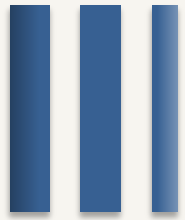


Verified Sequential Decision Making under Uncertainty

Dave Parker

UNIVERSITY OF OXFORD

CDT in Safe & Trusted AI, July 2025

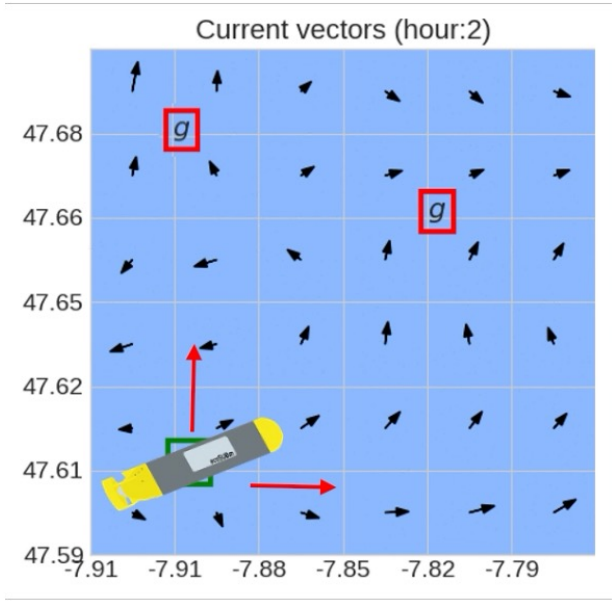
Overview

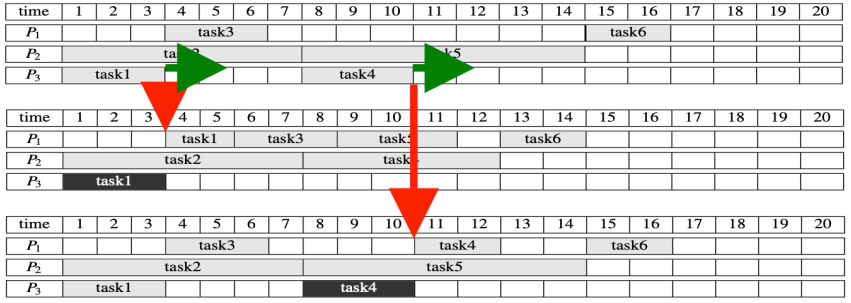
- Sequential decision making under uncertainty
- Formal verification: probabilistic model checking
  - key ideas and example applications
  - probabilistic models
  - temporal logic & automata

- Multi-agent decision making
  - stochastic games
- Data-driven models for decision making
  - robustness under epistemic uncertainty
- Neuro-symbolic decision making

Sequential decision making under uncertainty

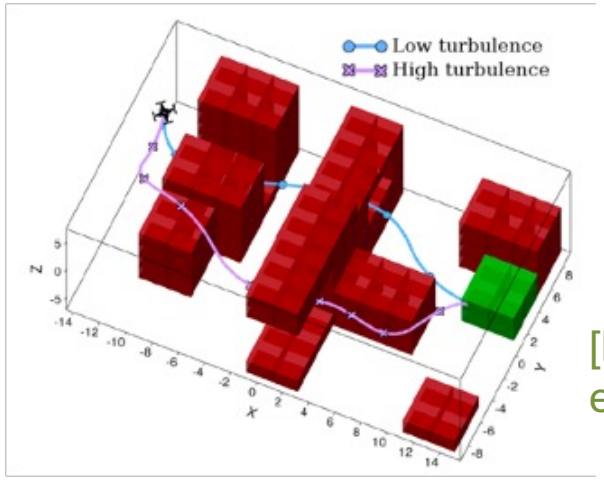
- Sequential decision making
  - iterative interaction with an environment to achieve a goal
  - sequential process of making observations and executing actions
  - applications in: health, energy, transportation, robotics, …
- Sequential decision making under uncertainty
  - noisy sensors, unpredictable conditions, lossy communication, human behaviour, hardware failures, …
- Trustworthy, safe and robust decision making
  - e.g. for safety-critical applications
  - needs rigorous/systematic quantification of uncertainty





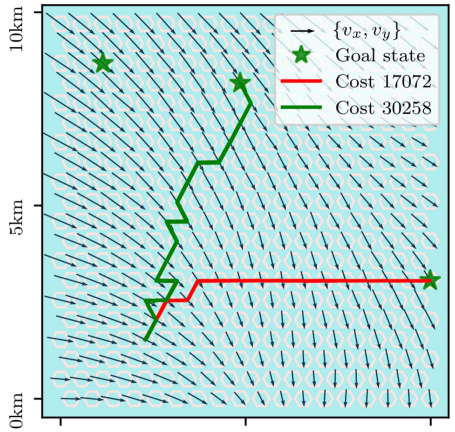
Applications & challenges

- Unmanned aerial vehicles
  - robust control in the presence of turbulence



[Badings et al.'23]

- Autonomous underwater vehicle
  - safe & effective navigation in unknown ocean currents

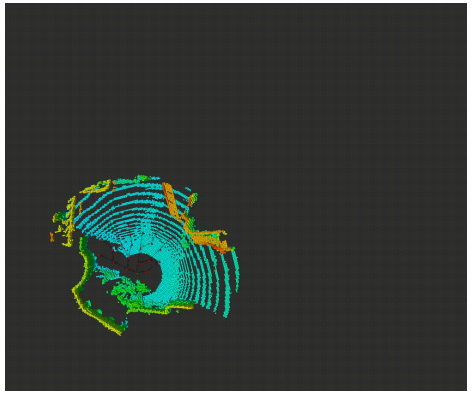


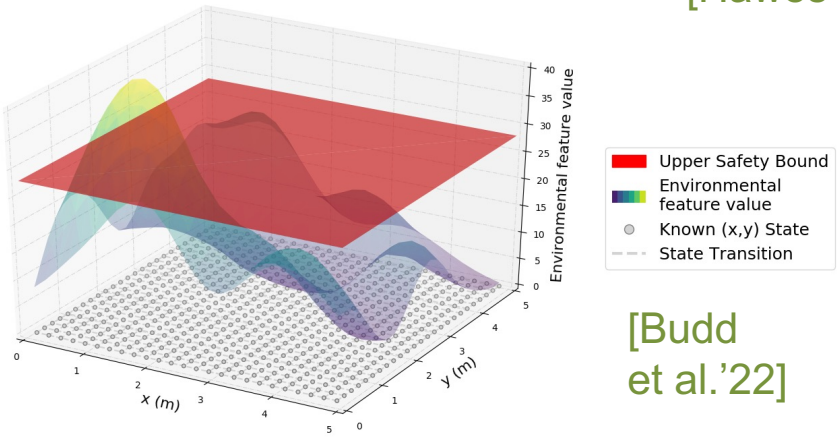
[Budd et al.'22]

- Mobile robots around humans
  - reliable navigation in offices & care homes

[Hawes et al.'17]

- Mine exploration
  - Safe exploration and mapping (avoiding radiation)

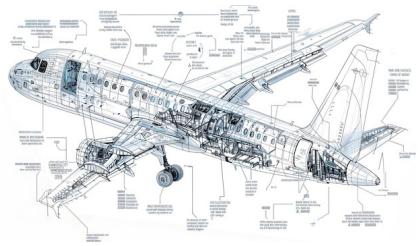




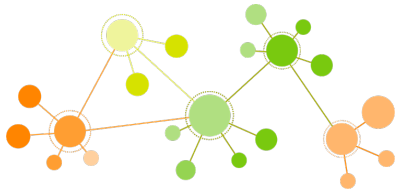
[Budd et al.'22]

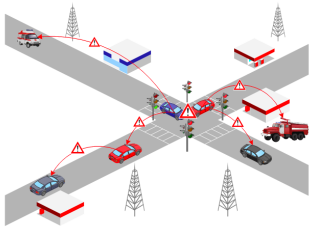
Formal verification

- Computer-aided formal verification
  - how do I (automatically) prove that my program/protocol/design is correct?
  - particularly important for safety critical systems


- How do we do this in the presence of uncertainty?
  - hardware failures, randomisation, unreliable sensors, unpredictable environments, …





Probabilistic model checking

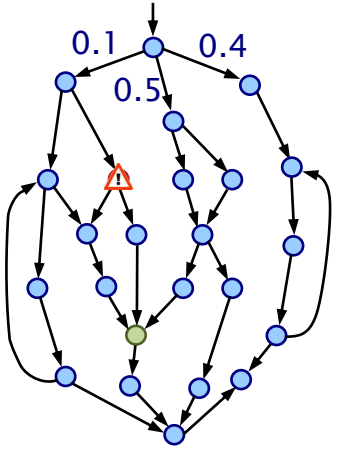
Probabilistic model checking

- Automated verification of stochastic systems
  - systematic construction and analysis of probabilistic models
  - key ingredients: logic, automata, probability

- Connections to:
  - Markov models, graph theory, artificial intelligence, control theory, optimisation, game theory, SAT, …

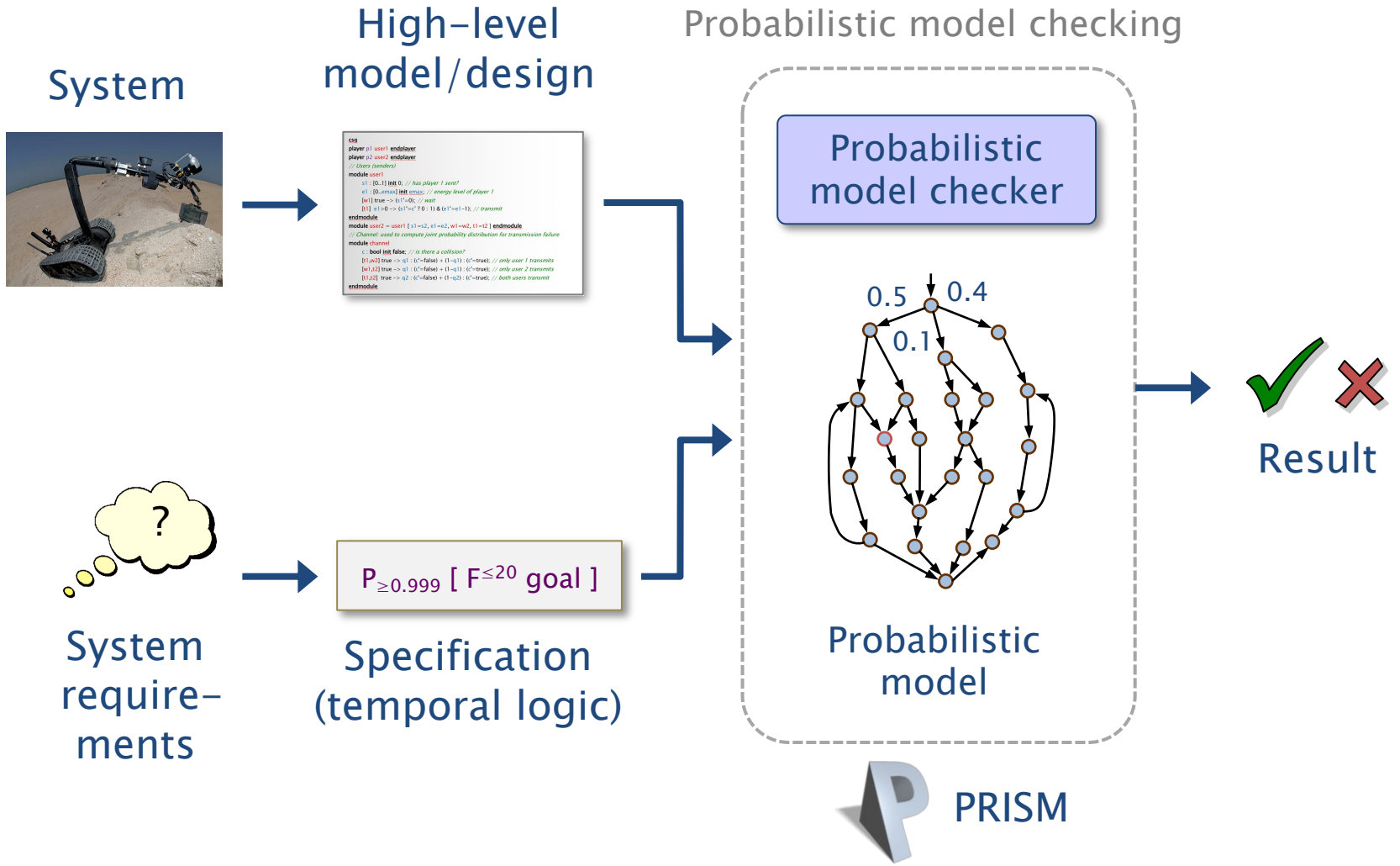
$$\textit{trigger} \rightarrow P_{\geq 0.999}\,[\,\Diamond^{\leq 2}\,\textit{deploy}\,]$$

- Key strengths: exhaustive + numeric analysis
  - often subtle interplay between probability + nondeterminism

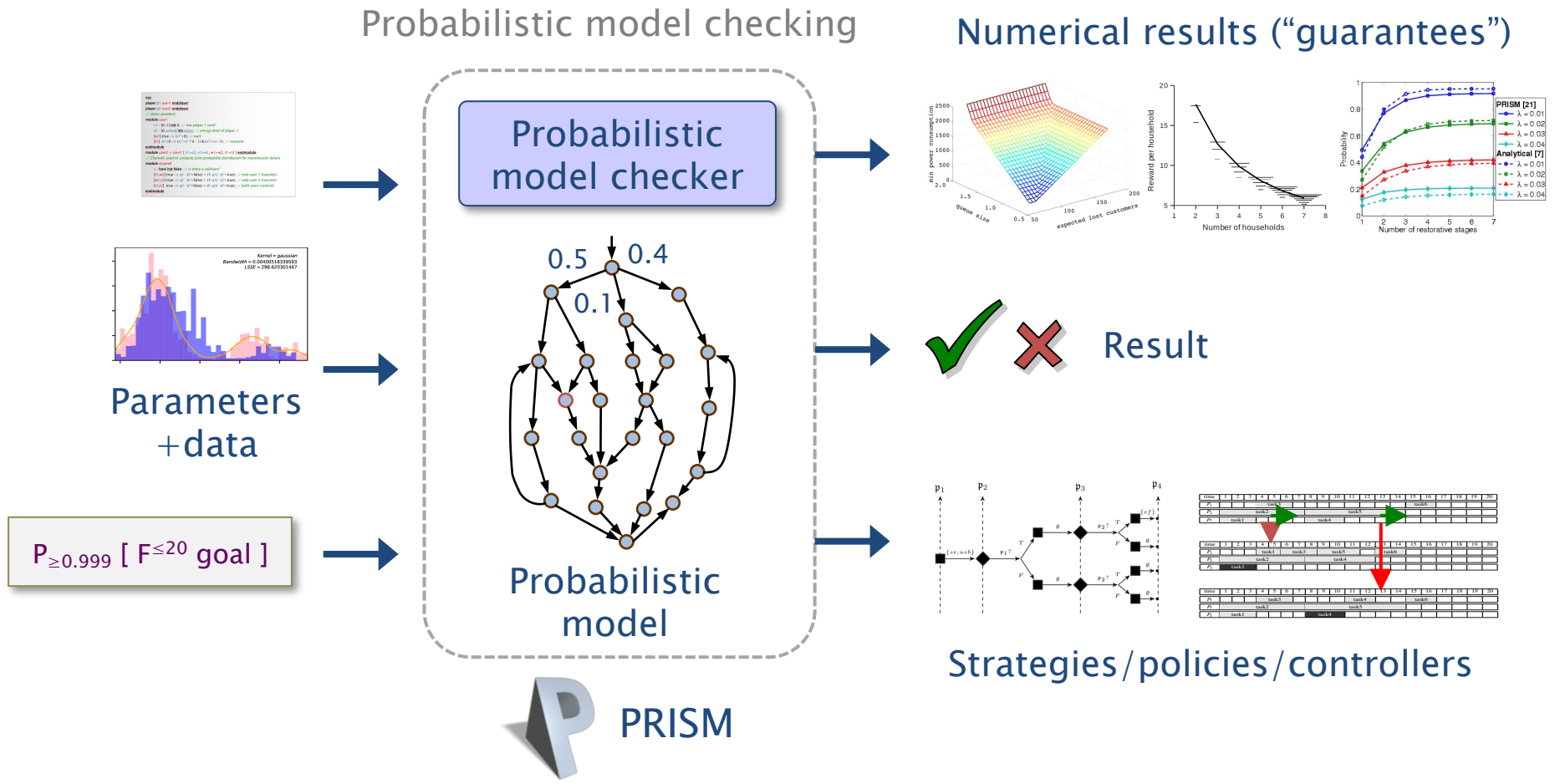


- Applications to:
  - airbag design, satellite reliability, pacemaker designs, communication/network protocols, computer security

