Robust Verification and Control under Uncertainty

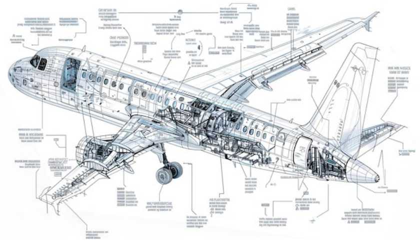
Dave Parker

UNIVERSITY OF OXFORD

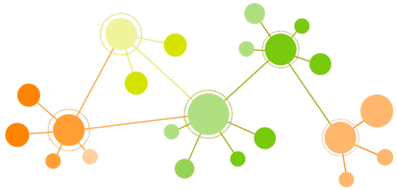
SETTA 2025

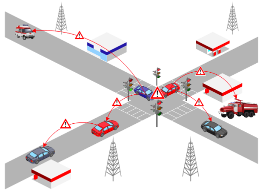
Formal verification under uncertainty

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


- This talk: advances in verifying autonomous systems under uncertainty
  1. verified control & sequential decision making
  2. verifying multiple agents acting autonomously and concurrently
  3. robust verification under models learnt from data





Overview

- Probabilistic model checking
  - key ideas, applications, trends

- Verification and control
  - Markov decision processes and beyond

- Multi-agent systems
  - probabilistic model checking with stochastic games
  - competitive or collaborative behaviour

- Robustness under model uncertainty
  - probabilistic model checking with epistemic uncertainty
  - robust guarantees for data-driven models

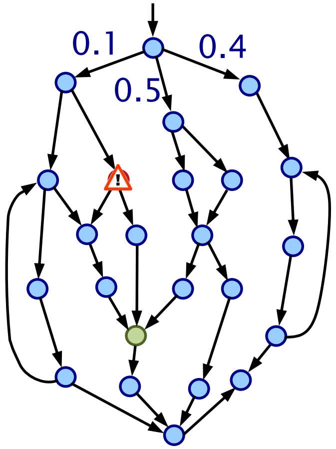
Probabilistic model checking

Probabilistic model checking

- Probabilistic model checking
  - systematic construction and analysis of probabilistic models
  - key ingredients: logic, automata, probability

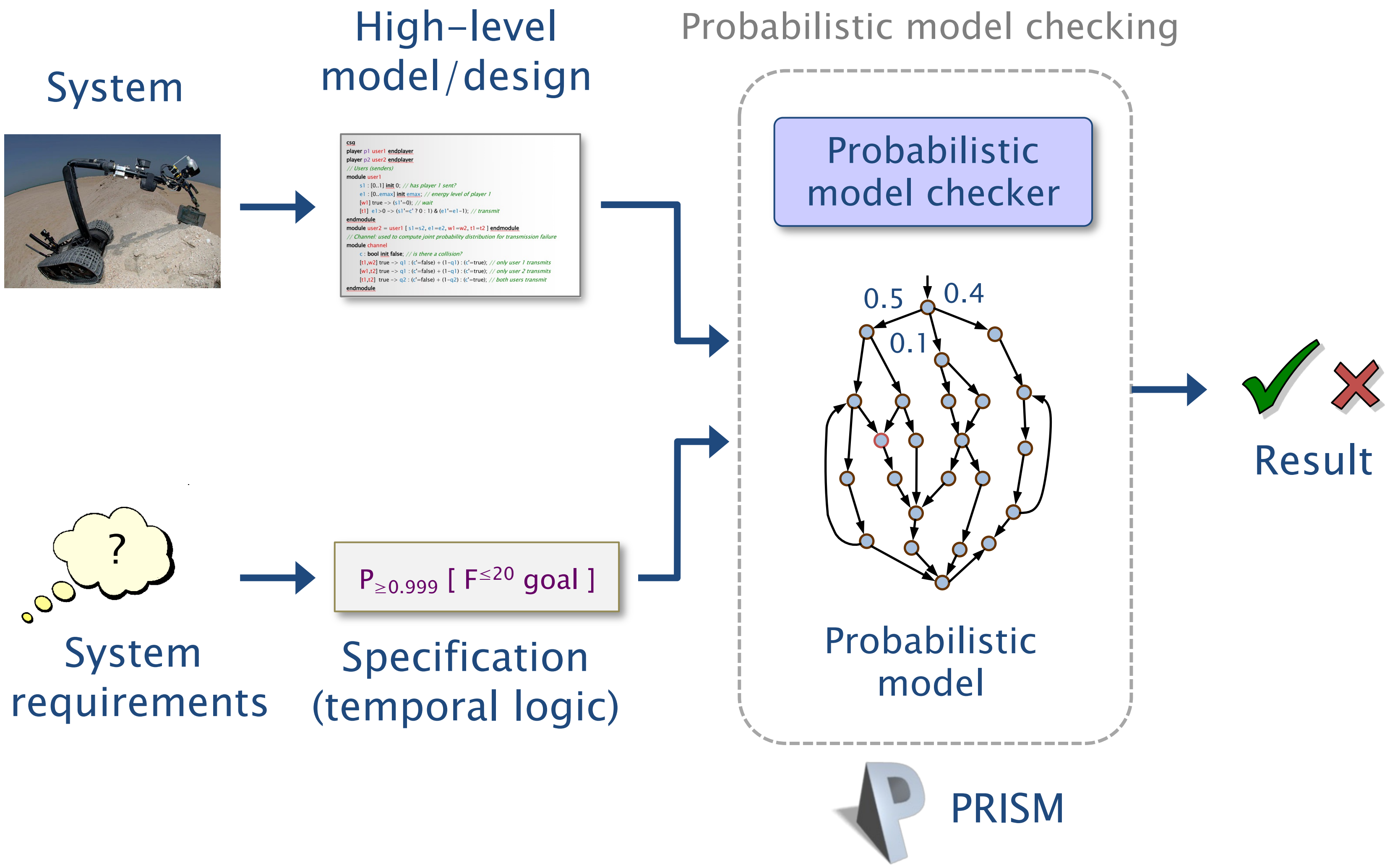
- Connections to:
  - Markov models, graph theory, artificial intelligence, control theory, optimisation, SAT/SMT, game theory, …

