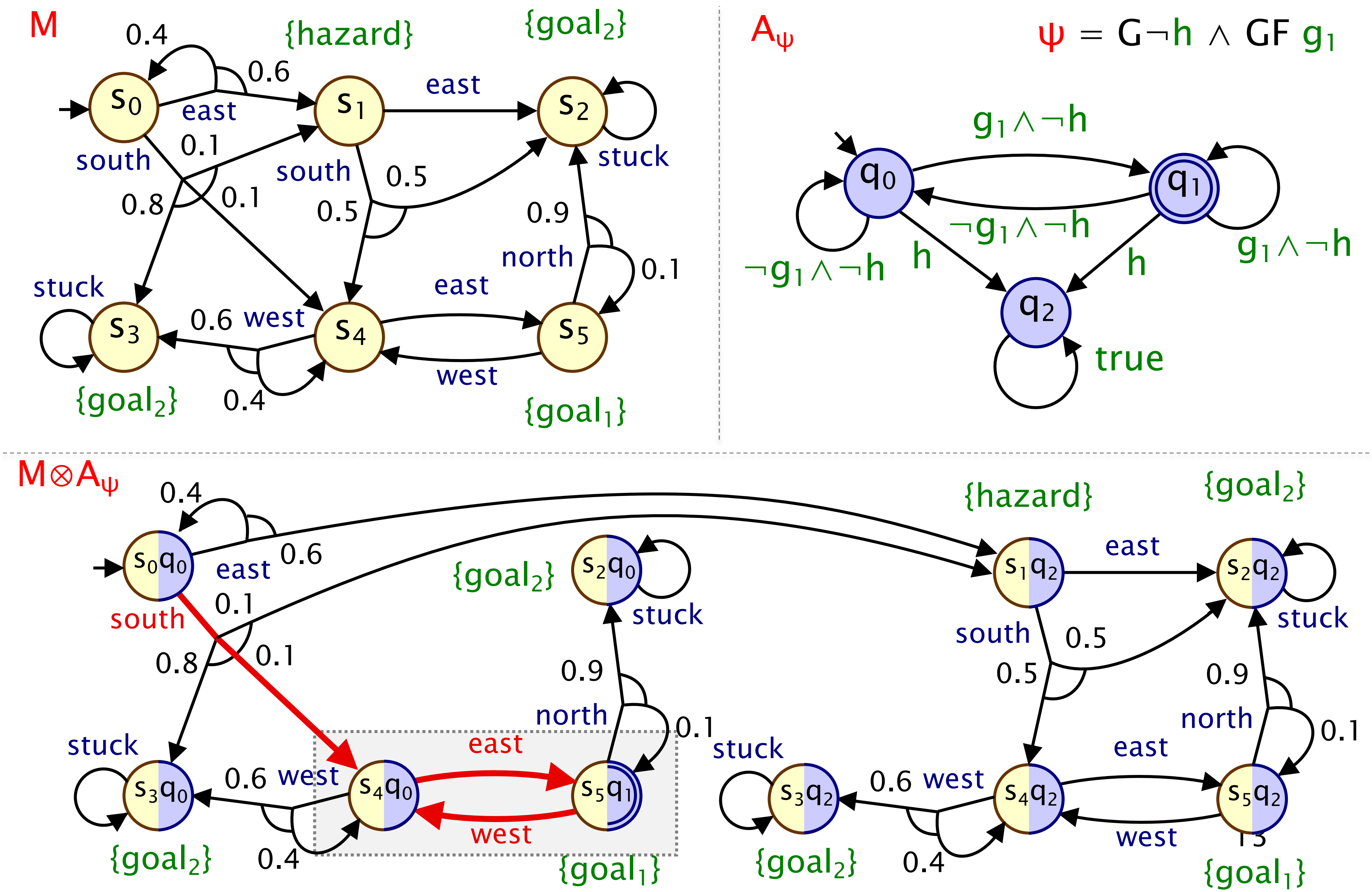


- Other useful LTL subclasses
  - GR(1), LTL\GU, …

LTL planning via product MDP



LTL planning via product MDP



Costs & Rewards

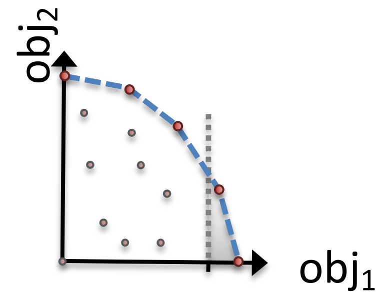
- Costs & rewards
  - i.e., values assigned to model states or state-action pairs

- Temporal logic examples
  - $R^{hazard}_{\leq 1.5}$ [$C^{\leq 20}$] – the expected number of times that the robot enters the hazard location within 20 steps is at most 1.5
  - $R^{energy}_{min=?}$ [F goal] – minimise the expected energy consumption until the the goal is reached
  - $R^{time}_{min=?}$ [$\neg zone_3$ U ($zone_1 \wedge$ (F $zone_4$))] – minimise expected time to patrol zones 1 then 4, without passing through 3

- Notes:
  1. the above use PRISM's R (reward) operator, even for costs
  2. discounted rewards are more rarely used in this context