Probabilistic model checking (PMC)

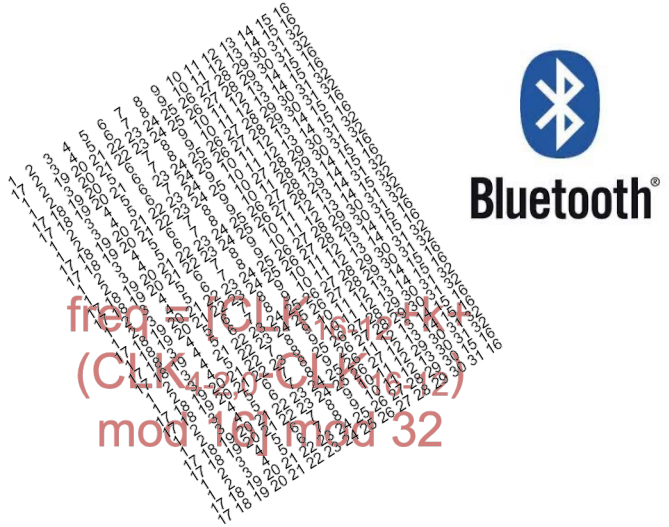


Probabilistic model checking

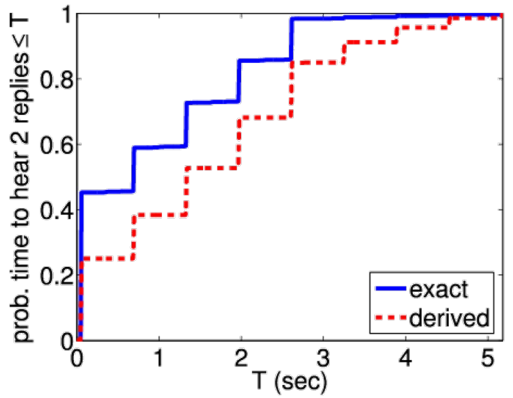


Example: Bluetooth

- Device discovery between a pair of Bluetooth devices
  - performance guarantees essential for this phase
- Complex discovery process
  - two asynchronous 28-bit clocks
  - pseudo-random hopping between 32 frequencies
  - random waiting scheme to avoid collisions



- Probabilistic model checking
  - worst-case expected time and probability for successful discovery
  - Markov chains with 17,179,869,184 initial configurations
  - exhaustive numerical analysis via symbolic model checking
  - highlights flaws in a simpler, analytic analysis

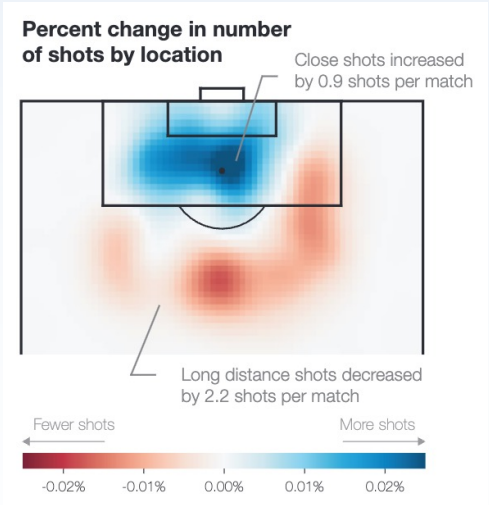


Diverse applications of PRISM

- Cloud computing
  - live migration of VMs
  - plan optimisation for performance guarantees

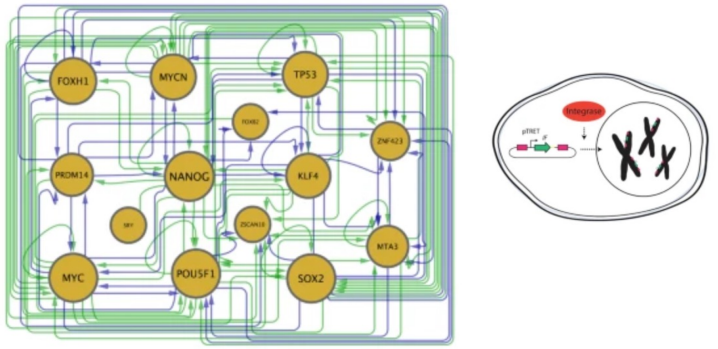
[Kikuchi/Matsumoto (Fujitsu), CLOUD'11] (Best paper)

- Football tactics ⚽
  - team strategies learnt from data
  - tactical efficiency analysed via probabilistic model checking



[Van Roy et al., JAIR'23, MIT-SSAC'24]

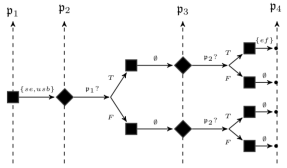
- Human-cell conversion
  - for disease models, gene therapies
  - design tool for optimisation and prediction, based on model checking

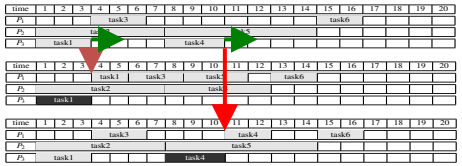


[Jung et al., Nature Communications'21]

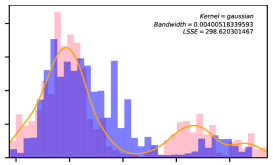
Trends in probabilistic model checking

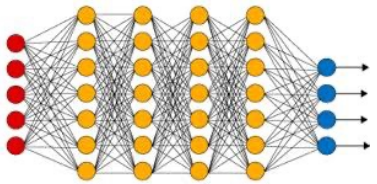
- From verification problems to control/synthesis
  - "correct-by-construction" from temporal logic specifications





- Increasing use/integration of learning
  - either to support modelling/verification
  - or deployed within the systems being verified





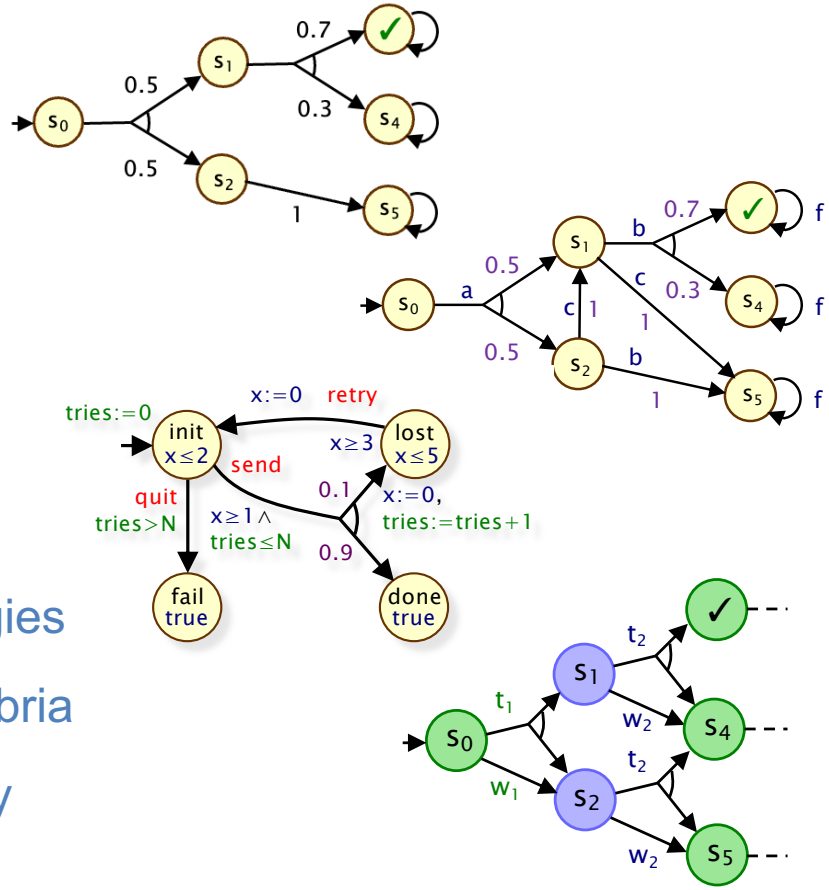
- Increasingly expressive/powerful classes of model
  - real-time, partial observability, epistemic uncertainty, multi-agent, …
  - leading to ever widening range of application domains

CTMC, CSG, DTMC, LTS, MDP, POMDP, POPTA, PTA, STPG, SMG, TPTG, IDTMC, IMDP

A zoo of probabilistic models

- Increasing variety (and complexity) of probabilistic models supported

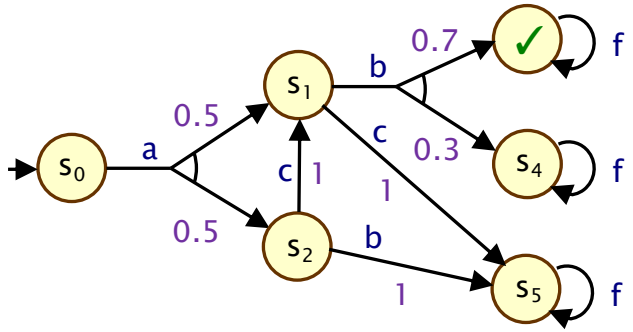
• discrete-time Markov chains	
• probabilistic automata	+ concurrency
• continuous-time Markov chains	+ exponential delays
• Markov decision processes (MDPs)	+ policies / control
• probabilistic timed automata	+ real-time clocks
• partially observable MDPs	+ observability
• stochastic multi-player games	+ multi-agent & strategies
• concurrent stochastic games	+ concurrency & equilibria
• interval Markov chains & MDPs	+ epistemic uncertainty



A zoo of probabilistic models

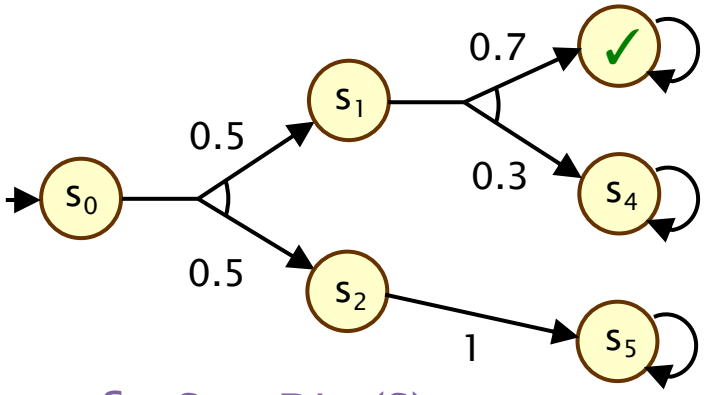
- Increasing variety (and complexity) of probabilistic models supported

discrete-time Markov chains	
probabilistic automata	+ concurrency
continuous-time Markov chains	+ exponential delays
Markov decision processes (MDPs)	+ policies / control
probabilistic timed automata	+ real-time clocks
partially observable MDPs	+ observability
stochastic multi-player games	+ multi-agent & strategies
concurrent stochastic games	+ concurrency & equilibria
interval Markov chains & MDPs	+ epistemic uncertainty



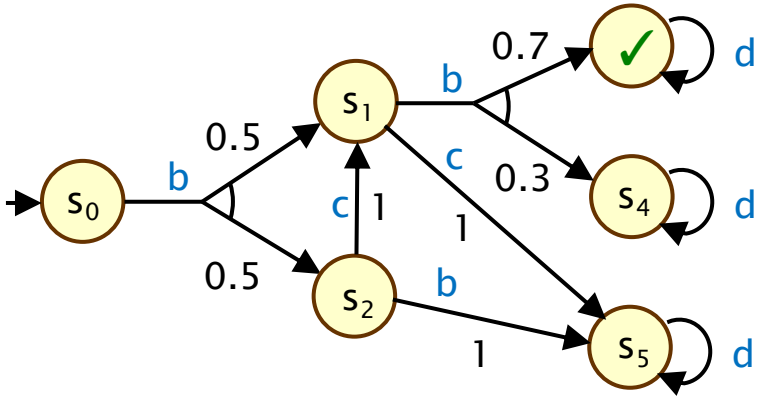
Probabilistic models

- Discrete-time Markov chains (DTMCs)
  - finite state space + discrete probabilities
  - core property: probabilistic reachability $Pr_s(F✓)$



$\delta : S \to Dist(S)$

- Markov decision processes (MDPs)
  - policies (or strategies) σ resolve actions based on history
  - e.g.: $P_{max=?}[F✓] = \sup_\sigma Pr_s^\sigma(F✓)$
  - what is the maximum probability of reaching ✓ achievable by any policy σ?

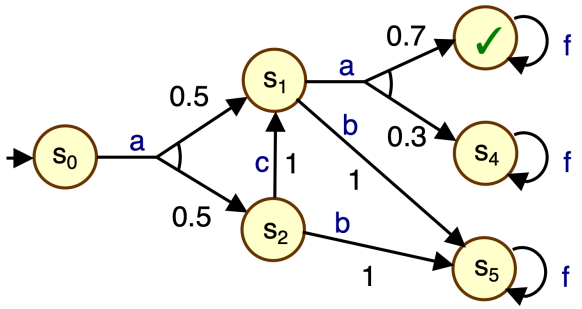


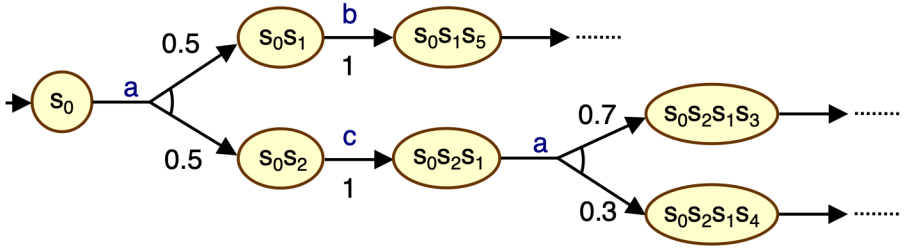
$\delta : S \times A \to Dist(S)$

- Models for probabilistic model checking:
  - mostly finite-state
  - mostly known in full

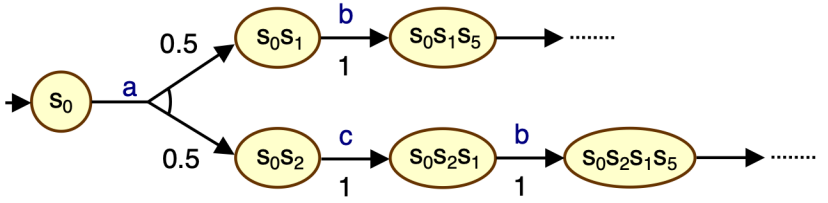
MDPs and policies

- Policies for an MDP differ in the use of memory and randomisation
  - each yields an induced Markov chain

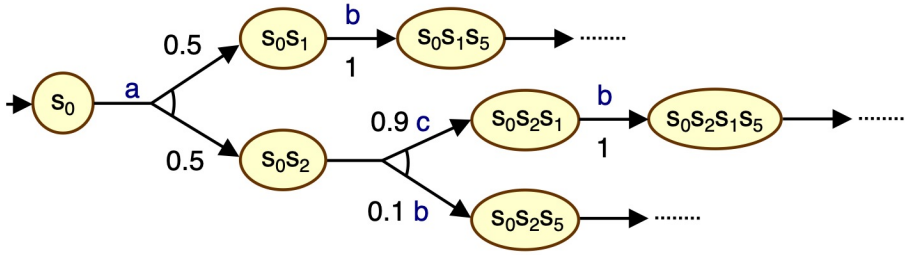




(finite-memory, deterministic)



(memoryless, deterministic)



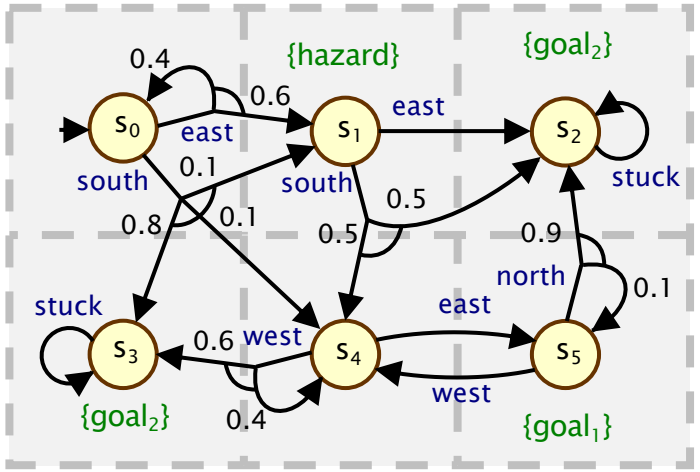
(memoryless, randomised)