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

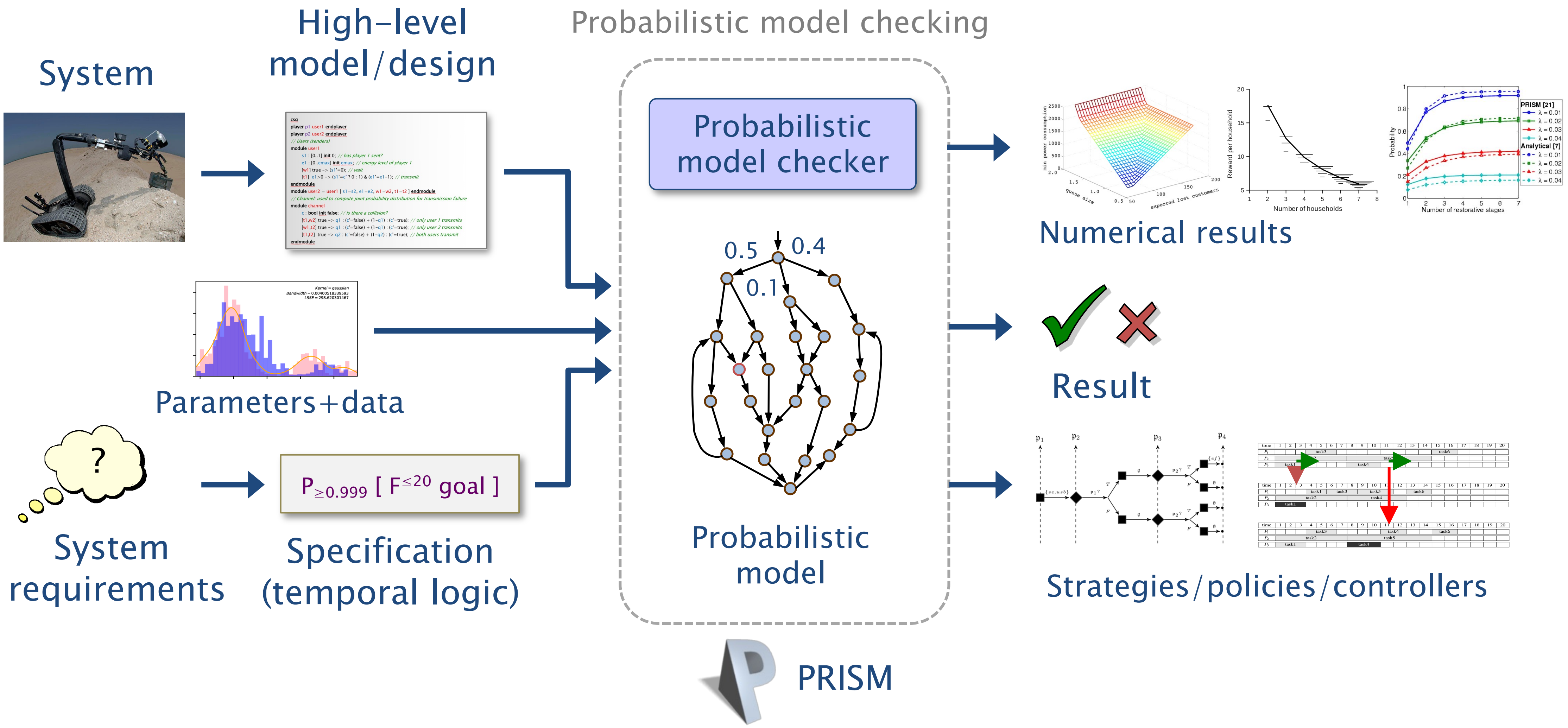


- Key strengths: exhaustive + numeric analysis
  - often subtle interplay between probability + nondeterminism
- Applications to:
  - airbag design, satellite reliability, pacemaker designs, communication/security/energy protocols, robotics, …