More temporal logic

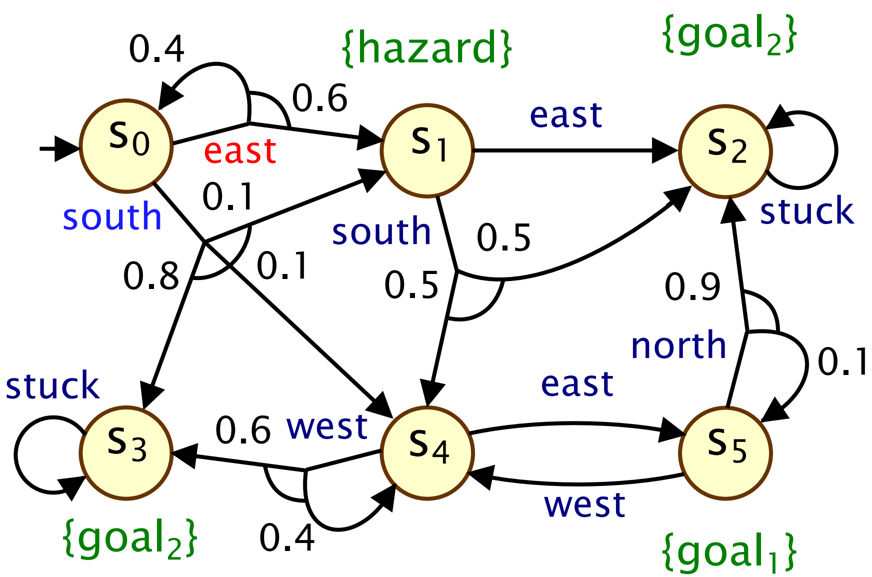
- Multi-objective queries
  - e.g. $\langle\langle * \rangle\rangle$ ($P_{max=?}$ [GF $goal_1$], $P_{\geq 0.7}$ [G ¬hazard])
  - max. objective 1 subject to constrained objective 2
  - also: achievability & Pareto queries

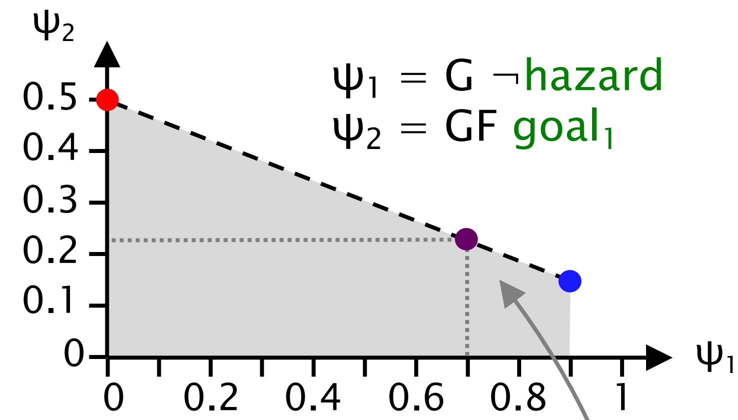


- Nested (branching-time) queries
  - e.g. $R_{min=?}$ [$P_{\geq 0.99}$ [$F^{\leq 10}$ base] U ($zone_1$ ∧ (F $zone_4$))]
  - "minimise expected time to visit zones 1 then 4, whilst ensuring the base can always be reliably reached

- And more
  - cost-bounded, conditional probabilities, quantiles
  - metric temporal logic, signal temporal logic
  - …

Multi-objective specifications





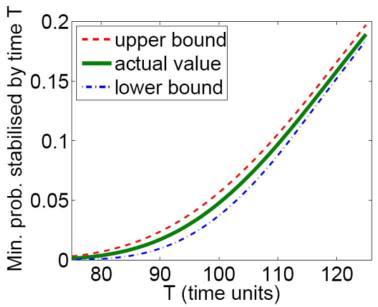
randomised, finite-memory optimal policy

- Achievability query
  - $P_{\geq 0.7}$ [G ¬hazard] ∧ $P_{\geq 0.2}$ [GF goal$_1$] ?
- Numerical query
  - $P_{max=?}$ [GF goal$_1$] such that $P_{\geq 0.7}$ [G ¬hazard] ?
- Pareto query
  - for $P_{max=?}$ [G ¬hazard], $P_{max=?}$ [GF goal$_1$] ?

Techniques & tools

Verification techniques

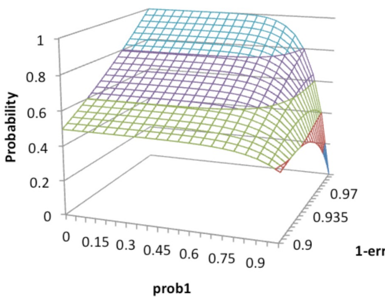
- Probabilistic model checking techniques
  - automata + graph analysis + numerical solution
  - often more focus on exhaustive/“exact”/optimal methods
  - e.g., for MDPs: value iteration (VI), linear programming
- But: known accuracy and convergence issues
  - interval iteration, sound VI, optimistic VI
  - separate convergence from above and below

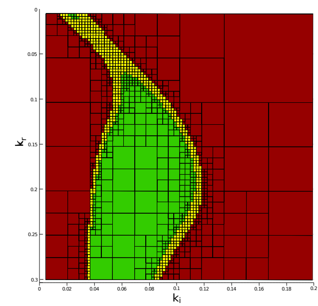


- Scalability vs accuracy/guarantees
  - scalability/efficiency is always an issue
  - statistical model checking: sampling-based methods
  - abstraction + sound bounds (often property driven)