Temporal logic

Temporal logic

- Formal specification of desired/required behaviour
  - formal language for quantitative guarantees
- 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
    $P_{=?}$ [F $goal_1$]
    $P_{max=?}$ [F $goal_1$]

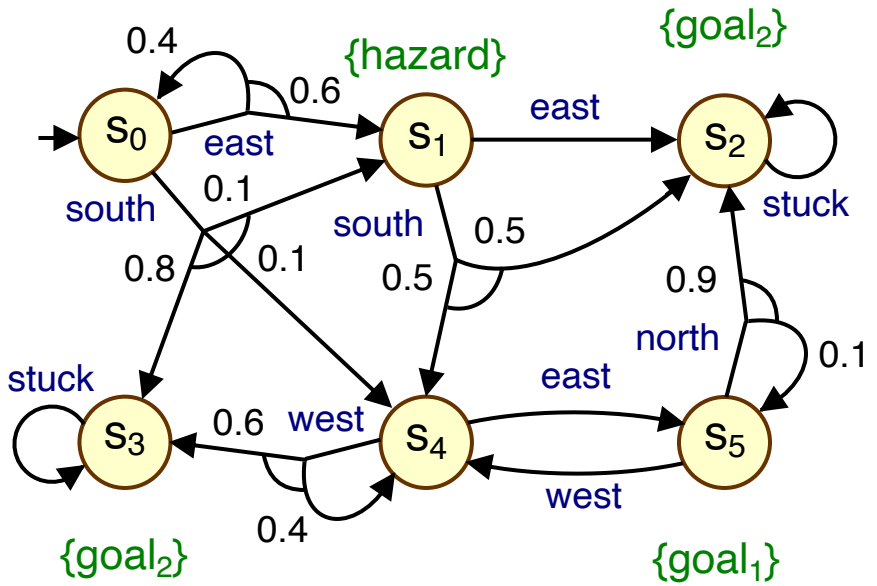
Example MDP (robot navigation)



- Extensions
  - richer temporal specs (LTL), multi-objective, costs/rewards, …

Correctness by construction

- Synthesise correct-by-construction controllers/policies/plans
  - based on temporal logic specifications (probabilistic guarantees)
  - verification vs synthesis of MDP policies

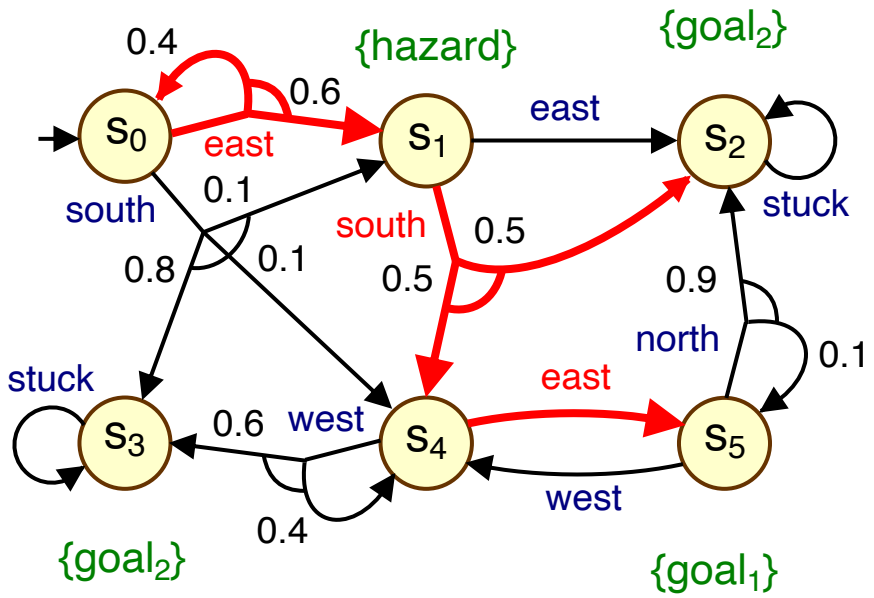


Can we guarantee reaching $goal_1$ with probability 0.5?

$P_{\geq 0.5}$ [F $goal_1$]

Correctness by construction

- Synthesise correct-by-construction controllers/policies/plans
  - based on temporal logic specifications (probabilistic guarantees)
  - verification vs synthesis of MDP policies



Can we guarantee reaching $goal_1$ with probability 0.5?

$P_{\geq 0.5}$ [F $goal_1$]

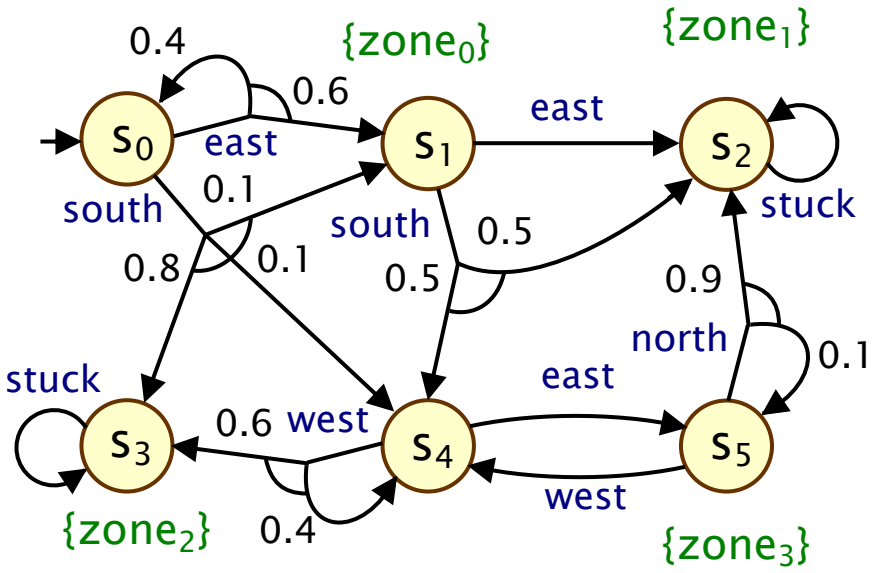
How do we maximise the probability of reaching $goal_1$?

$P_{max=?}$ [F $goal_1$]

(optimal policy is deterministic and memoryless)

Correctness by construction

- Synthesise correct-by-construction controllers/policies/plans
  - based on temporal logic specifications (probabilistic guarantees)
  - verification vs synthesis of MDP policies



With high probability, complete the task
"inspect zones 3 then 1, without passing through zone 0"
whilst always remaining close to the charging dock.

$P_{>0.99}$ [$\neg zone_0$ U ($zone_3 \wedge$ (F $zone_1$))]
$\wedge \forall G\ P_{>0.95}$ [$F^{\leq 100}$ $zone_2$)]

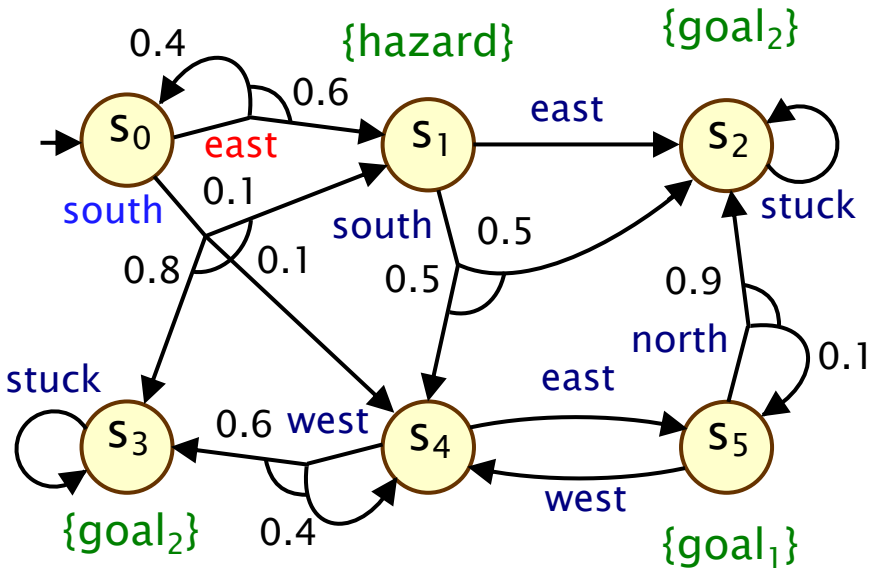
Linear temporal logic (LTL)

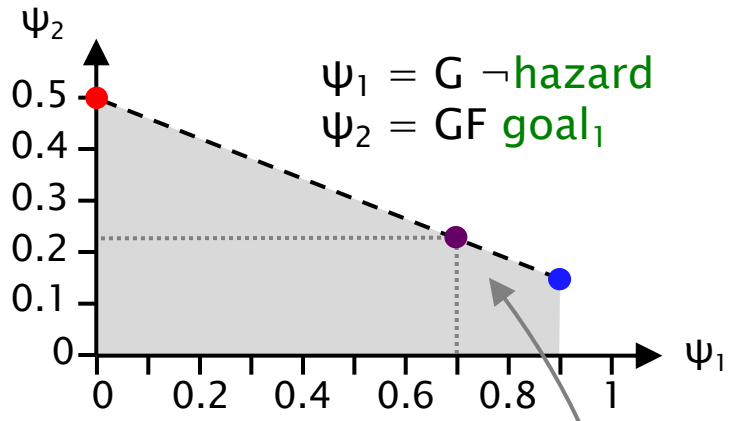
- LTL (linear temporal logic) syntax:
  - $\psi ::= \text{true} \mid a \mid \psi \wedge \psi \mid \neg\psi \mid X\,\psi \mid \psi\ U\ \psi \mid F\,\psi \mid G\,\psi$
- Propositional logic + temporal operators:
  - a is an atomic proposition (labelling a state)
  - $X\,\psi$ means "ψ is true in the next 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:
  - ◯ (next), ◇ (eventually), □ (always) , U (until)

Linear temporal logic (LTL)

- LTL (linear temporal logic) 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"
- Example: robot task specifications in LTL
  - e.g. $P_{>0.7}\,[\,(G\neg hazard) \wedge (GF\,goal_1)\,]$ – "the probability of avoiding hazard and visiting $goal_1$ infinitely often is > 0.7"
  - e.g. $P_{max=?}\,[\,\neg zone_3\,U\,(zone_1 \wedge (F\,zone_4))\,]$ – "max. probability of patrolling zone 1 (whilst avoiding zone 3) then zone 4?"

Multi-objective specifications





randomised, (finite-memory) optimal policy

- Achievability query
  - $P_{\geq 0.7}$ [G ¬hazard] $\wedge$ $P_{\geq 0.2}$ [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 $goal_1$] ?

More temporal logic

- Costs & rewards
  - i.e., accrued values assigned to model states or transitions
  - e.g., $R^{time}_{min=?}$ [$\neg zone_3$ U ($zone_1 \wedge$ (F $zone_4$))]
  - minimise expected time to patrol zone 1 (whilst avoiding zone 3) then zone 4?
- Nested (branching-time) queries
  - e.g. $R^{bat}_{min=?}$ [$P_{\geq 0.99}$ [$F^{\leq 10}$ base] U ($zone_1 \wedge$ (F $zone_4$))]
  - "minimise expected battery usage to visit zones 1 then 4, whilst (initially) ensuring the base can always be reliably reached
- And more
  - cost-bounded, conditional probabilities, quantiles
  - metric temporal logic, signal temporal logic, …

Benefits of temporal logic

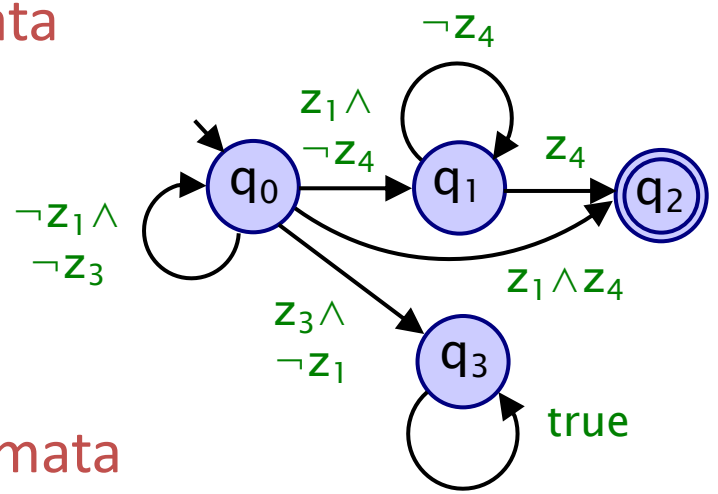
- Unambiguous, flexible, tractable behavioural specification
  - broad range of quantitative properties expressible
- (Probabilistic) guarantees on safety, performance, etc.
  - meaningful properties: event probabilities, time, energy,…

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

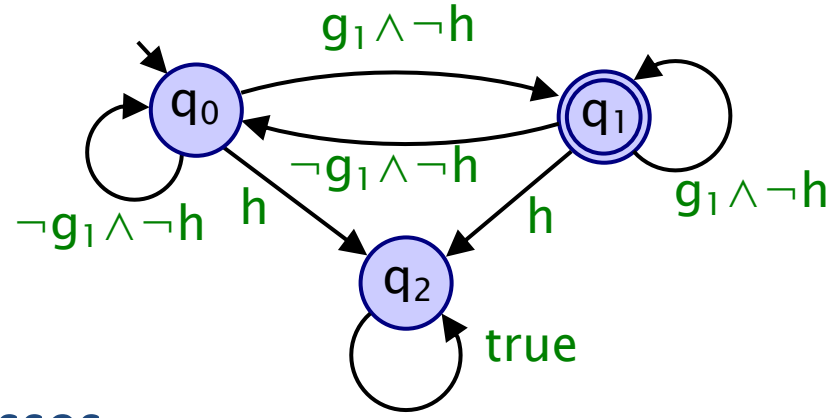
  - compare to: ad-hoc reward structures, e.g. with discounting
  - caveat: accuracy of model (and its solution)
- Efficient LTL-to-automata translation
  - optimal (finite-memory) policy synthesis (via product MDP)
  - correctness monitoring / shielding
  - task progress metrics

LTL & automata

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

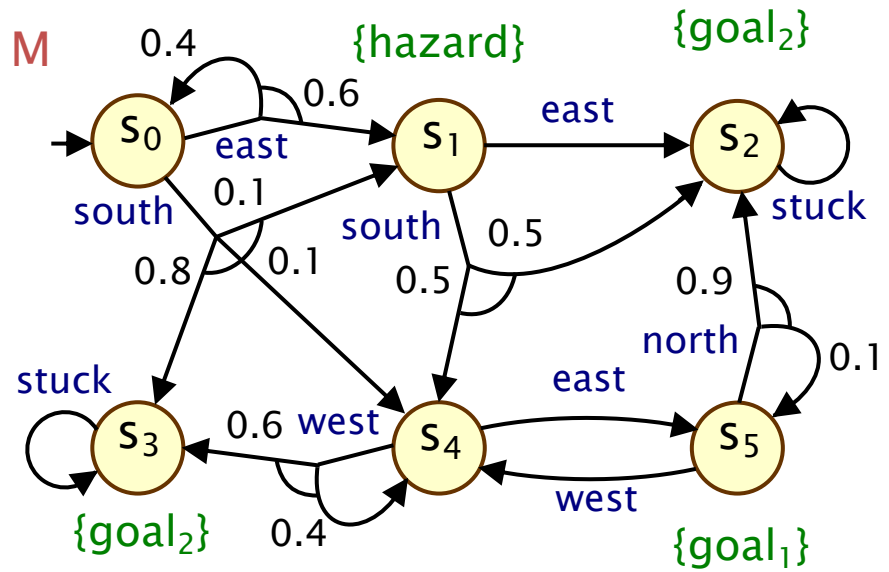


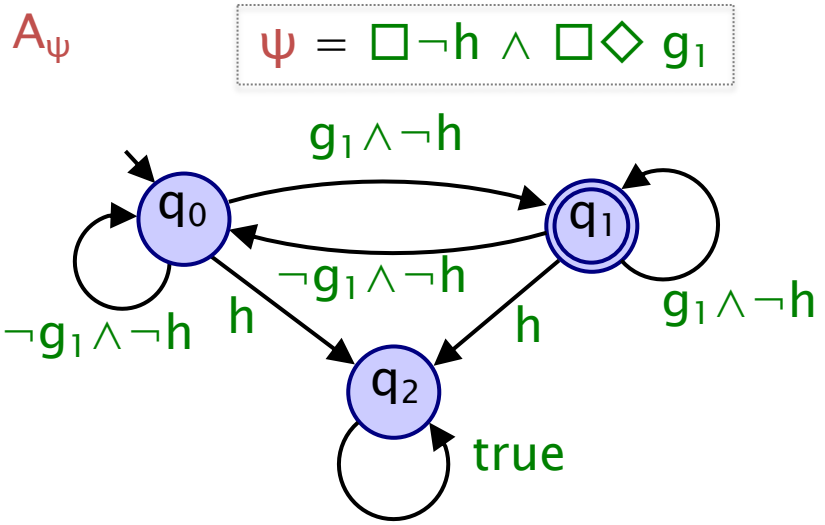
- Full LTL: e.g. (deterministic) Rabin/Buchi automata
  - $G\neg hazard \wedge GF\ goal_1$