Probabilistic model checking (PMC)

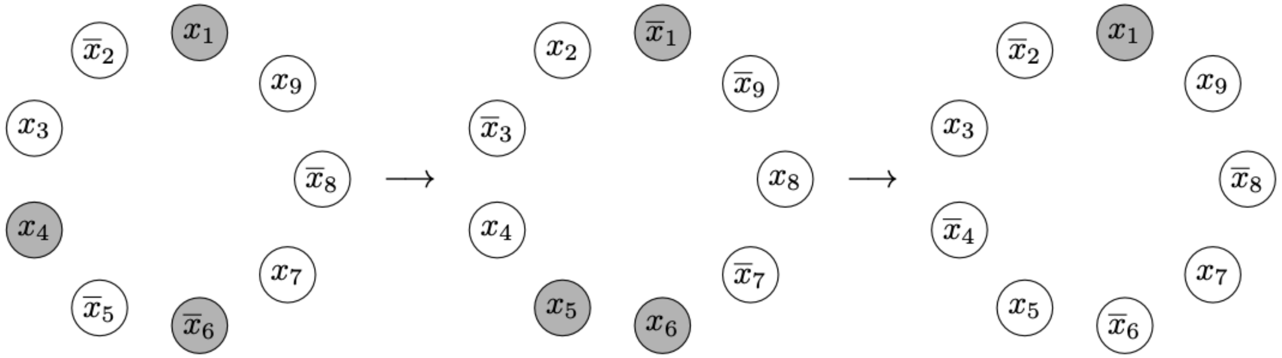


Probabilistic model checking (PMC)



Example: Self-stabilisation

- Randomised self-stabilisation (leader election) algorithm [Herman'90]
  - for a ring of N identical synchronous processes, some of which have tokens

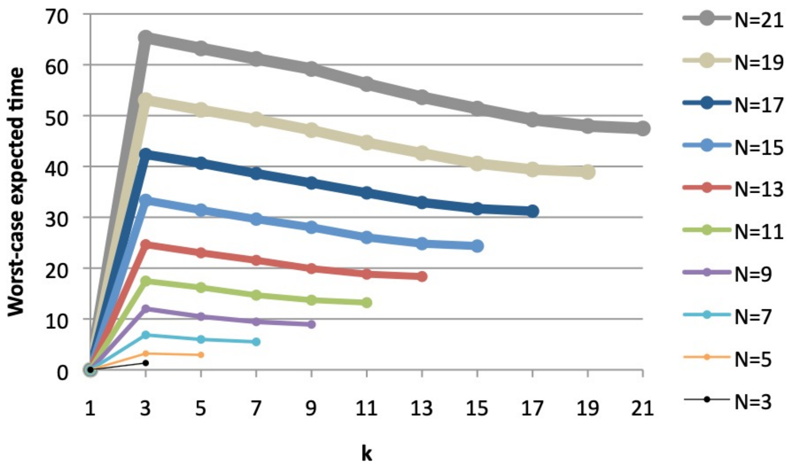


  - each process with a token randomly (i) keeps it or (ii) passes it to its right neighbour
  - if a process has two tokens, both are eliminated

- Correctness <u>and</u> performance
  - is this guaranteed to stabilise (to one token)? (yes)
  - what (and when) is the worst-case expected run time? …

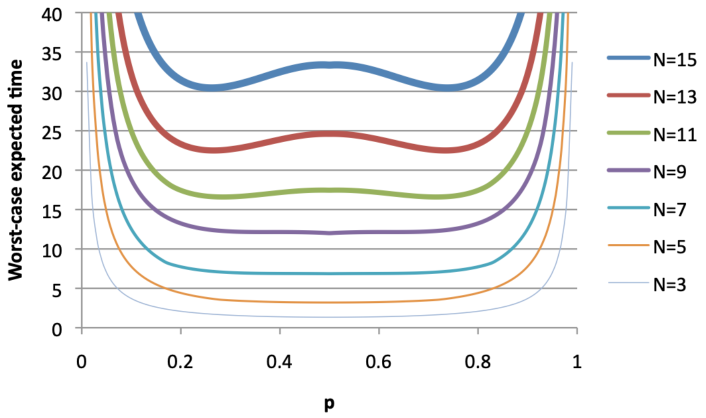
Example: Self-stabilisation

- Conjecture [McIver&Morgan'05]
  - worst-case expected runtime occurs with k=3 initial tokens

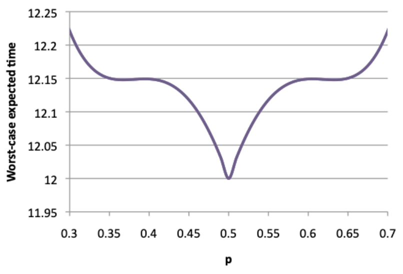


  - validated with PRISM for finite configurations
  - finally proved 10 years later

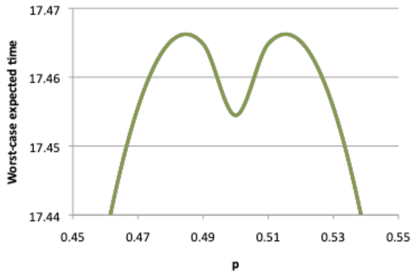
- Can we improve performance
  - using a biased coin?