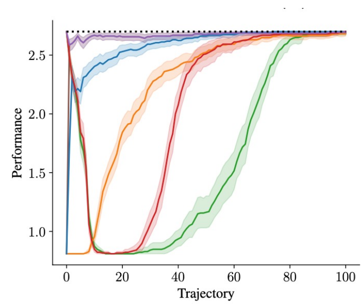
Probabilistic verification: directions

- Research directions
  - parametric model checking
    - e.g., for parameter synthesis, sensitivity analysis

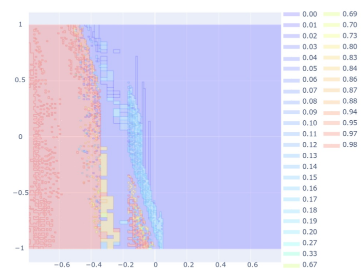




  - quantification of uncertainty
    - e.g. robust verification with interval MDPs, convex optimisation



  - verification + machine learning
    - learnt policies
      e.g. (sampling/heuristics? neural nets?)
    - learnt models + parameters



Verification tools

- Probabilistic verification tools
  - PRISM (and PRISM-games), STORM, MODEST, ePMC
  - general purpose probabilistic model checking tools, wide range of models (Markov chains, (PO)MDPs, games), many temporal logics & solution techniques
- Real-time verification tools
  - UPPAAL (and UPPAAL-Stratego/Tiga/CORA/SMC/…)
  - timed automata, plus stochastic & game variants
- Also many other specialised tools
  - PET (partial exploration, sampling)
  - Prophesy (parametric techniques)
  - FAUST^2, StocHy (continuous space/hybrid systems)
  - …

Modelling languages

- Example languages for formal model specification
  - PRISM: textual language, based on guarded commands
  - UPPAAL: graphical/textual description of automata networks

Modelling languages

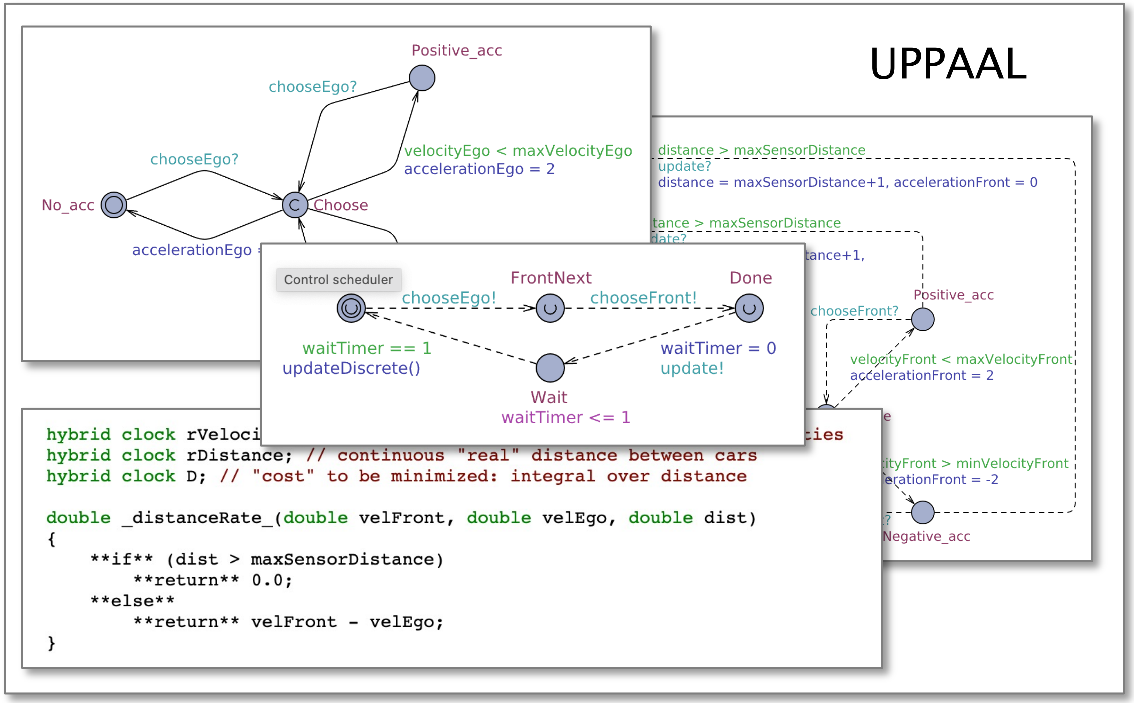
- Example languages for formal model specification
  - PR ... nds
  - UP ... networks