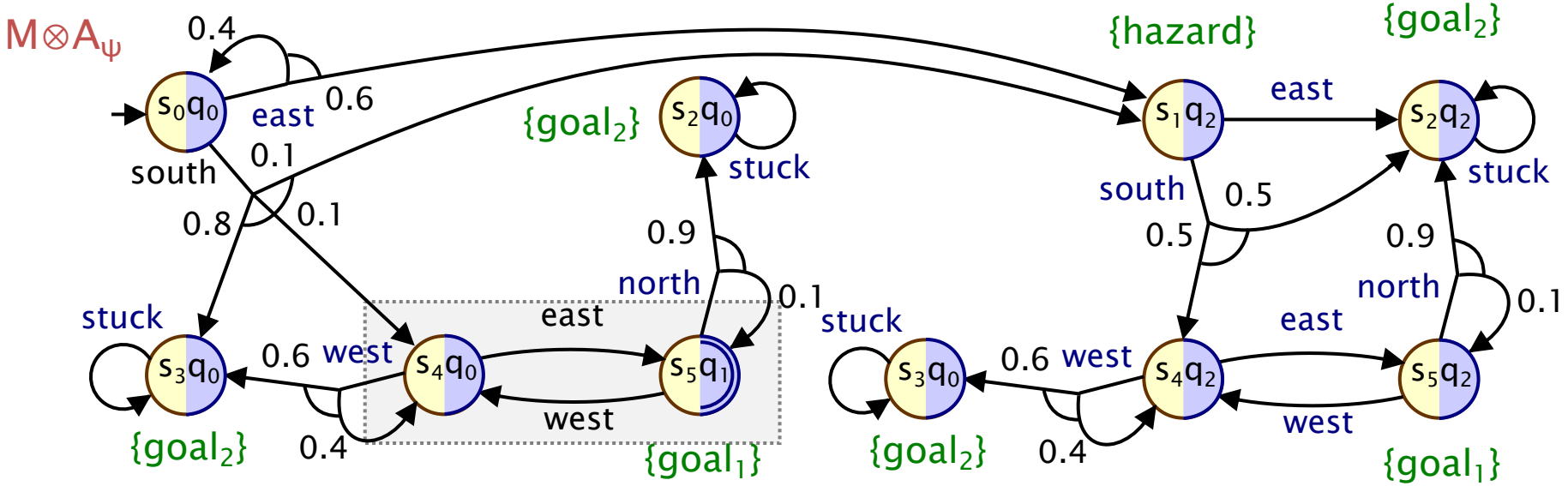


- Many other useful LTL/automata subclasses…

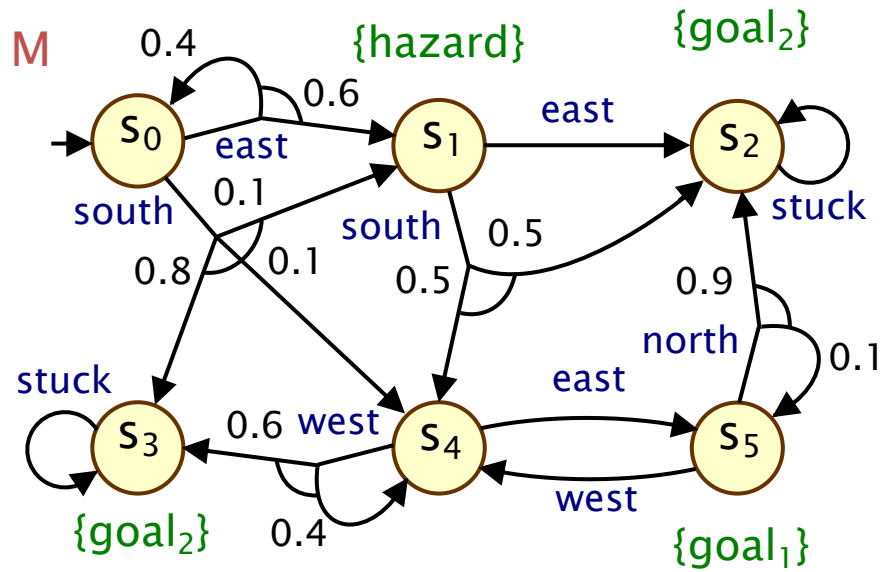
LTL model checking via product MDP

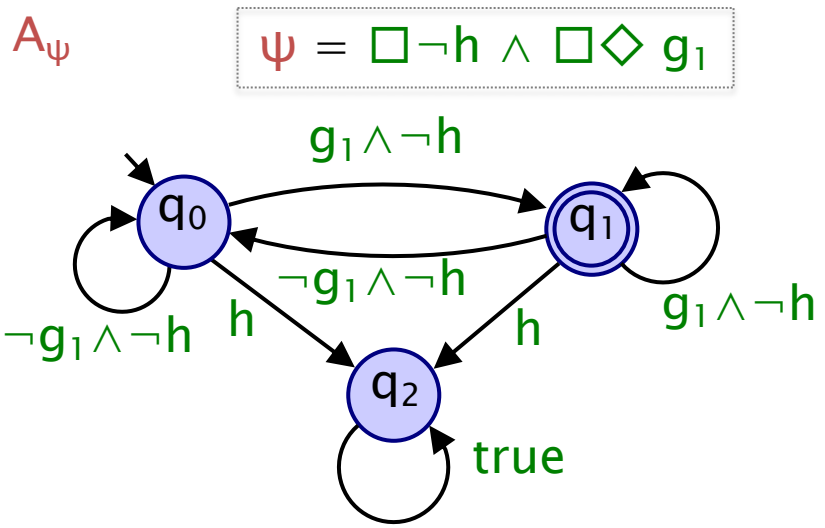


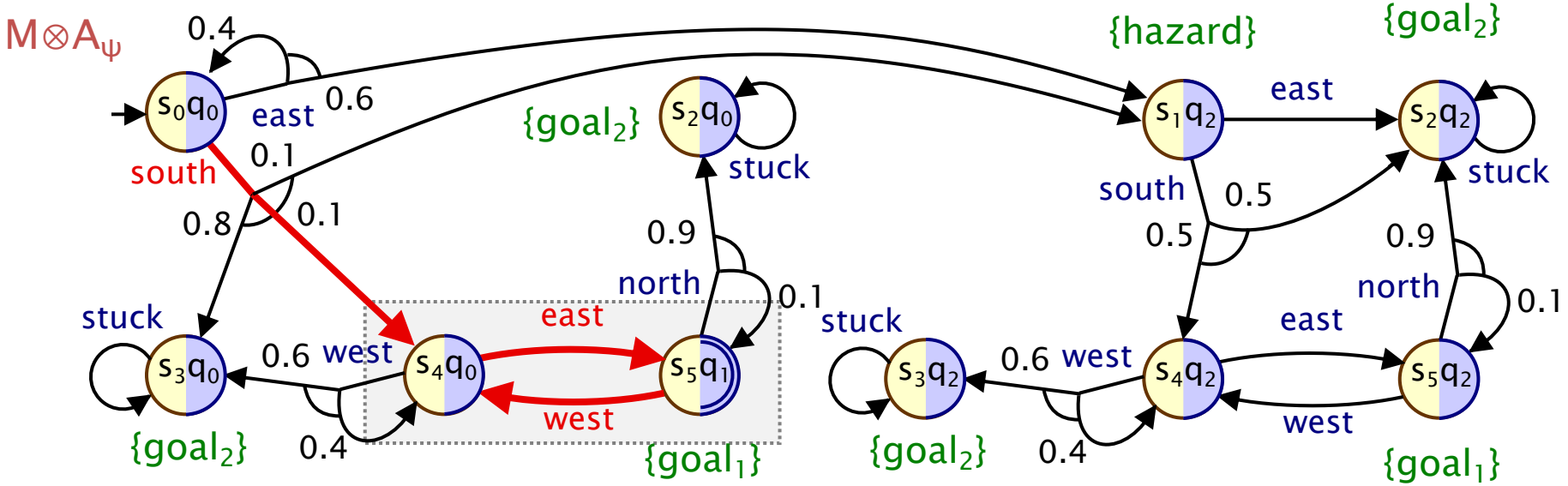




LTL model checking via product MDP

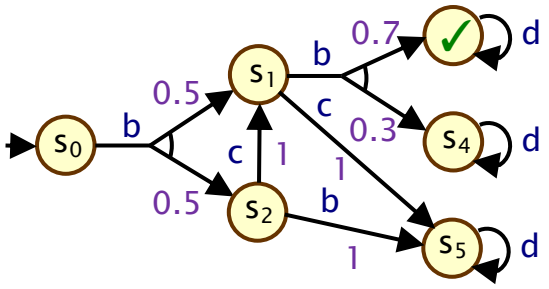






Verification techniques

- 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



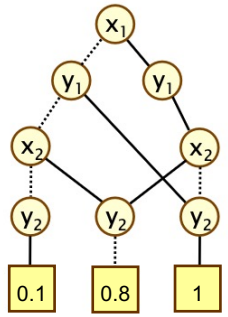
- Example (MDPs):
  - max. probability of reaching ✓
  - values $p(s) = \sup_\sigma Pr_s^\sigma(F\checkmark)$ are the least fixed point of:

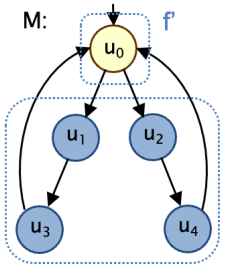
$$p(s) = \begin{cases} 1 & \text{if } s \models \checkmark \\ \max_a \Sigma_{s'} \delta(s,a)(s') \cdot p(s') & \text{otherwise} \end{cases}$$

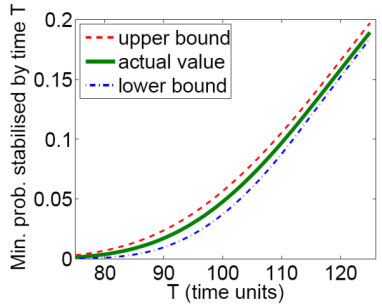
- But: VI has known accuracy and convergence issues
  - interval iteration, sound VI, optimistic VI
  - separate convergence from above and below

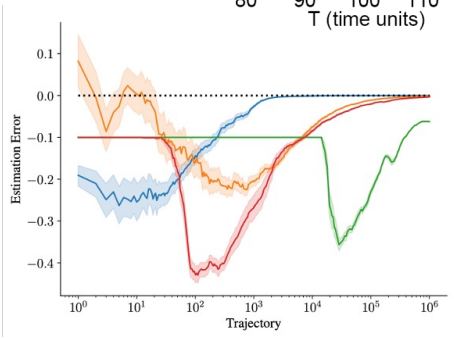
Scalability & efficiency

- Scalability & efficiency are always key challenges
  - many approaches investigated…
- Symbolic probabilistic model checking
  - i.e., (multi-terminal) binary decision diagrams
- Model reductions
  - bisimulation minimisation
  - abstraction + sound bounds (property driven)
- Sampling (simulation) based methods
  - statistical model checking, PAC guarantees, heuristics, …
- Trade-off: scalability/efficiency vs. accuracy/guarantees
  - spectrum of “correctness” : exact, floating-point correct, ε-correct, probably ε-correct, …



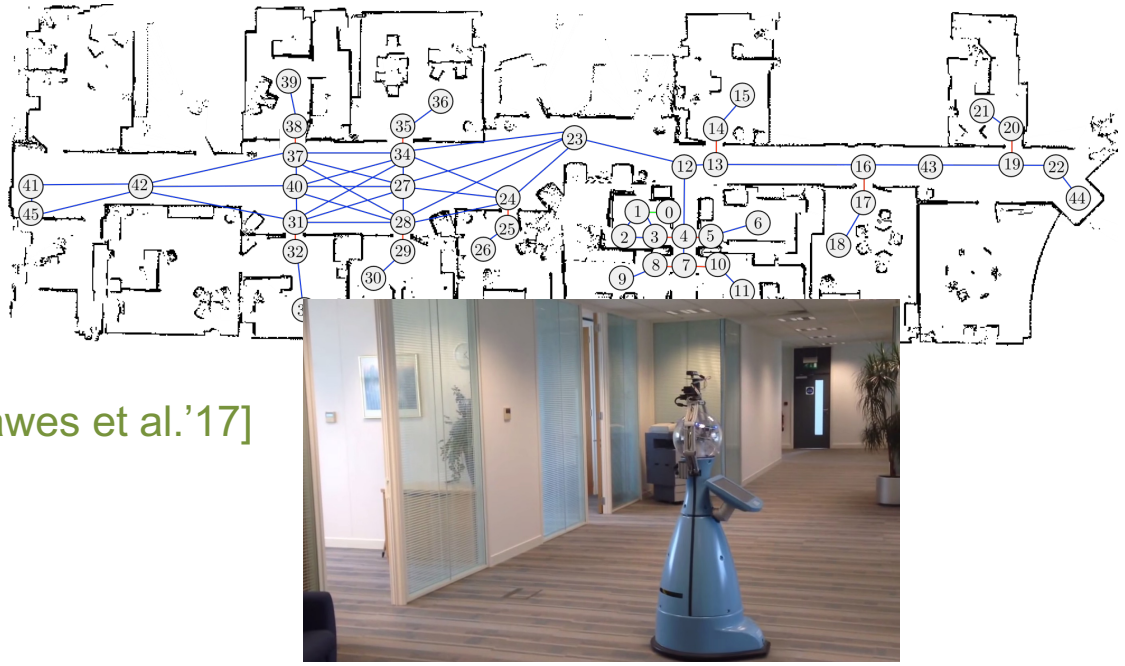






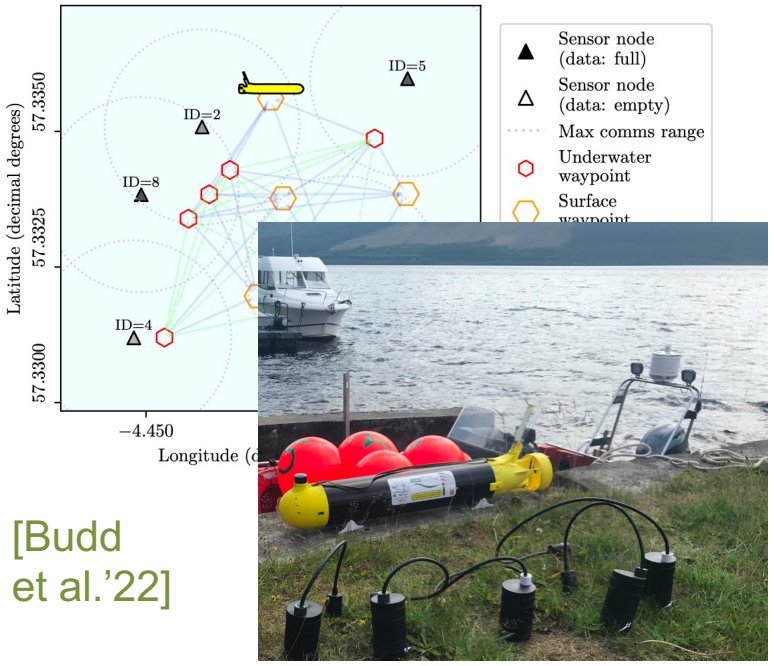
Example: Robot deployments

- Mobile robots in offices/care homes
  - Convert MDP policies to navigation controllers
  - ROS module based on PRISM
  - 100s of hrs of autonomous deployment