  - zooming in…



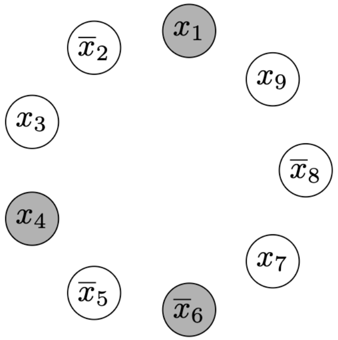
(b) $N = 9$



(c) $N = 11$

PRISM modelling language

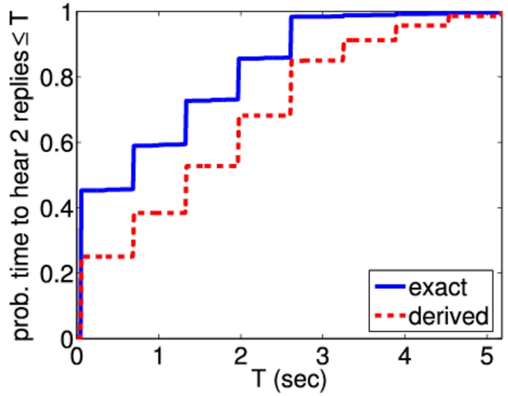
```
// herman's self stabilising algorithm [Her90]
// gxn/dxp 13/07/02

// the procotol is synchronous with no nondeterminism (a DTMC)
dtmc

const double p = 0.5;

// module for process 1
module process1

        // Boolean variable for process 1
        x1 : [0..1];

        [step]  (x1=x3) -> p : (x1'=0) + 1-p : (x1'=1);
        [step] !(x1=x3) -> (x1'=x3);

endmodule

// add further processes through renaming
module process2 = process1 [ x1=x2, x3=x1 ] endmodule
module process3 = process1 [ x1=x3, x3=x2 ] endmodule

// cost - 1 in each state (expected number of steps)
rewards "steps"
        true : 1;
endrewards

// set of initial states: all (i.e. any possible initial configuration of tokens)
init
        true
endinit

// formula, for use in properties: number of tokens
// (i.e. number of processes that have the same value as the process to their left)
formula num_tokens = (x1=x2?1:0)+(x2=x3?1:0)+(x3=x1?1:0);

// label - stable configurations (1 token)
label "stable" = num_tokens=1;
```



Example: Bluetooth

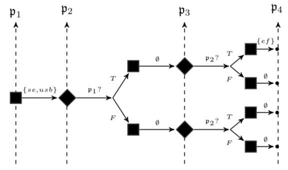
- Device discovery between a pair of Bluetooth devices
  - performance guarantees essential for this phase
- Complex discovery process
  - two asynchronous 28-bit clocks
  - pseudo-random hopping between 32 frequencies
  - random waiting scheme to avoid collisions

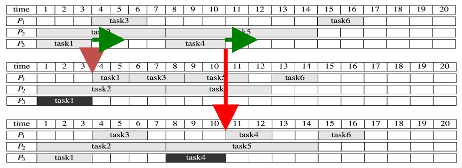
- Probabilistic model checking with PRISM
  - worst-case expected time and probability for successful discovery
  - Markov chains with 17,179,869,184 initial configurations
  - exhaustive numerical analysis via symbolic model checking
  - highlights flaws in a simpler, analytic analysis



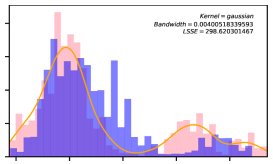
Trends in probabilistic model checking

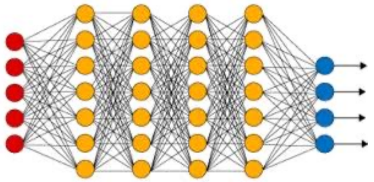
- From verification problems to control/synthesis
  - "correct-by-construction" from temporal logic specifications





- 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

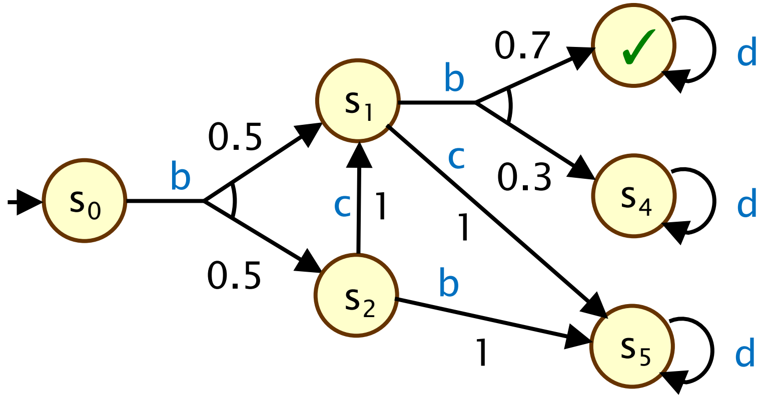
CTMC, CSG, DTMC, LTS, MDP, POMDP, POPTA, PTA, STPG, SMG, TPTG, IDTMC, IMDP

Verification & control

(with Markov decision processes)