PRISM-games

```
csg // Model type: concurrent stochastic game
player p1 user1 endplayer    player p2 user2 endplayer
// Parameters
const int emax;  const double q1;  const double q2 = 0.9 * q1;
// Modules: users (senders) + channel
module user1
        s1 : [0..1] init 0; // has player 1 sent?
        e1 : [0..emax] init emax; // energy level of player 1
        [w1] true -> (s1'=0); // wait
        [t1]  e1>0 -> (s1'=c' ? 0 : 1) & (e1'=e1-1); // transmit
endmodule
module user2 = user1 [ s1=s2, e1=e2, w1=w2, t1=t2 ] endmodule
module channel
        c : bool init false; // is there a collision?
        [t1,w2] true -> q1 : (c'=false) + (1-q1) : (c'=true); // only user 1 transmits
        [w1,t2] true -> q1 : (c'=false) + (1-q1) : (c'=true); // only user 2 transmits
        [t1,t2]  true -> q2 : (c'=false) + (1-q2) : (c'=true); // both users transmit
endmodule
// Reward structures: energy usage
rewards "energy" [t1] true: 1.5; [t2] true: 1.2; endrewards
```

Modelling languages

- Example languages for formal model specification
  - PRISM: textual language, based on guarded commands
  - UPPAAL: graphical/textual description of automata networks



Modelling languages

- Example languages for formal model specification
  - PRISM: textual language, based on guarded commands
  - UPPAAL: graphical/textual description of automata networks
- Some key modelling language features
  - Compositional model specifications
    - components, parallel composition, communication
  - Parameterised models
    - probabilities, sizes, components
- Challenges
  - language/tool interoperability
    - e.g., JANI (models), PPDDL (planning), HOAF (automata), tool APIs
  - modelling stochasticity/uncertainty
    - probabilistic programming languages?

Models, models, models…

- Wide range of probabilistic models

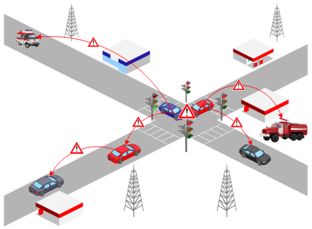
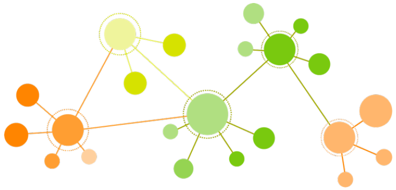
  discrete states & probabilities: Markov chains
  + nondeterminism: Markov decision processes (MDPs)
  + real-time clocks: probabilistic timed automata (PTAs)
  + uncertainty: interval MDPs (IMDPs)
  + partial observability: partially observable MDPs (POMDPs)
  + multiple players: (turn-based) stochastic games
  + concurrency: concurrent stochastic games

- And many others
  - stochastic timed automata
  - stochastic hybrid automata
  - Markov automata
  - …

Multi-agent planning

Verification with stochastic games

- How do we plan rigorously 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

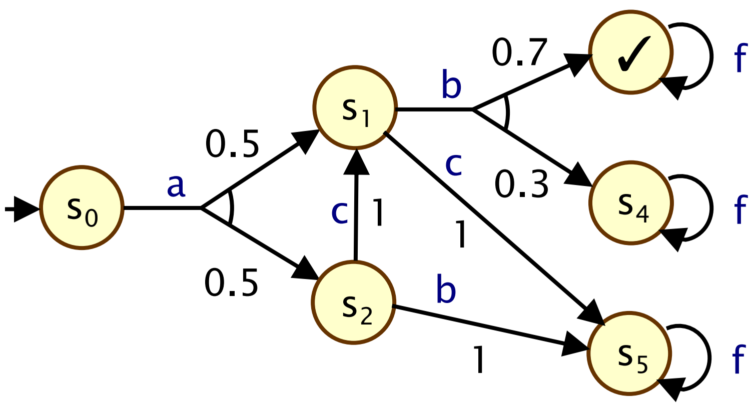
- Verification with stochastic multi-player games
  - verification (and synthesis) of strategies that are robust in adversarial settings and stochastic environments

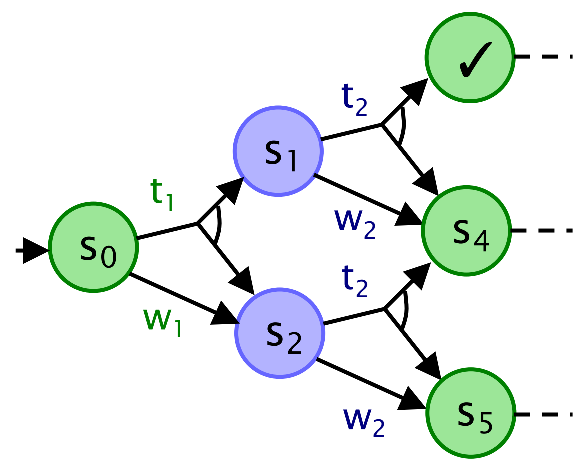
Stochastic multi-player games

- 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:

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

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

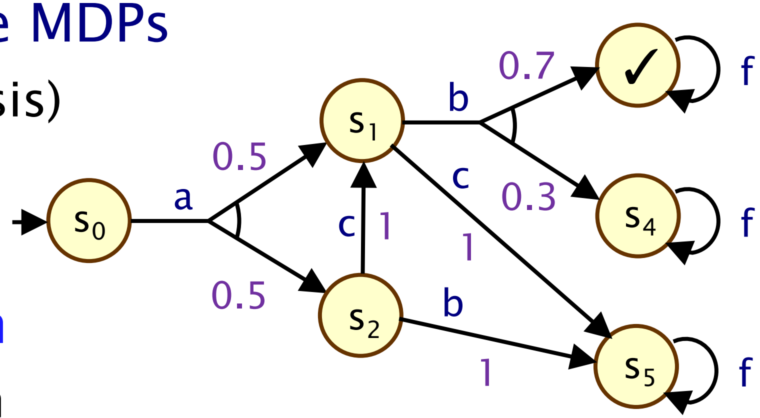
$$\rho ::= I^{=k} \mid C^{\leq k} \mid F\,\phi$$