[Hawes et al.'17]

- Underwater autonomous vehicles
  - efficient/reliable retrieval of data from sensor networks
  - PRISM-generated control policies outperform hand-designed ones



[Budd et al.'22]

Overview

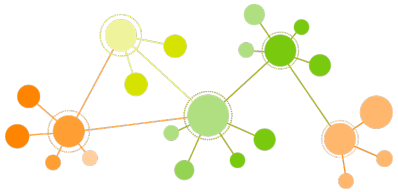
- Sequential decision making under uncertainty
- Formal verification: probabilistic model checking
  - key ideas and example applications
  - probabilistic models
  - temporal logic & automata

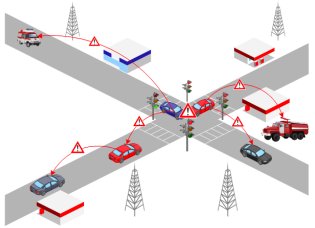
- Multi-agent decision making
  - stochastic games
- Data-driven models for decision making
  - robustness under epistemic uncertainty
- Neuro-symbolic decision making

Multi-agent decision making

Stochastic multi-agent systems

- How do we verify/control stochastic systems with…
  - multiple agents acting autonomous and concurrently
  - competitive or collaborative behaviour between agents, possibly with differing goals
  - learnt components for e.g. control/perception





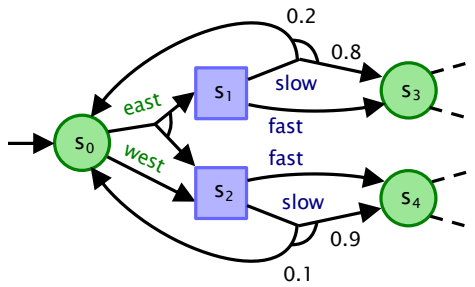
- Applications:
  - distributed protocols for consensus/security
  - multi-robot systems
  - autonomous vehicles

- Probabilistic model checking
  - with stochastic multi-player games

A zoo of probabilistic models

- Increasing variety (and complexity) of probabilistic models supported

discrete-time Markov chains	
probabilistic automata	+ concurrency
continuous-time Markov chains	+ exponential delays
Markov decision processes (MDPs)	+ policies / control
probabilistic timed automata	+ real-time clocks
partially observable MDPs	+ observability
stochastic multi-player games	+ multi-agent & strategies
concurrent stochastic games	+ concurrency & equilibria
interval Markov chains & MDPs	+ epistemic uncertainty



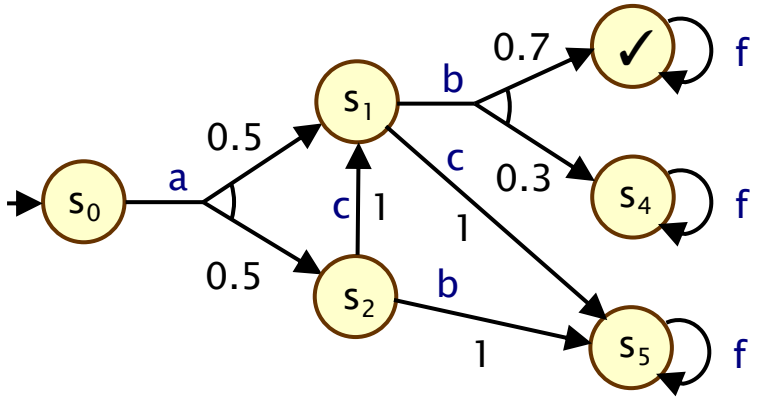
Multiple players with differing strategies and objectives

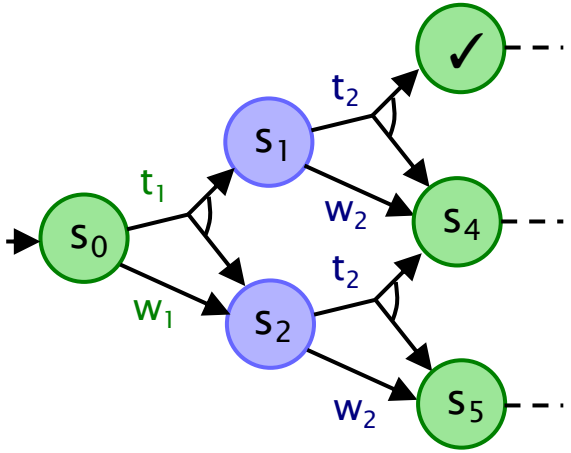
Stochastic multi-player games

- (Turn-based) stochastic multi-player games
  - strategies + probability + multiple players
  - player i controls subset of states S_i

Markov decision processes (MDPs)

Turn-based stochastic games (TSGs)





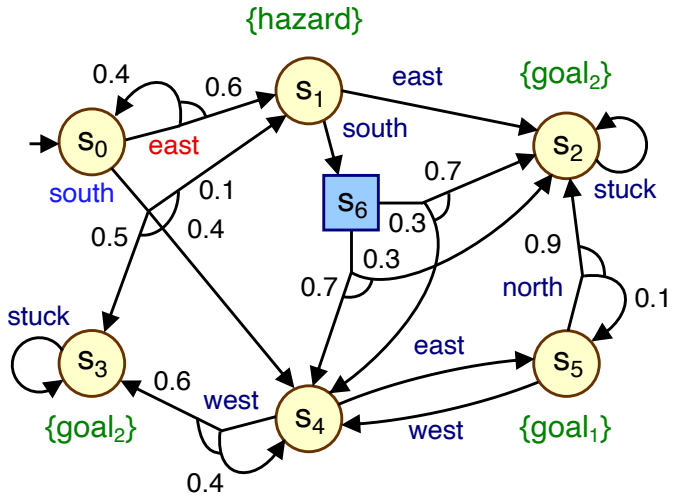
$\delta : S \times A \rightarrow Dist(S)$

$S = S_1 \uplus \ldots \uplus S_n$

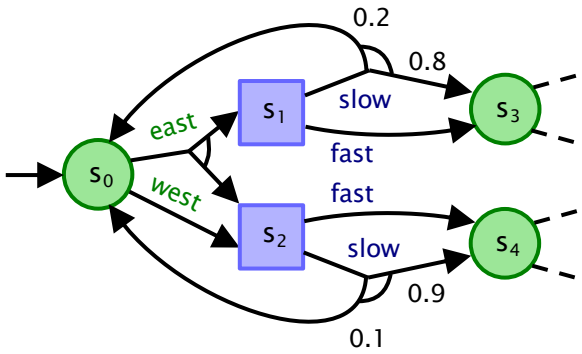
Modelling with turn-based games

- Turn-based stochastic games well suited to some (but not all) scenarios

Uncontrollable/unknown navigation interference



Shared autonomy: human-robot control



Property specification: rPATL

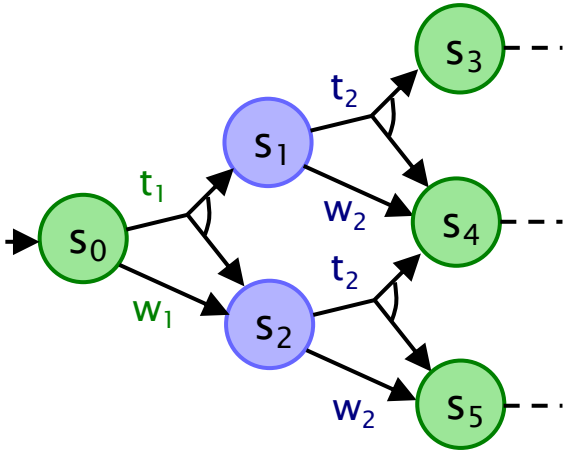
- rPATL (reward probabilistic alternating temporal logic)
  - zero-sum, branching-time temporal logic for stochastic games
  - coalition operator $\langle\langle C \rangle\rangle$ of ATL
    + probabilistic (P) and reward (R) operators
- Example:
  - $\langle\langle \{robot_1, robot_3\} \rangle\rangle P_{max=?} [F (goal_1 \vee goal_3)]$
  - "what strategies for robots 1 and 3 maximise the probability of reaching their goal locations, regardless of the strategies of other robots"

Can be seen as a mixture of control and verification

- Other additions:
  - (co-safe) linear temporal logic
    $\neg zone_3 \; U \; (room_1 \wedge (F \; room_4 \wedge F \; room_5)$
  - nested specifications
    $\langle\langle \{robot_1, robot_3\} \rangle\rangle R_{min=?} [$
    $\langle\langle \{robot_1\} \rangle\rangle P_{\geq 0.99} [F^{\leq 10} \; base]$
    $U \; (zone_1 \wedge (F \; zone_4))]$
    "minimise expected time for joint task, while ensuring base reliably reached"

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:

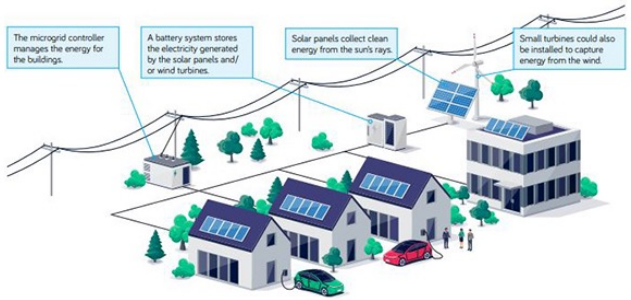
$$p(s) = \begin{cases} 1 & \text{if } s \models ✓ \\ \max_a \Sigma_{s'} \delta(s,a)(s') \cdot p(s') & \text{if } s \not\models ✓ \text{ and } s \in S_1 \\ \min_a \Sigma_{s'} \delta(s,a)(s') \cdot p(s') & \text{if } s \not\models ✓ \text{ and } s \in S_2 \end{cases}$$

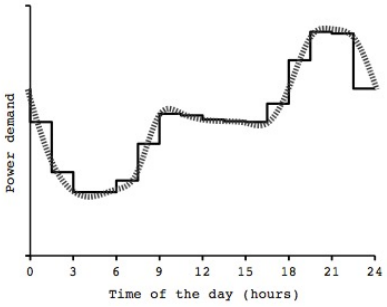
  - and more: graph-algorithms, sequences of fixed points, …

- Implementation
  - symbolic (BDD-based) version also developed
  - big gains on some models
  - also benefits for strategy compactness

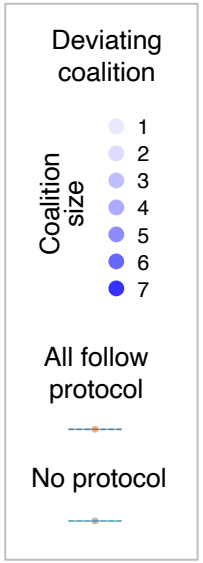
Example: Energy protocols

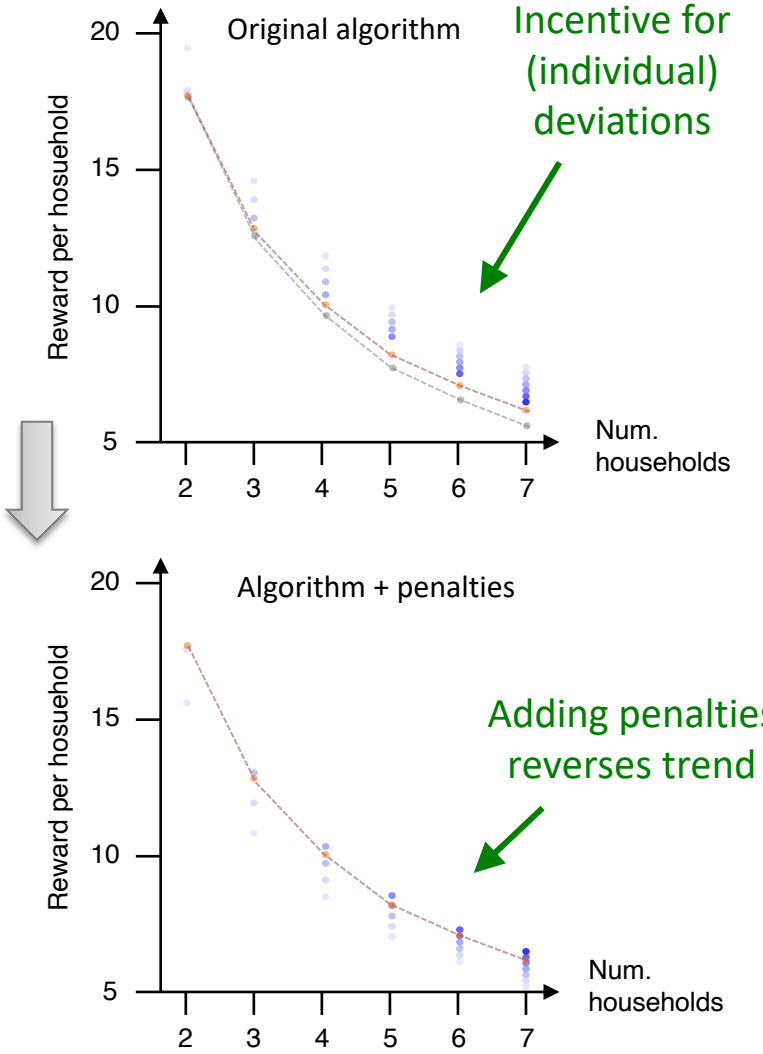
- Demand management protocol for microgrids
  - randomised back-off to minimise peaks
- Stochastic game model checking
  - allow users to collaboratively cheat (ignore protocol)
  - models of up to ~6 million states
  - exposes protocol weakness
    (incentive for clients to act selfishly)
  - propose/verify simple fix using penalties





PRISM-games

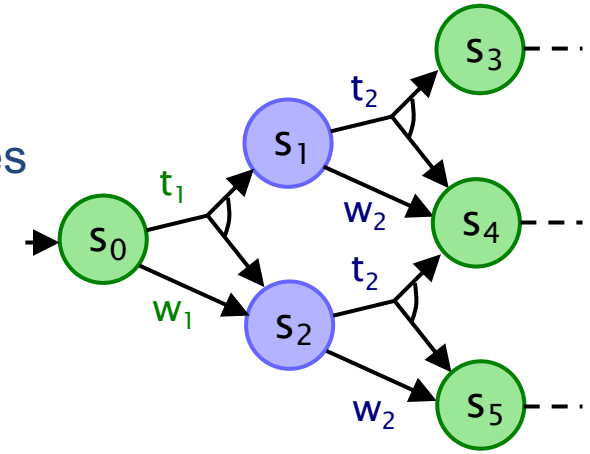


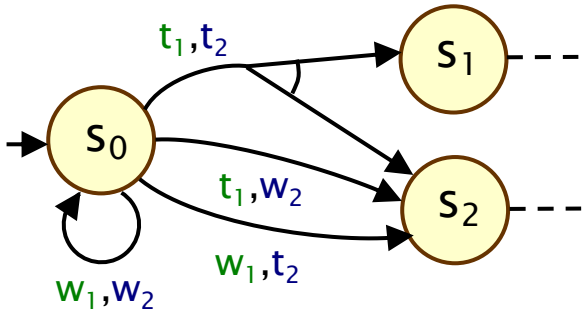


Concurrent stochastic games

- Need a more realistic model of components operating concurrently
- Concurrent stochastic games (CSGs)
  - (also known as Markov games, multi-agent MDPs)
  - players choose actions concurrently & independently
  - jointly determines (probabilistic) successor state

Turn-based stochastic games (TSGs)