Markov decision processes

- Markov decision processes (MDPs)
  - probability + nondeterminism (over actions A)
  - transition probabilities: $\delta : S \times A \rightarrow Dist(S)$



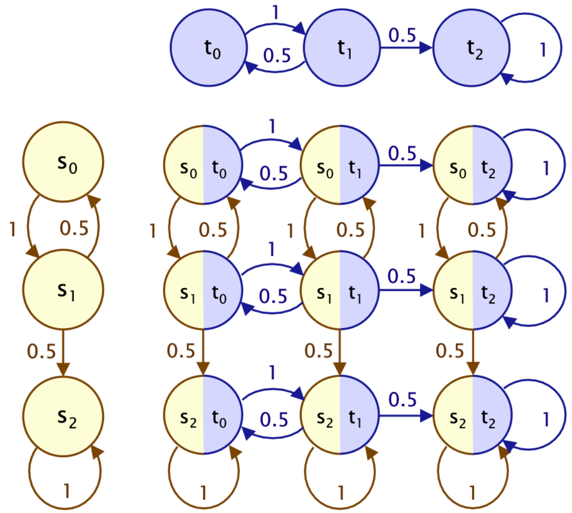
- Policies
  - policies σ : Path(s) → Dist(A) resolve actions based on history
  - can be memoryless/finite-memory and deterministic/randomised

- Probabilistic reachability
  - $P_{max=?}$ [F ✓] = $\sup_\sigma Pr_s^\sigma$(F ✓)
  - what is the maximum probability of reaching ✓ achievable by any policy σ?
  - optimal policies are deterministic and memoryless
  - solvable with dynamic programming (known as value iteration)

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

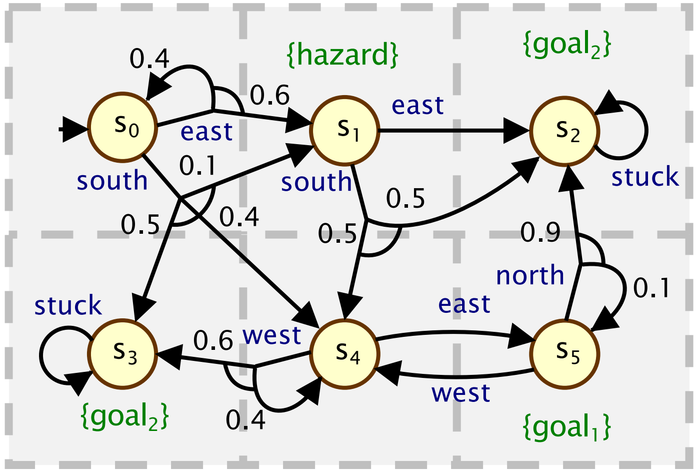
MDPs: Verification vs. control

- MDPs (or probabilistic automata) model probability + concurrency
  - e.g. randomised distributed algorithms/protocols



What is the worst-case probability of (or expected time for) termination under any possible process scheduling?

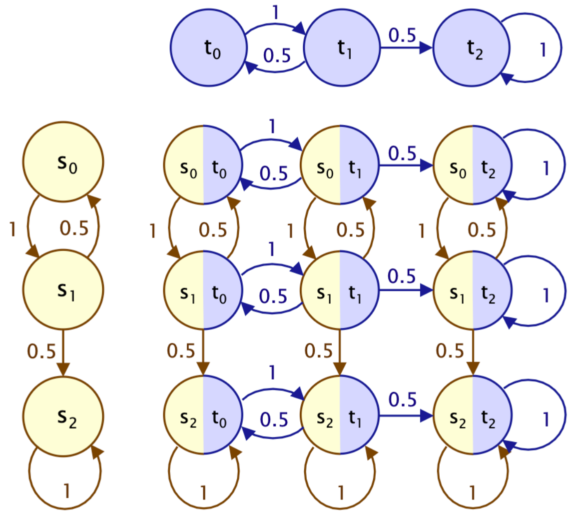
- MDPs are also a standard model for sequential decision making under uncertainty
  - e.g. robot navigation in uncertain environments
  - e.g. task scheduling on fault-prone processors



What is the optimal navigation policy for reaching a goal location whilst avoiding the hazard?