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

Reminder: 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:

$\delta : S \times Act \rightarrow Dist(S)$

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

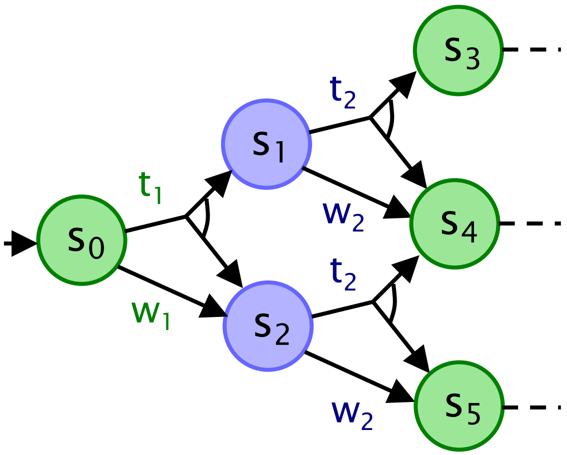
$$p(s) = \begin{cases} 1 & \text{if } s \models \checkmark \\ \max_a \Sigma_{s'} \delta(s,a)(s') \cdot p(s') & \text{otherwise} \end{cases}$$

  - basis for iterative numerical computation

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

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: $\sup_{\sigma_1} \inf_{\sigma_2} Pr_s^{\sigma_1,\sigma_2}(F ✓)$
  - complexity: NP ∩ coNP (if we omit some reward operators)



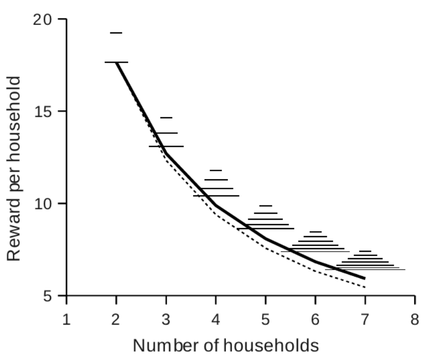
- We again use value iteration
  - values $p(s)$ are the least fixed point of:

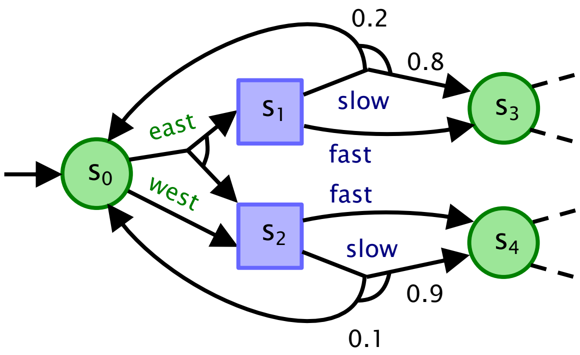
$$p(s) = \begin{cases} 1 & \text{if } s \models ✓ \\ \max_a \sum_{s'} \delta(s,a)(s') \cdot p(s') & \text{if } s \not\models ✓ \text{ and } s \in S_1 \\ \min_a \sum_{s'} \delta(s,a)(s') \cdot p(s') & \text{if } s \not\models ✓ \text{ and } s \in S_2 \end{cases}$$

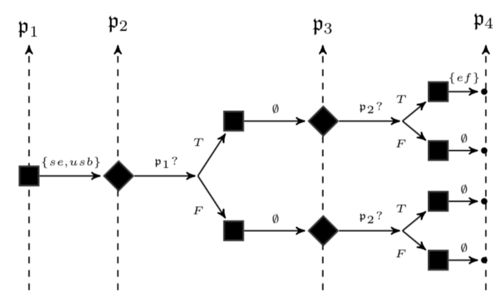
  - and more: graph-algorithms, sequences of fixed points, …

Applications

- Example application domains (PRISM-games)
  - 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