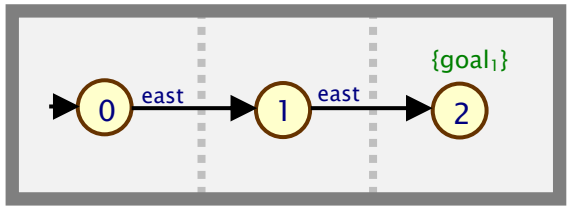


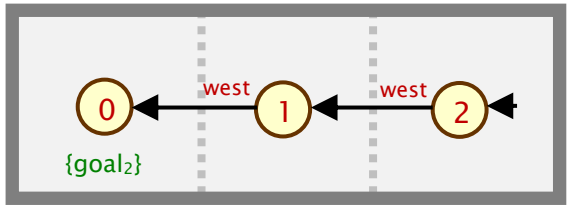


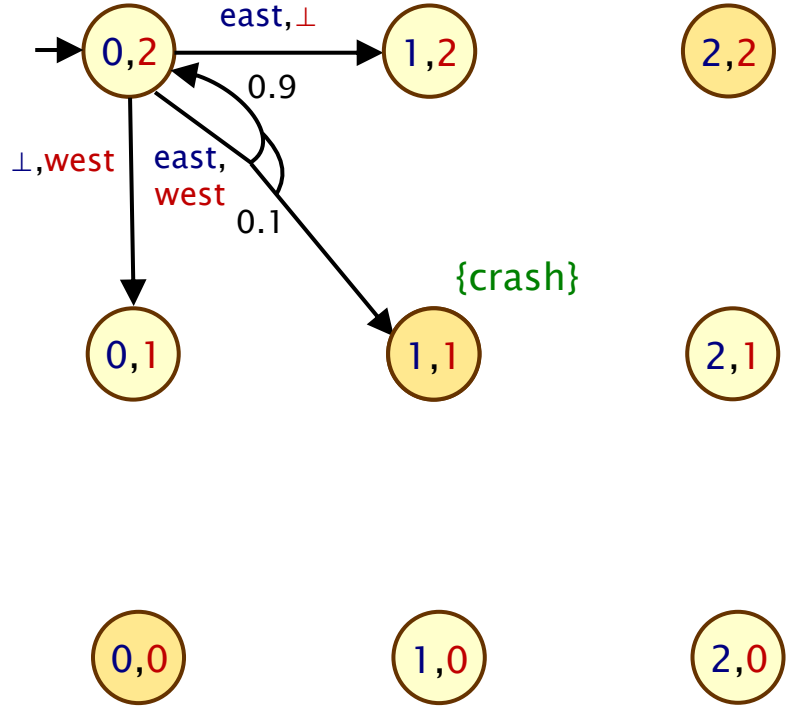
Concurrent stochastic games (CSGs)

$\delta : S \times (A_1 \cup \{\bot\}) \times \ldots \times (A_n \cup \{\bot\}) \rightarrow \text{Dist}(S)$

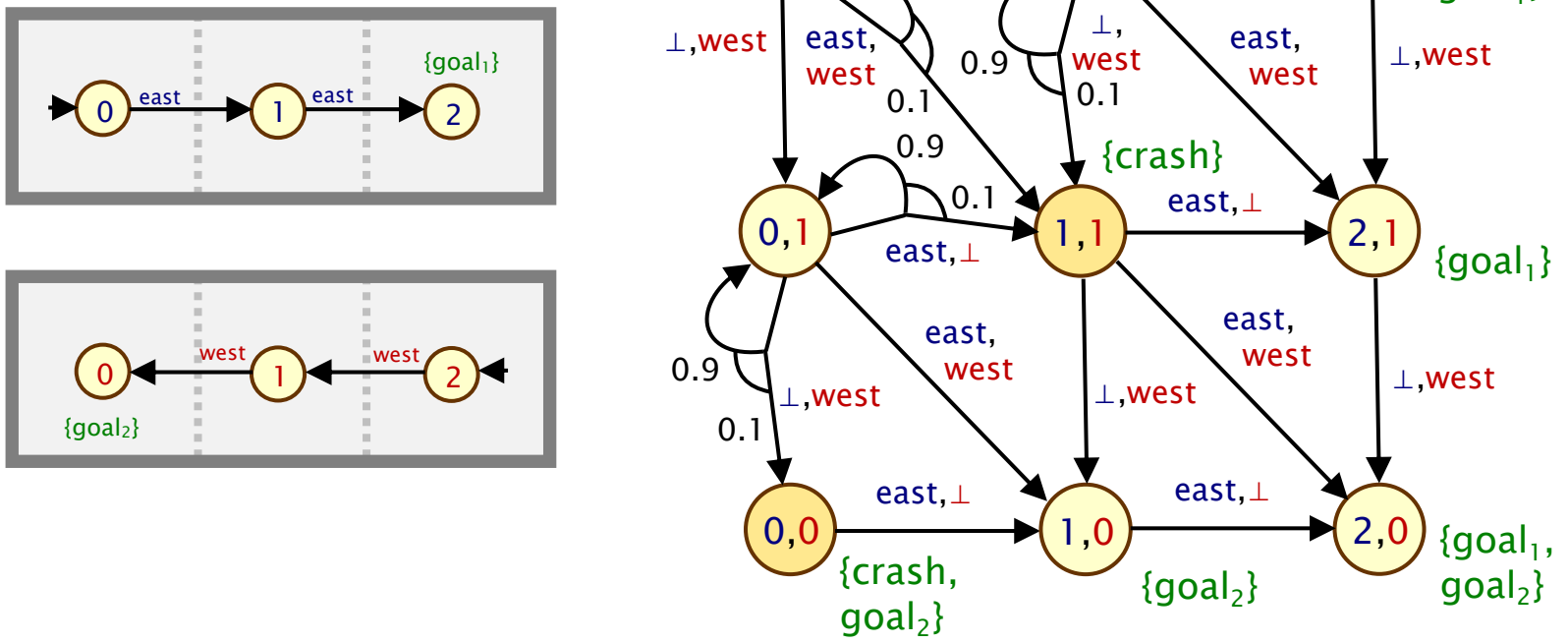
CSG for 2 robots on a 3x1 grid





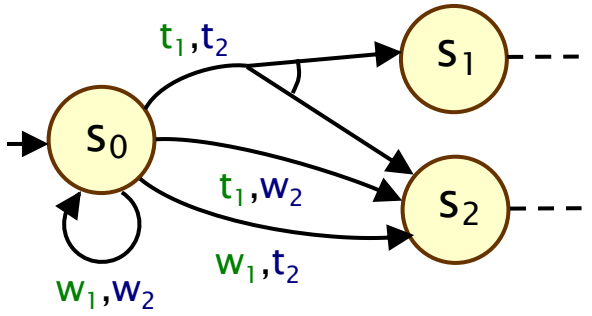


CSG for 2 robots on a 3x1 grid



rPATL model checking for CSGs

- Same overall rPATL model checking algorithm
  - key ingredient is now solving (zero-sum) 2-player CSGs (PSPACE)
  - note that optimal strategies are now randomised



- 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:

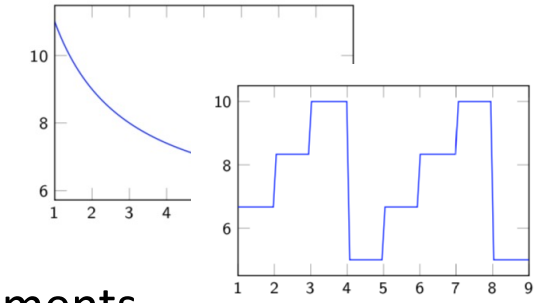
$$p(s) = \begin{cases} 1 & \text{if } s \models \checkmark \\ val(Z) & \text{if } s \not\models \checkmark \end{cases}$$

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

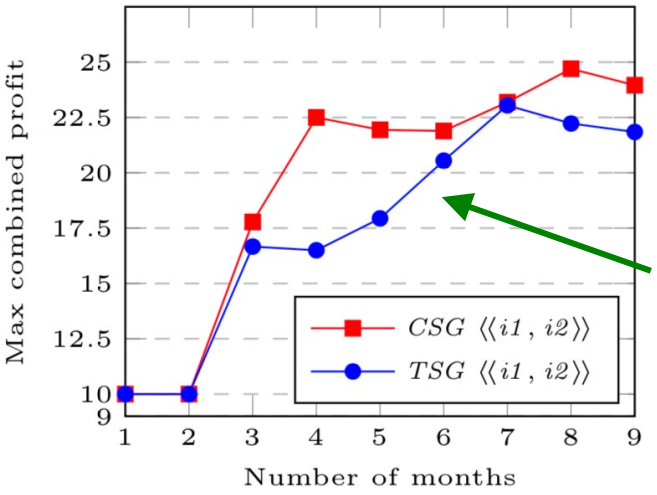
- Implementation
  - matrix games solved as linear programs
    - (LP problem of size $|A|$)
  - required for every iteration/state
    - which is the main bottleneck
  - but we solve CSGs of ~3 million states

Example: Future markets investor

- 3-player CSG modelling interactions between:
  - stock market, evolves stochastically
  - two investors i_1, i_2 decide when to invest
  - market decides whether to bar investors
  - various profit models; reduced for simultaneous investments



- Investor strategy synthesis via rPATL model checking
  - $\langle\langle \text{investor}_1, \text{investor}_2 \rangle\rangle \, R^{\text{profit}_{1,2}}_{\text{max}=?} [\, F \, \text{finished}_{1,2} \,]$
  - non-trivial optimal (randomised) investment strategies
  - concurrent game (CSG) yields more realistic results (market has less observational power over investors)



Too pessimistic: unrealistic strategy for adversary

Equilibria-based properties

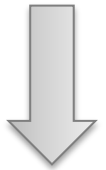
- Beyond zero-sum games:
  - players/components may have distinct objectives but which are not directly opposing (zero-sum)
- We use Nash equilibria (NE)
  - no incentive for any player to unilaterally change strategy
  - actually, we use ε-NE, which always exist for CSGs

$\sigma=(\sigma_{1,\dots},\sigma_n)$ is an ε-NE for objectives $X_1,\dots,X_n$ iff:
for all i : $E_s^{\sigma}(X_i) \geq \sup \{ E_s^{\sigma'}(X_i) \mid \sigma'=\sigma_{-i}[\sigma_i'] \text{ and } \sigma_i' \in \Sigma_i \} - \varepsilon$

- We extend rPATL model checking for CSGs
  - with social-welfare Nash equilibria (SWNE)
  - i.e., NE which also maximise the joint sum $E_s^{\sigma}(X_1) + \dots E_s^{\sigma}(X_n)$

Zero-sum properties

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

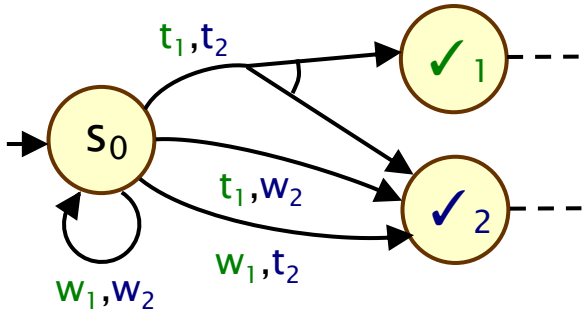


$\langle\langle robot_1{:}robot_2 \rangle\rangle_{max=?}$
$(P\, [\, F^{\leq k}\, goal_1\,] + P\, [F^{\leq k}\, goal_2])$

Equilibria-based properties (SWNE)

Model checking for Nash equilibria

- Model checking for CSGs with equilibria
  - needs solution of bimatrix games
  - (basic problem is EXPTIME)
  - strategies need history and randomisation



- Implementation
  - we adapt a known approach using labelled polytopes, and implement via SMT
  - optimisations: filtering of dominated strategies
  - solve CSGs of ~2 million states

- We further extend the value iteration approach:

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

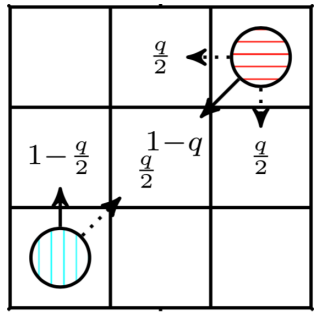
standard MDP analysis

bimatrix game

  - where Z_1 and Z_2 encode matrix games similar to before

Example: multi-robot coordination

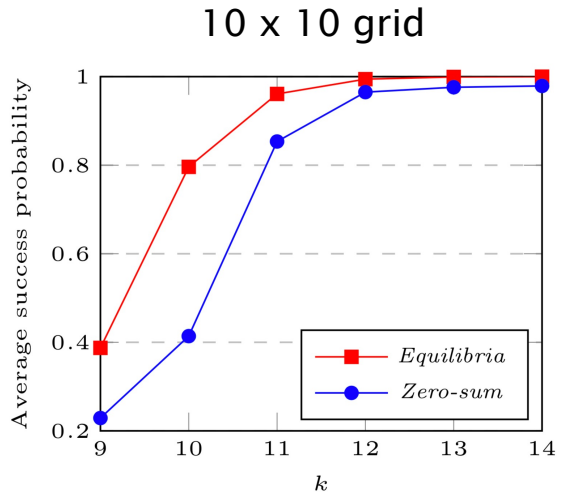
- 2 robots navigating an m x m gridworld
  - start at opposite corners, goals are to navigate to opposite corners
  - obstacles modelled stochastically



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

Collaboration helps: better performance from equilibria

ε-NE found typically have ε=0



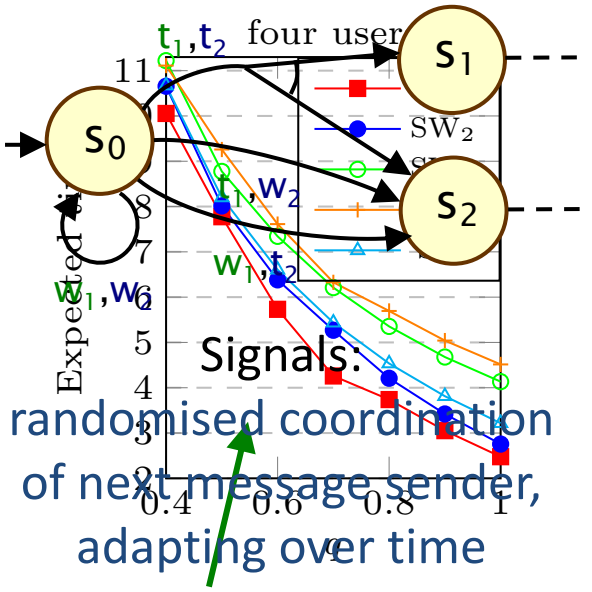
Faster and fairer equilibria

- Limitations of (social welfare) Nash equilibria for CSGs:
  1. can be computationally expensive, especially for >2 players
  2. social welfare optimality is not always equally beneficial to players

- Correlated equilibria
  - correlation: shared (probabilistic) signal + map to local strategies
  - synthesis: support enumeration + nonLP (Nash) -> LP (correlated)
  - experiments: much faster to synthesise (4-20x faster)

- Social fairness
  - alternative optimality criterion: minimise difference in objectives
  - applies to both Nash/correlated: slight changes to optimisation

Example: Aloha communication protocol



social fairness (SF)
more equitable
than social welfare (WF_i)

Tool support: PRISM-games

- PRISM-games
  - supports turn-based/concurrent SGs, zero-sum/equilibria
    - and more (co-safe LTL, multi-objective, real-time extensions, …)
  - explicit-state and symbolic implementations
  - custom modelling language extending PRISM

```
csg
player p1 user1 endplayer
player p2 user2 endplayer
// Users (senders)
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
// Channel: used to compute joint probability distribution for transmission failure
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
```

- Growing interest: other (TSG) tools becoming available
  - Tempest, EPMC, PET, PRISM-games extensions

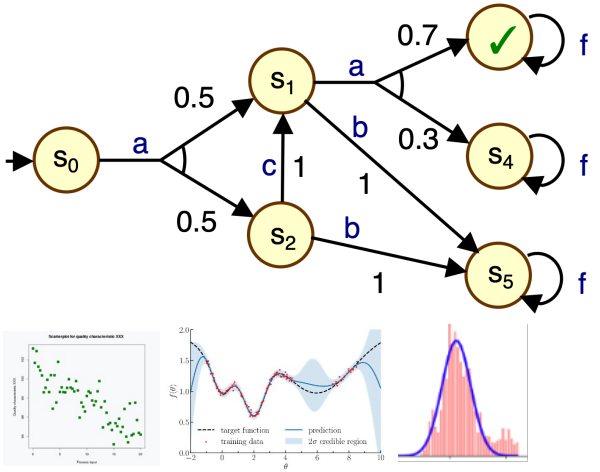
- Many other example application domains
  - attack-defence trees, self-adaptive software architectures, human-in-the-loop UAV mission planning, trust models, collective decision making, intrusion detection policies

prismmodelchecker.org/games/

Robust decision making

Reasoning about uncertainty

- Markov decision processes (MDPs) and variants
  - standard models for sequential decision making under uncertainty
  - stochastic processes quantify uncertainty
  - but parameters of these often need to be estimated from data

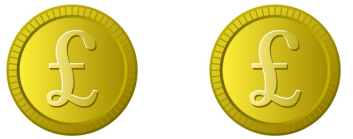


- We distinguish between:
- Aleatoric uncertainty (randomness intrinsic to environment)
  - e.g., sensor noise, actuator failure, human decisions
- Epistemic uncertainty (quantifies lack of knowledge)
  - reducible: can reduce by collecting more data/observations
  - e.g., poor model quality due to low number of measurements

(1)

P(H)=P(T)=0.5

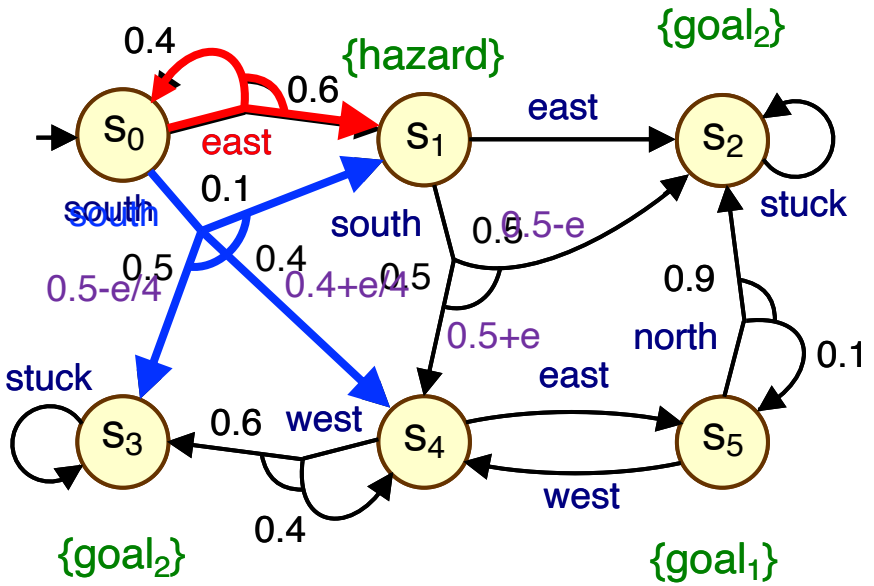
(2) ?