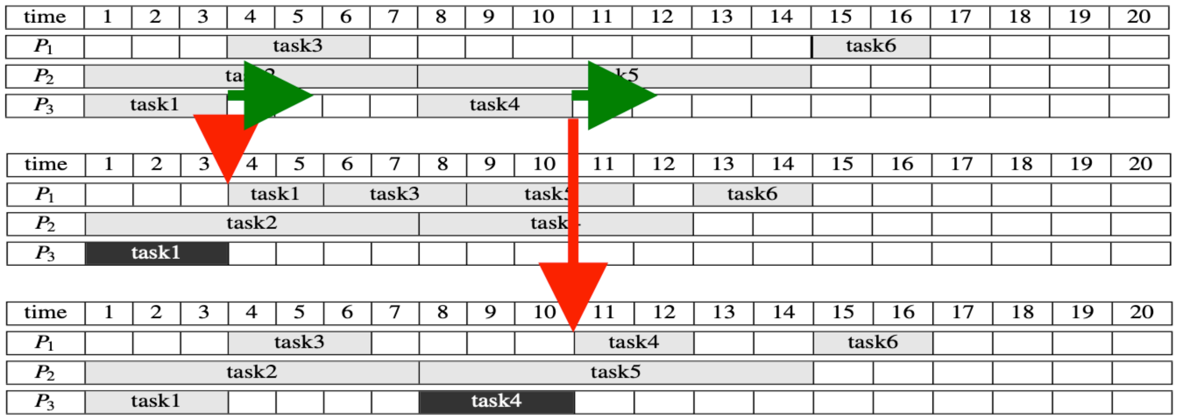
MDPs: Verification vs. control

- MDPs (or probabilistic automata) model probability + concurrency
  - e.g. randomised distributed algorithms/protocols



What is the worst-case probability of (or expected time for) termination under any possible process scheduling?

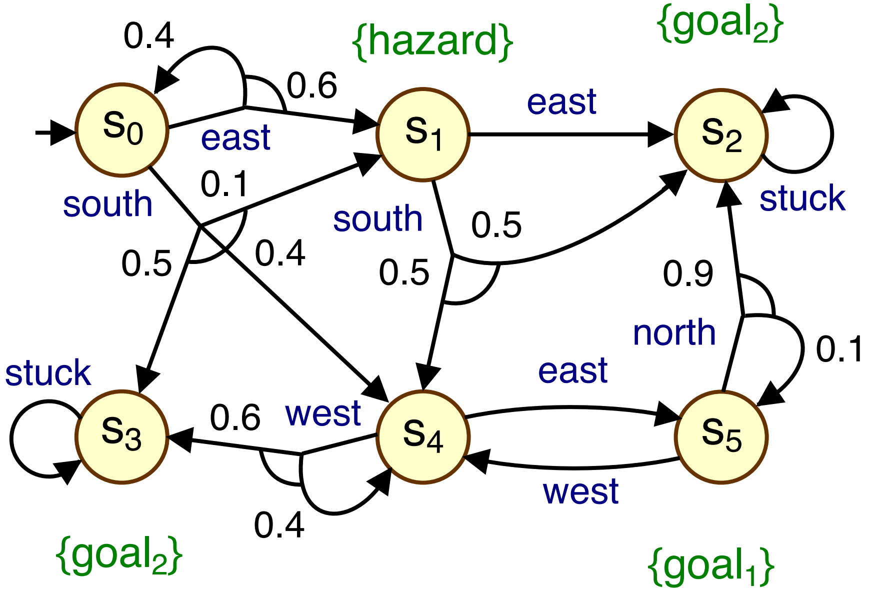
- MDPs are also a standard model for sequential decision making under uncertainty
  - e.g. robot navigation in uncertain environments
  - e.g. task scheduling on fault-prone processors



What is the optimal task scheduling to maximise successful/timely completion?

Correctness by construction

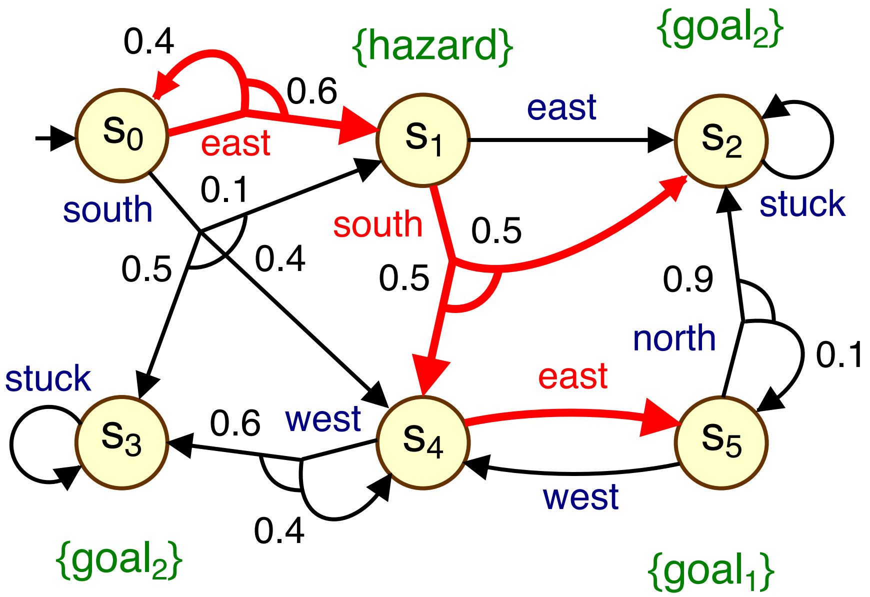
- Synthesise correct-by-construction controllers/policies/plans
  - based on temporal logic specifications, e.g., in PCTL
  - synthesis of MDP policies + probabilistic guarantees on safety/performance