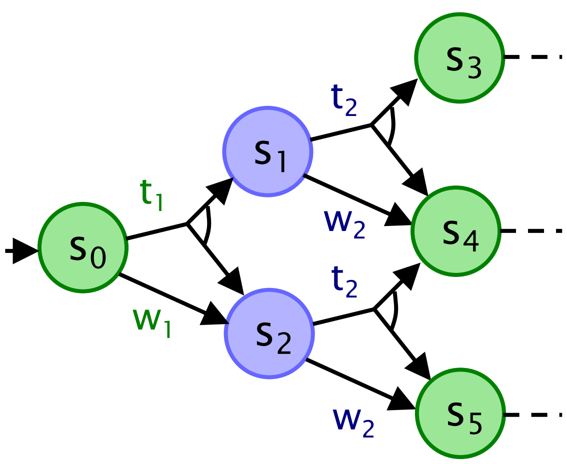


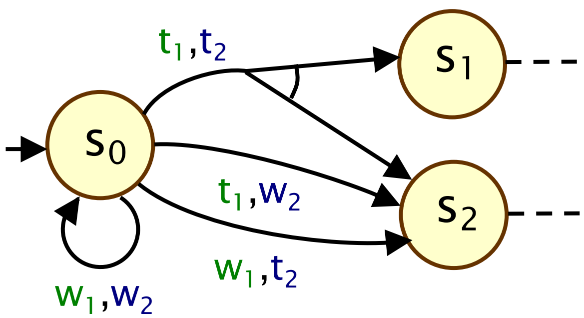


Concurrent stochastic games

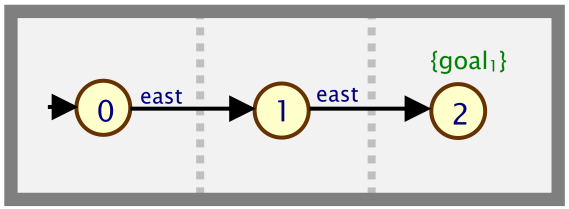
- Motivation:
  - more realistic model of components operating concurrently, making action choices <u>without</u> knowledge of others

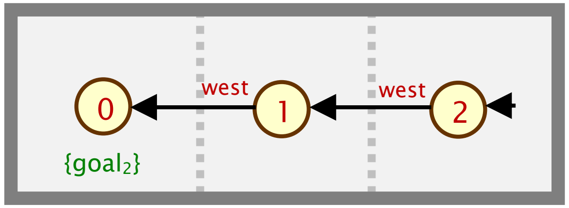
Turn-based stochastic games (TSGs) → Concurrent stochastic games (CSGs)