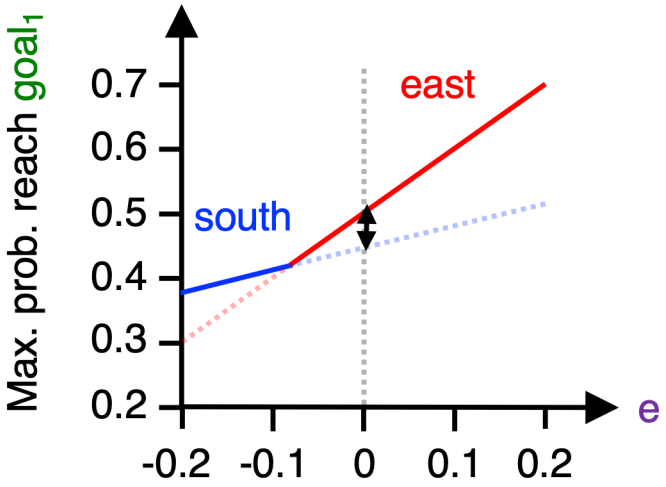


P(H)=1 P(T)=1

MDPs + epistemic uncertainty

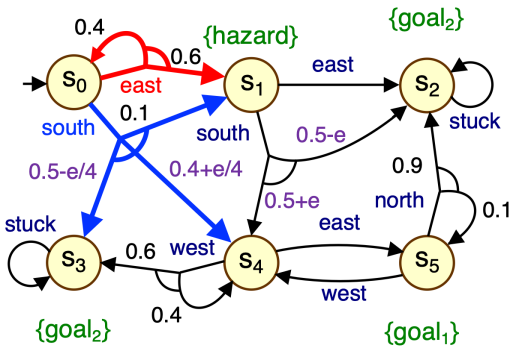
- MDPs for sequential decision making under (aleatoric) uncertainty
  - modelled here using transition probabilities (often learnt from data)
- Policies can be sensitive to small perturbations in transition probabilities
  - so "optimal" policies can in fact be sub-optimal





MDPs + epistemic uncertainty

- MDPs for sequential decision making under (aleatoric) uncertainty
  - modelled here using transition probabilities (often learnt from data)
- Policies can be sensitive to small perturbations in transition probabilities
  - so "optimal" policies can in fact be sub-optimal
- Uncertain MDPs: MDPs + epistemic uncertainty (model uncertainty)
  - we focus here on uncertainty in transition probabilities



Key questions:

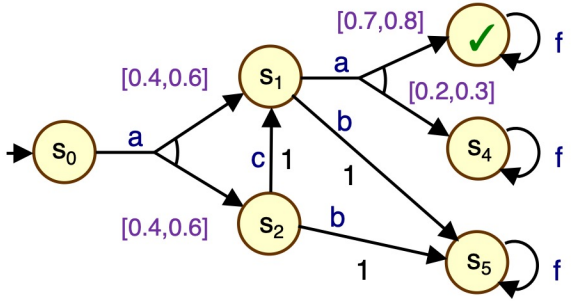
- how to model (and solve for) epistemic uncertainty?
- what guarantees do we get?
- is it statistically accurate?
- how computationally efficient is it?

Uncertain MDPs

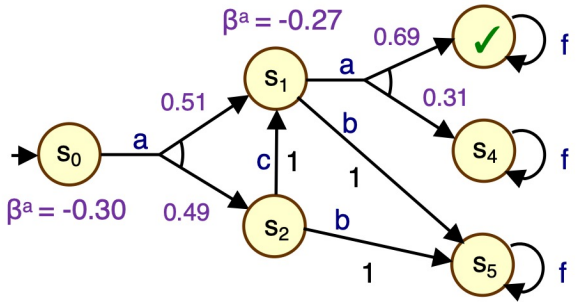
- An uncertain MDP (uMDP), also called a robust MDP
  - can be seen as an MDP with a set $\mathcal{P}$ of transition functions
  - i.e., each $\delta \in \mathcal{P}$ is of the form $\delta : S \times A \to \text{Dist}(S)$
  - we often specify separate uncertainty sets $\mathcal{P}_{s,a} \subseteq \text{Dist}(S)$

- Some examples of uMDPs

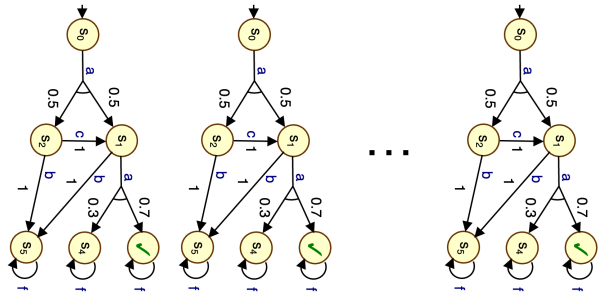
Interval MDPs (IMDPs)



Likelihood MDPs



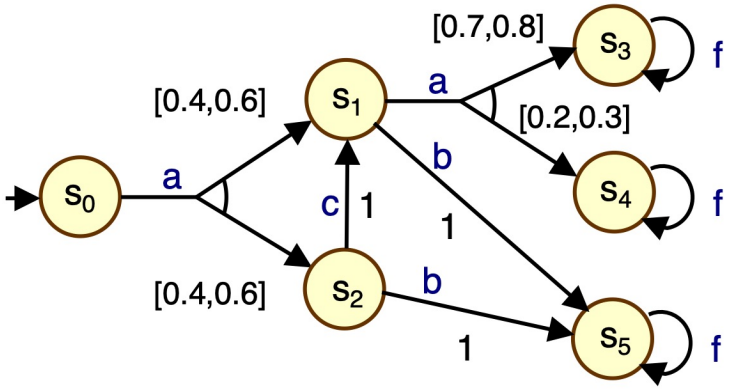
Sampled MDPs



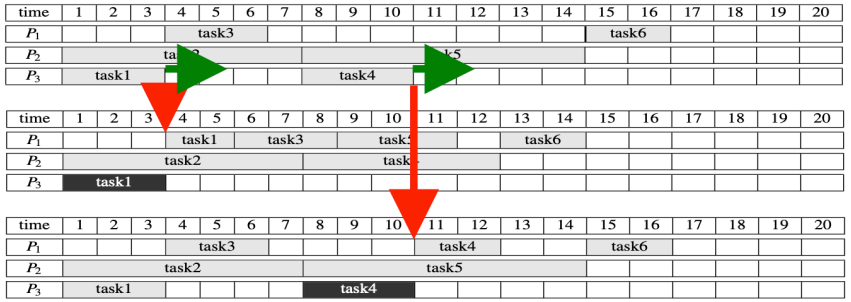
- Other (non-set) representations are possible: dynamic, Bayesian, …

Uncertainty set dependencies

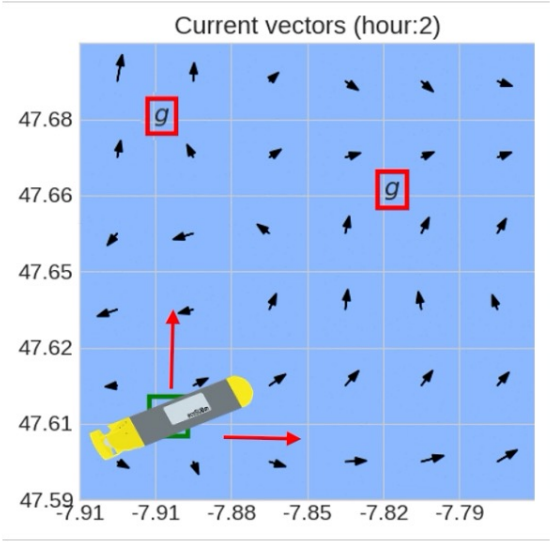
- We often assume (s,a)-rectangularity
  - no dependencies between uncertainty sets: $\mathcal{P} = \times_{(s,a)\in S\times A} \mathcal{P}_{s,a}$
  - computational tractability vs. modelling accuracy



- When might dependences between uncertainties arise?
  - often from shared model parameters



Task scheduling in the presence of faulty processors

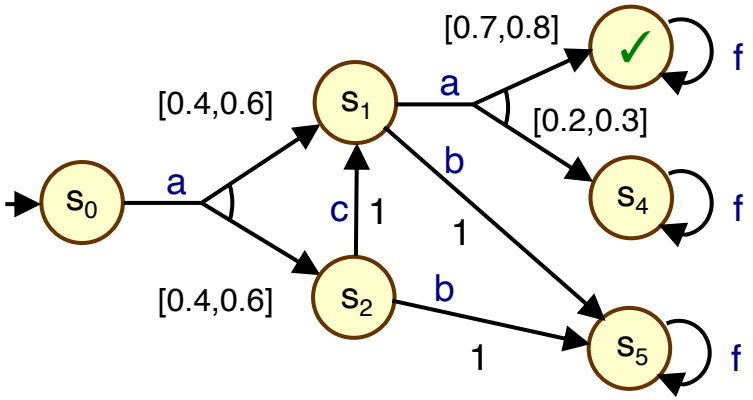


Underwater vehicle control in unknown ocean currents

Robust control

- For now, we consider a robust view of uncertainty
  - i.e., we focus on worst-case (adversarial, pessimistic) scenarios

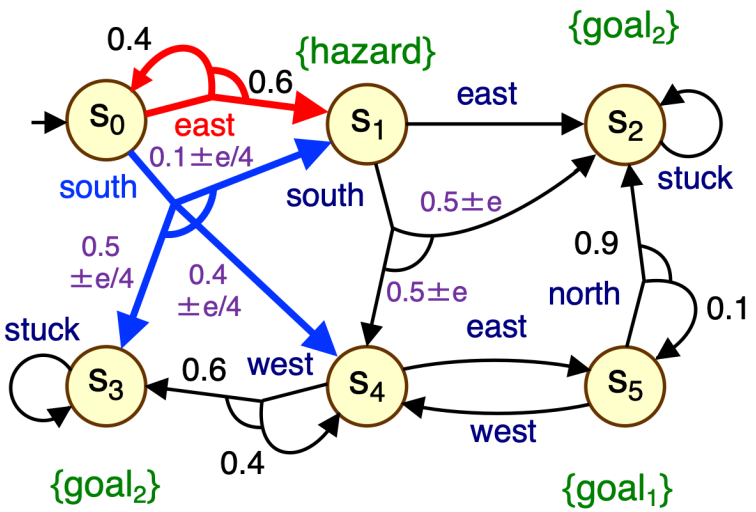
- Robust policy evaluation:
  - policies σ are defined as for MDPs
  - as are objectives e.g. $P_{max=?}$ [F ✓]
  - for a (maximising) policy σ:
  - worst-case value: $\inf_{\delta \in \mathcal{P}} Pr_s^{\delta,\sigma}$ (F ✓)



- Robust control (policy optimisation):
  - optimal worst-case value p* = $\sup_\sigma \inf_{\delta \in \mathcal{P}} Pr_s^{\delta,\sigma}$ (F ✓)
  - optimal worst-case policy σ* = $\text{argsup}_\sigma \inf_{\delta \in \mathcal{P}} Pr_s^{\delta,\sigma}$ (F ✓)
  - p* represents a robust guarantee, i.e., $P_{\geq p^*}$ [F ✓] always holds

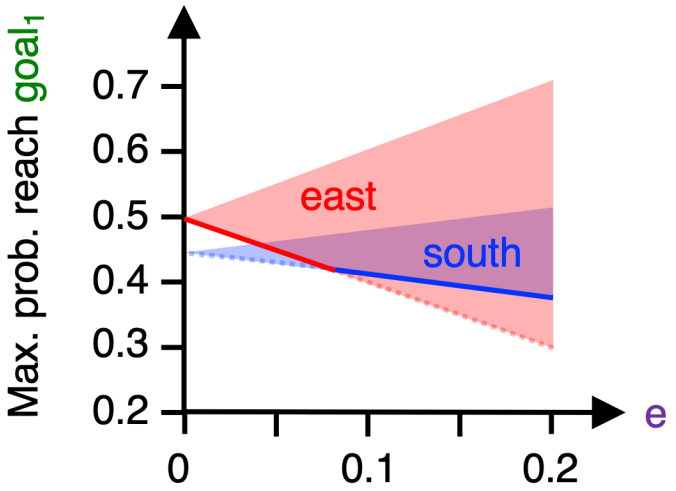
Running example: Robust control

- An IMDP for the robot example
  - uncertainty added to two state-action pairs



  - Note: the degree of uncertainty (e) in states s_1 and s_2 is correlated here (but the actual transition probabilities are not)

- Robust control
  - for any e, we can pick a "robust" (optimal worst-case) policy
  - and give a safe lower bound on its performance

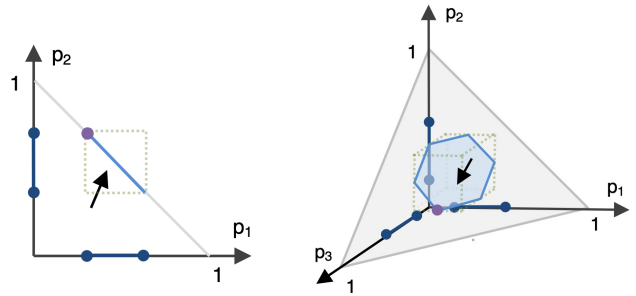


Robust control

- Can be solved with robust value iteration

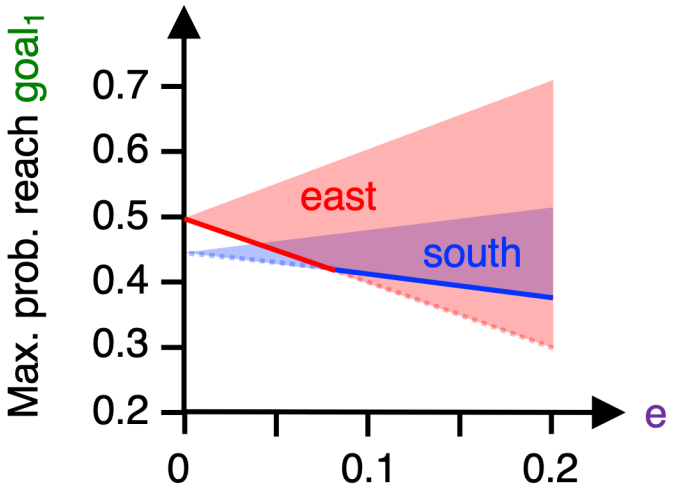
$$p(s) = \begin{cases} 1 & \text{if } s \models \checkmark \\ \max_a \min_{\delta \in \mathcal{P}_{s,a}} \Sigma_{s'} \, \delta(s,a)(s') \cdot p(s') & \text{if } s \not\models \checkmark \end{cases}$$