- Can we guarantee reaching goal$_1$ with probability 0.5?
  $P_{\geq 0.5}$ [F goal$_1$]

Correctness by construction

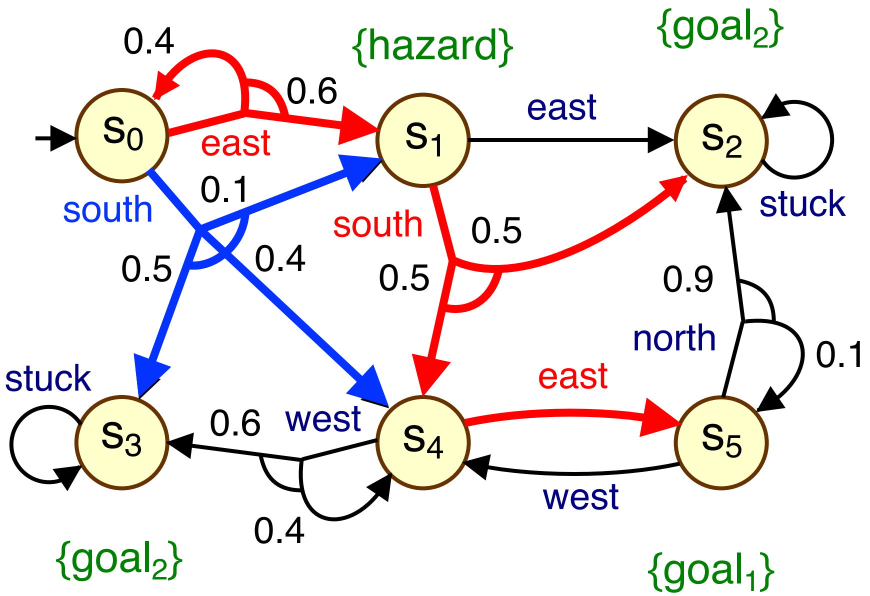
- Synthesise correct-by-construction controllers/policies/plans
  - based on temporal logic specifications, e.g., in PCTL
  - synthesis of MDP policies + probabilistic guarantees on safety/performance



- 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$_1$ whilst avoiding hazard locations?
  $P_{max=?}$ [¬hazard U goal$_1$]

Correctness by construction

- Synthesise correct-by-construction controllers/policies/plans
  - based on temporal logic specifications, e.g., in PCTL
  - synthesis of MDP policies + probabilistic guarantees on safety/performance

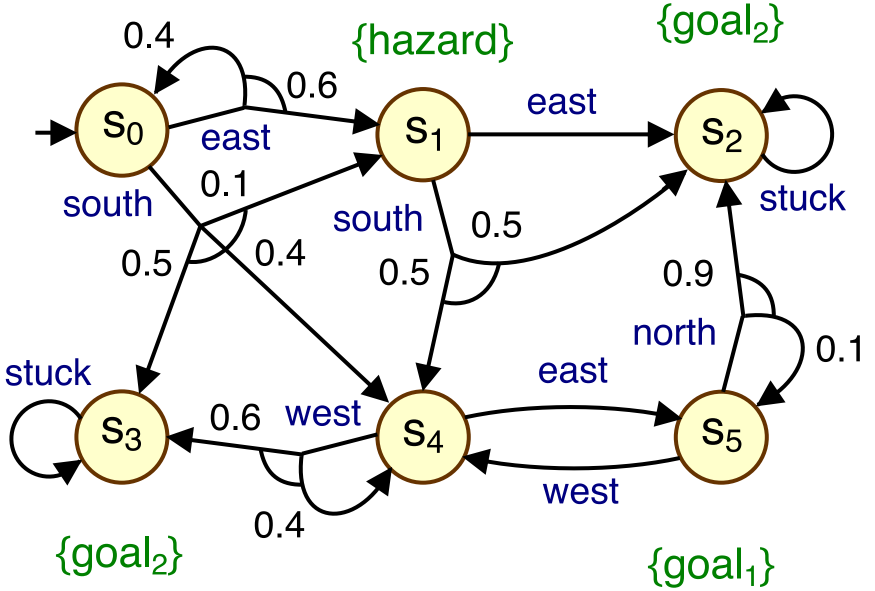


- Can we guarantee reaching goal$_1$ with probability 0.5?
  $P_{\geq 0.5}$ [F goal$_1$]

- How do we maximise the probability of reaching goal$_1$ whilst avoiding hazard locations?
  $P_{max=?}$ [¬hazard U goal$_1$]

Correctness by construction

- Synthesise correct-by-construction controllers/policies/plans
  - based on temporal logic specifications, e.g., in PCTL
  - synthesis of MDP policies + probabilistic guarantees on safety/performance



More complex specifications?