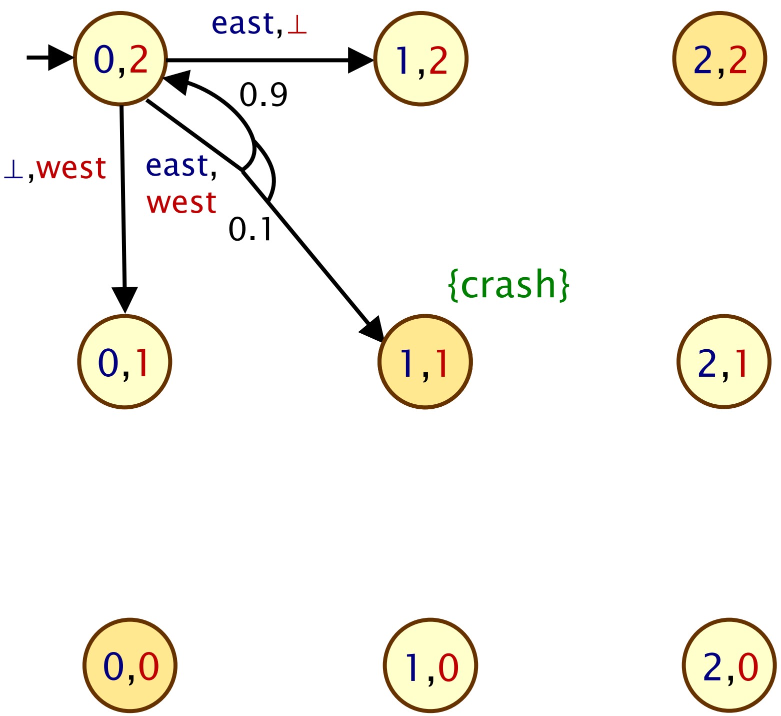




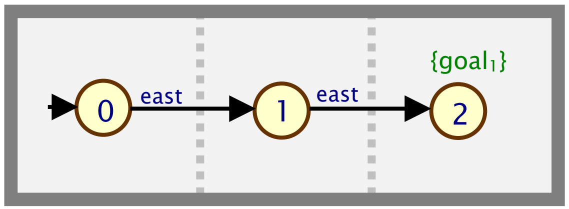
CSG for 2 robots on a 3x1 grid

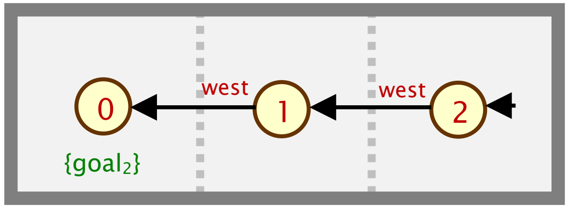


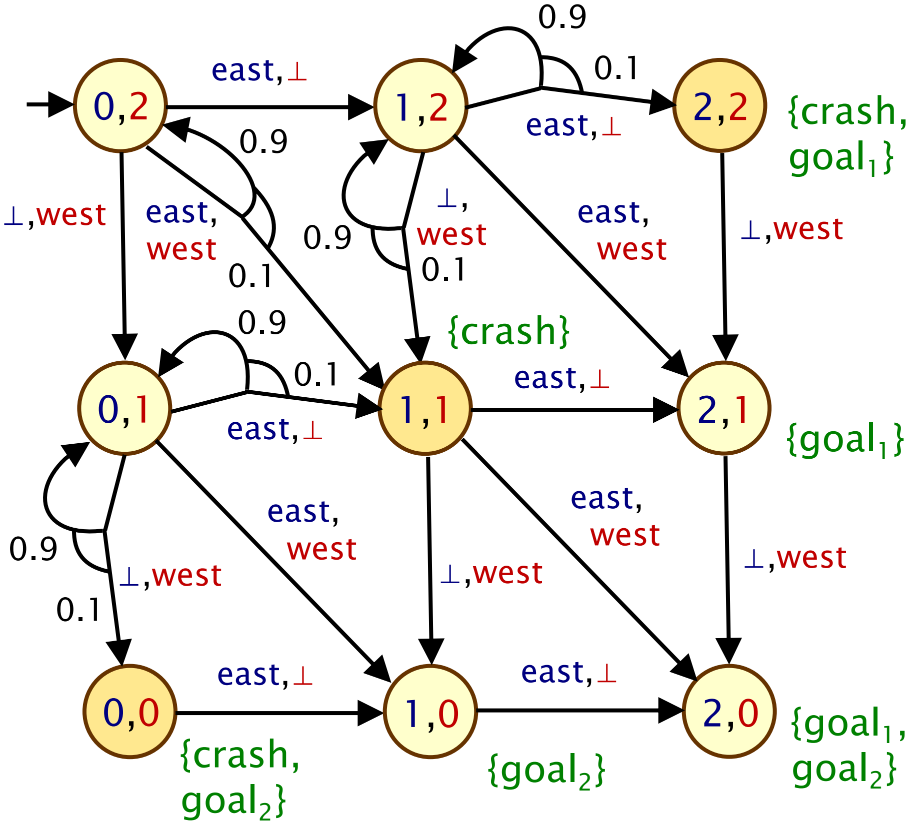




CSG for 2 robots on a 3x1 grid







Concurrent stochastic games

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

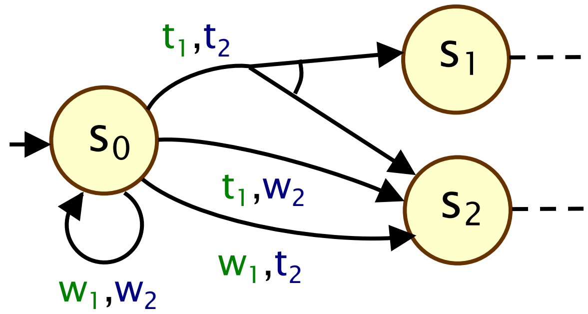
rPATL model checking for CSGs

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

$$p(s) = \begin{cases} 1 & \text{if } s \models \checkmark \\ val(Z) & \text{if } s \not\models \checkmark \end{cases}$$

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

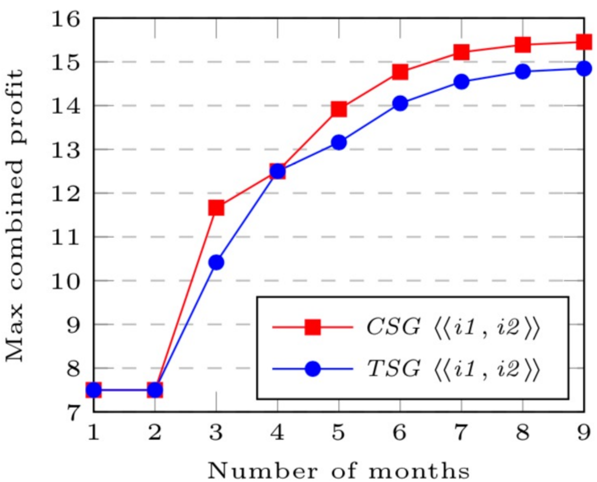
- So each iteration solves a matrix game for each state
  - LP problem of size |A|, where A = action set

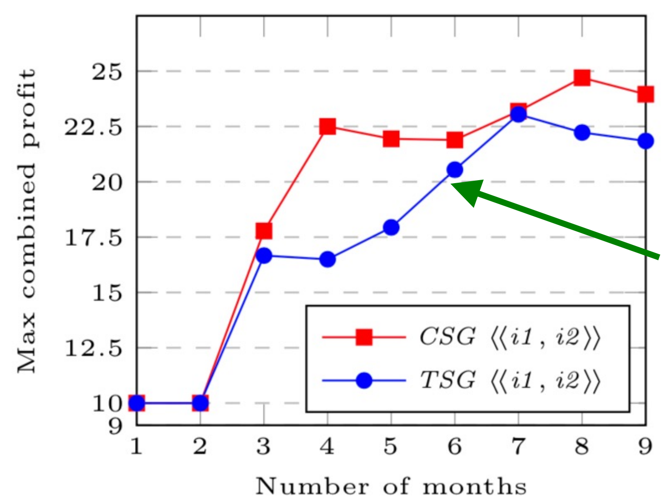


Example: Future markets investor

- 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

- 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 $\sigma=(\sigma_{1,\dots,}\sigma_n)$ for a CSG is an ϵ-NE for state s and objectives $X_1,\dots,X_n$ 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

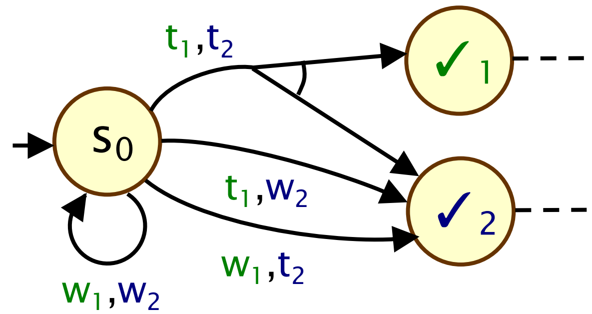
- Key idea: formulate model checking (strategy synthesis) in terms of social-welfare Nash equilibria (SWNE)
  - these are NE which maximise the sum $E_s^{\sigma}(X_1) + \ldots E_s^{\sigma}(X_n)$
  - i.e., optimise the players combined goal