  - various techniques for solving inner optimisation problems



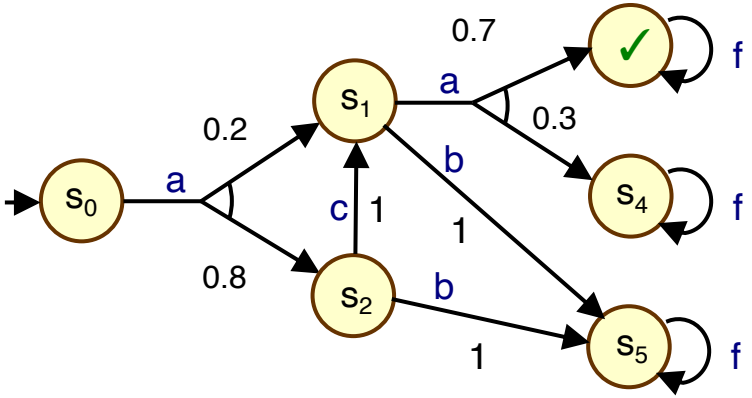
- Implemented/available in PRISM

- Robust control
  - for any e, we can pick a "robust" (optimal worst-case) policy
  - and give a safe lower bound on its performance



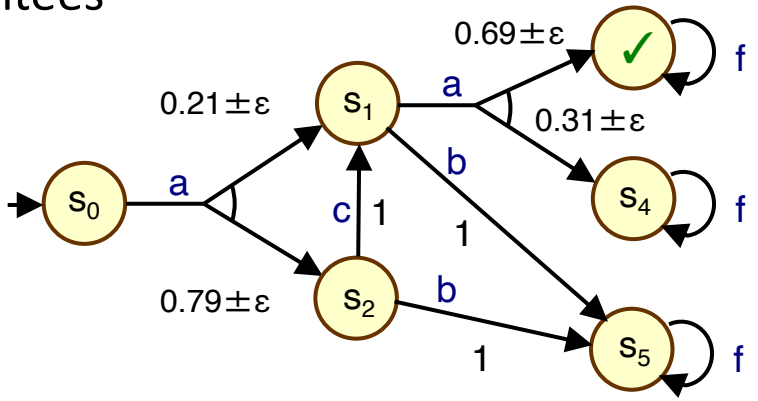
Learning IMDPs

- We can learn IMDP models from samples of transitions/trajectories
  - of the (fixed, but unknown) "true" MDP
  - either online (interactively) or offline (from existing logs)



- Uncertainty sets in the IMDP
  - are based on confidence intervals
  - around point estimates for transition probabilities $P_s^a(s_i)$
  - yielding probably approximately correct (PAC) guarantees
  - we fix an error rate γ and compute an error ε

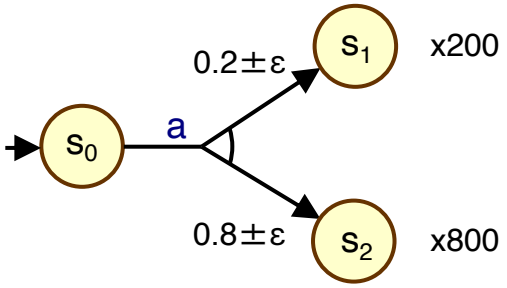
$$Pr(\delta \in \mathcal{P}) \geq 1 - \gamma$$



Learning IMDPs

- For each state s and action a
  - we have sample counts $N = \#(s,a)$ and $k_i = \#(s,a,s_i)$
  - the point estimate for the transition is: $\tilde{P}_s^a(s_i) \approx k_i/N$
  - the confidence interval is: $\tilde{P}_s^a(s_i) \pm \varepsilon$ where $\varepsilon = \sqrt{\log(2/\gamma)/2N}$
  - with PAC guarantee: $Pr(P_s^a(s_i) \in \tilde{P}_s^a(s_i) \pm \varepsilon) \geq 1 - \gamma$

(via Hoeffding's inequality)

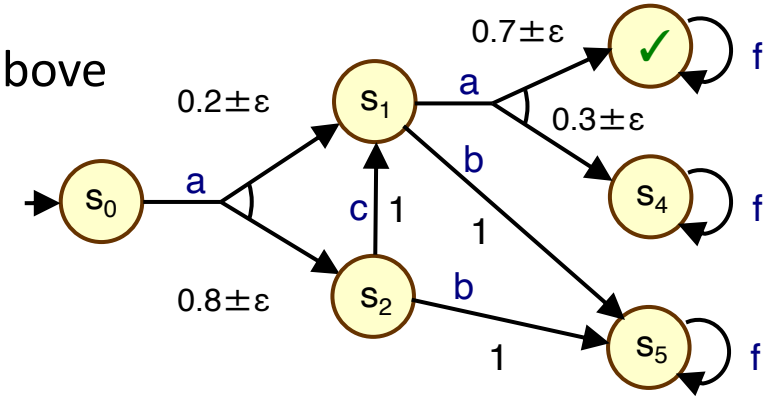


- We can lift this to the whole IMDP
  - building uncertain transition set $\mathcal{P}$ using intervals as above

$$Pr(\delta \in \mathcal{P}) \geq 1 - \gamma$$

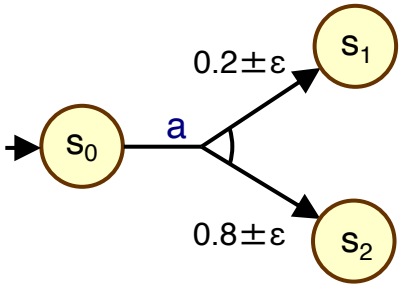
(after distributing error rate γ)

  - and also to our robust guarantees $P_{\geq p^*}$ [F ✓]



Learning IMDPs

- For each state s and action a
  - we have sample counts $N = \#(s,a)$ and $k_i = \#(s,a,s_i)$
  - the point estimate for the transition is: $\tilde{P}_s^a(s_i) \approx k_i/N$
  - the confidence interval is: $\tilde{P}_s^a(s_i) \pm \varepsilon$ where $\varepsilon = \sqrt{\log(2/\gamma)/2N}$
  - with PAC guarantee: $Pr(P_s^a(s_i) \in \tilde{P}_s^a(s_i) \pm \varepsilon) \geq 1 - \gamma$



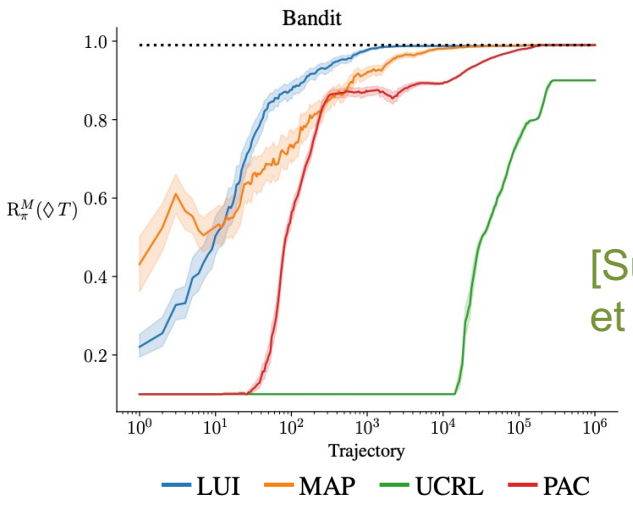
(via Hoeffding's inequality)

- We can lift this to the whole IMDP
  - building uncertain transition set $\mathcal{P}$ using intervals as above

$$Pr(\delta \in \mathcal{P}) \geq 1 - \gamma$$

(after distributing error rate γ)

  - and also to our robust guarantees $P_{\geq p^*}$ [F ✓]

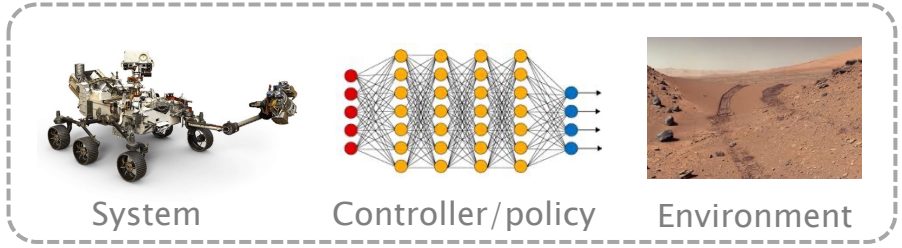


[Suilen et al.'22]

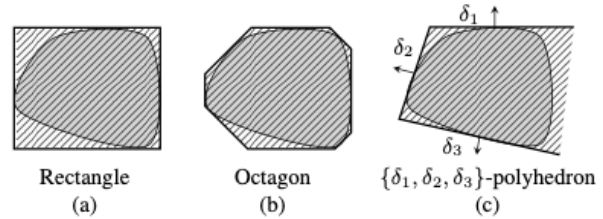
Neuro-symbolic decision making

Deep reinforcement learning

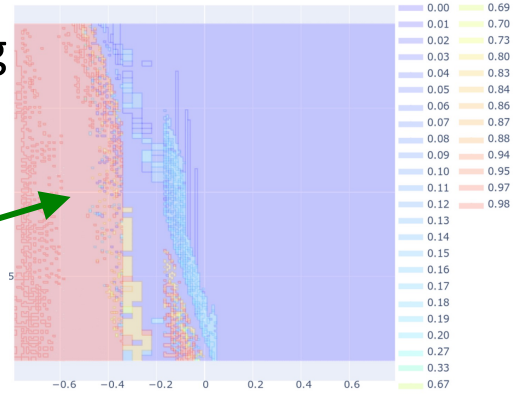
- Tackling more realistic problems
  - continuous state spaces & more complex dynamics
- Verification of learning-based systems
  - e.g., deep reinforcement learning
  - neural network (NN) learnt for strategy actions/values
- First steps: single-agent verification, fixed policy
  - deterministic dynamical system + control faults
  - combine polyhedral abstractions with probabilistic model checking
  - conservative abstraction of NN-controlled dynamics over a finite horizon, via MILP



deep reinforcement learning



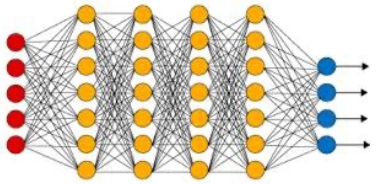
upper bounds on failure probabilities for initial regions

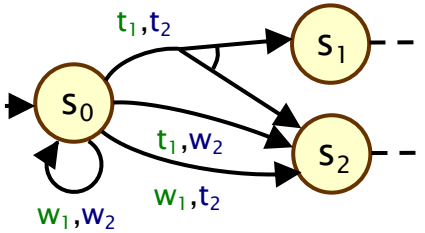


pendulum benchmark

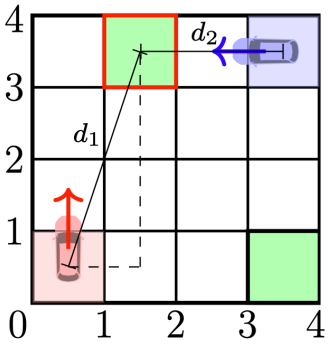
Neuro-symbolic games

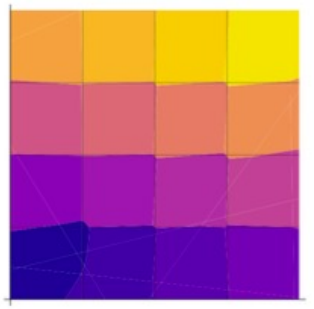
- Mixture of neural components + symbolic/logical components
  - simpler than end-to-end neural control problem; aids explainability
  - here: neural networks (or similar) for perception tasks
  - plus: local strategies for control decisions





- Neuro-symbolic CSGs
  - finite-state agents + continuous-state environment E
    - $S = (Loc_1 \times Per_1) \times (Loc_2 \times Per_2) \times S_E$
  - agents use a (learnt) perception function to observe E
    - $obs_i : (Loc_1 \times Loc_2) \times S_E \rightarrow Per_i$
  - CSG-like joint actions update state probabilistically



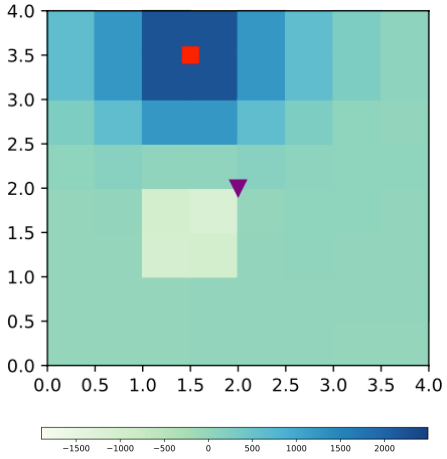


- Example: dynamic vehicle parking
  - NN maps exact vehicle position to perceived grid cell

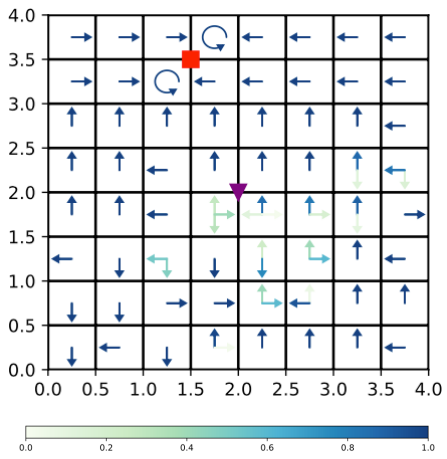
Model checking neuro-symbolic CSGs

- Strategy synthesis for zero-sum (discounted) expected reward
  - for now, we assume full observability
- Value iteration (VI) approach
  - continuous state-space decomposed into regions
  - further subdivision at each iteration
  - we define a class of piecewise-continuous value functions, preserved by NNs and VI
- Implementation
  - pre-image computations of NNs
  - polytope representations of regions
  - LPs to solve zero-sum games at each step

Dynamic vehicle parking with larger (8x8) grid and simpler (regression) perception



Value function (fragment)



Optimal strategy (fragment)

Wrapping up

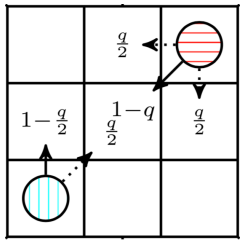
Overview

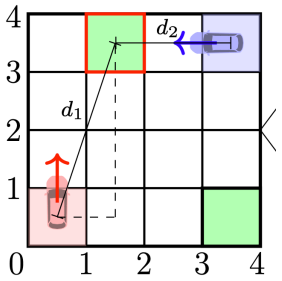
- Sequential decision making under uncertainty
- Formal verification: probabilistic model checking

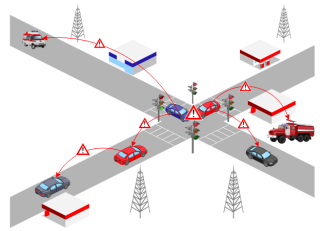
- Multi-agent decision making
- Data-driven models for decision making
- Neuro-symbolic decision making

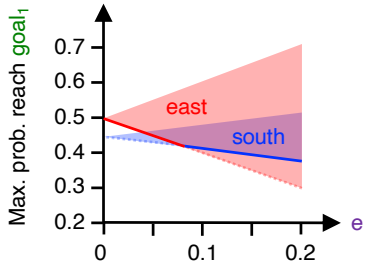
Challenges & directions

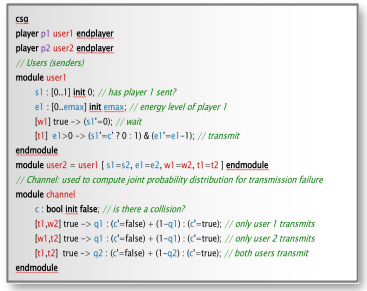
- Partial information/observability
  - e.g., leveraging progress on POMDPs
- Managing robustness and uncertainty
  - e.g., stability of randomised strategies
- Modelling language design and extensions
  - e.g., for specifying uncertainty
  - e.g., more flexible interchange of components and strategies
- Further classes of equilibria
  - e.g. Stackelberg equilibria for automotive/security applications
- Improving scalability & efficiency
  - e.g. symbolic methods for CSGs, compositional solution

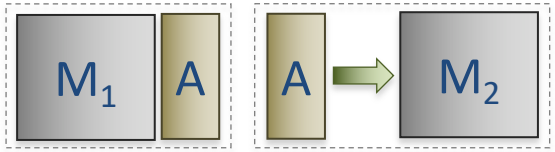












prismmodelchecker.org