- 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:

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

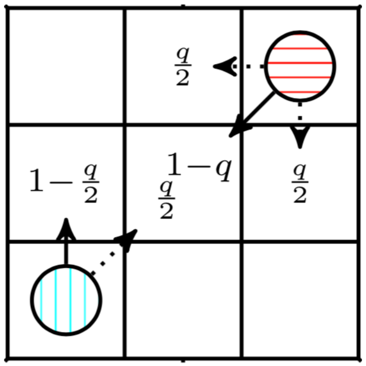
standard MDP analysis (second and third cases)

bimatrix game (fourth case)

  - where Z_1 and Z_2 encode matrix games similar to before

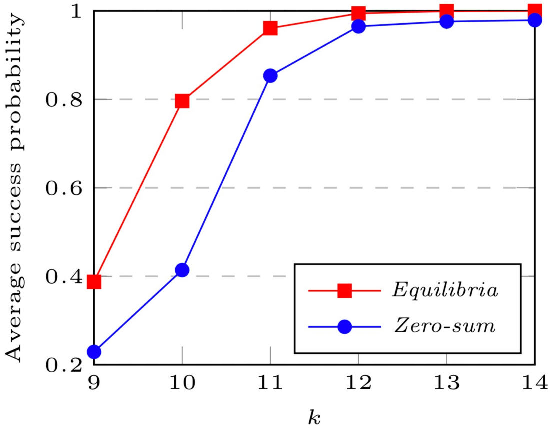
Example: multi-robot coordination

- 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
  - $\langle\langle robot_1:robot_2 \rangle\rangle_{max=?}$ (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

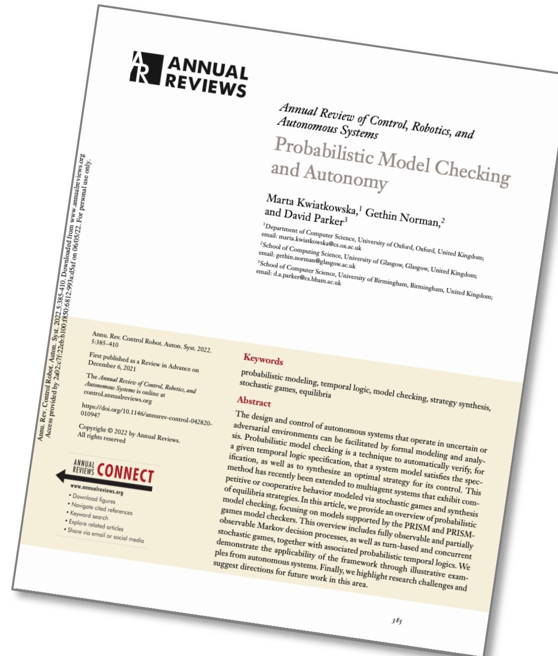


Conclusions

Conclusions

- Planning & formal verification
  - temporal logics & automata
  - tools, techniques, modelling languages
  - multi-agent systems

- Challenges
  - partial information/observability
  - managing model uncertainty
  - integration with machine learning
  - scalability & efficiency vs accuracy



Annual Review of Control, Robotics, and Autonomous Systems

Probabilistic Model Checking and Autonomy

Marta Kwiatkowska,[1] Gethin Norman,[2] and David Parker[3]

[1]Department of Computer Science, University of Oxford, Oxford, United Kingdom; email: marta.kwiatkowska@cs.ox.ac.uk

[2]School of Computing Science, University of Glasgow, Glasgow, United Kingdom; email: gethin.norman@glasgow.ac.uk

[3]School of Computer Science, University of Birmingham, Birmingham, United Kingdom; email: d.a.parker@cs.bham.ac.uk

Annu. Rev. Control Robot. Auton. Syst. 2022. 5:385–410

First published as a Review in Advance on December 6, 2021

The *Annual Review of Control, Robotics, and Autonomous Systems* is online at control.annualreviews.org

https://doi.org/10.1146/annurev-control-042820-010947

Keywords

probabilistic modeling, temporal logic, model checking, strategy synthesis, stochastic games, equilibria

Abstract

The design and control of autonomous systems that operate in uncertain or adversarial environments can be facilitated by formal modeling and analysis. Probabilistic model checking is a technique to automatically verify, for a given temporal logic specification, that a system model satisfies the specification, as well as to synthesize an optimal strategy for its control. This method has recently been extended to multiagent systems that exhibit competitive or cooperative behavior modeled via stochastic games and synthesis of equilibria strategies. In this article, we provide an overview of probabilistic model checking, focusing on models supported by the PRISM and PRISM-games model checkers. This overview includes fully observable and partially observable Markov decision processes, as well as turn-based and concurrent stochastic games, together with associated probabilistic temporal logics. We demonstrate the applicability of the framework through illustrative examples from autonomous systems. Finally, we highlight research challenges and suggest directions for future work in this area.

385

More details and references here