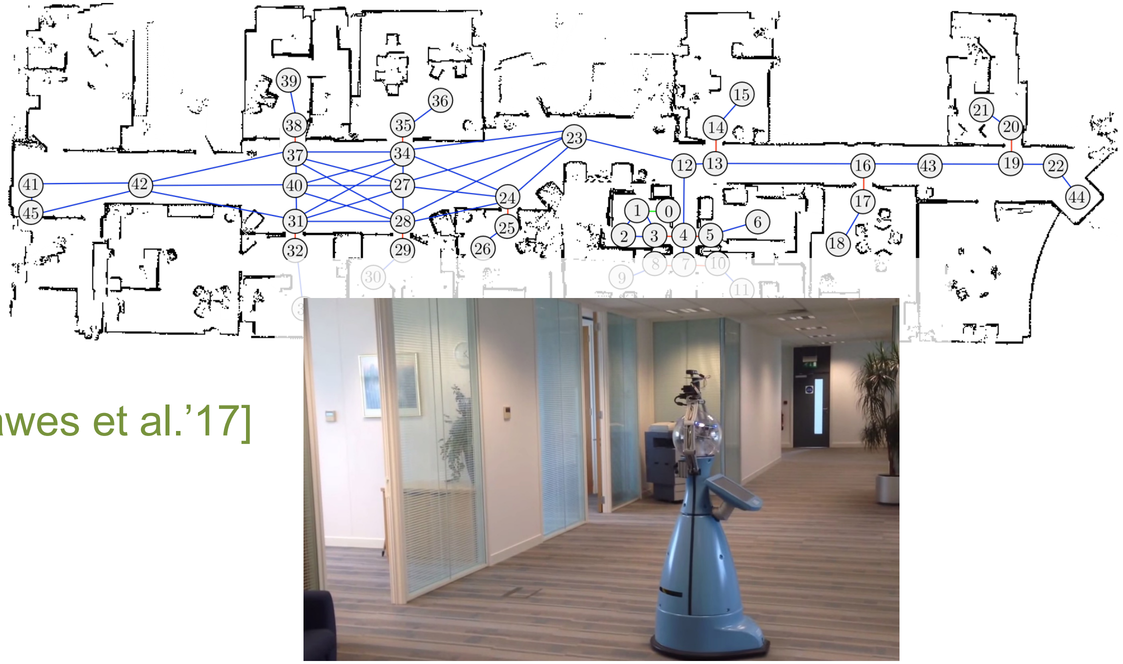
- With high probability, complete the task "visit goal1 then goal2, without passing through hazard" whilst always remaining close to the charging dock.

$P_{>0.99}$ [¬hazard U ($goal_1$ ∧ (F $goal_2$))]
∧ ∀G $P_{>0.95}$ [$F^{\leq 100}$ charge)]

- Optimal finite-memory policies synthesised via automata products

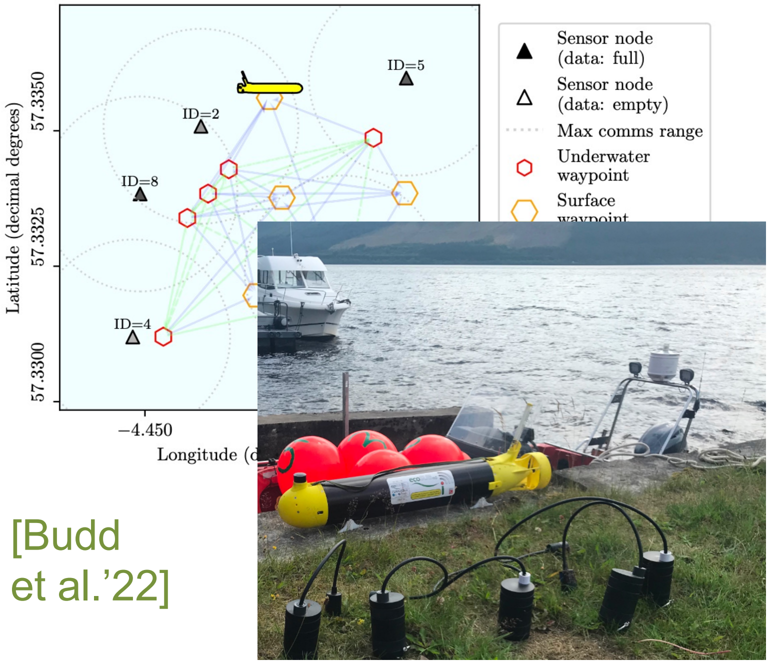
Examples: Robot deployments

- 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

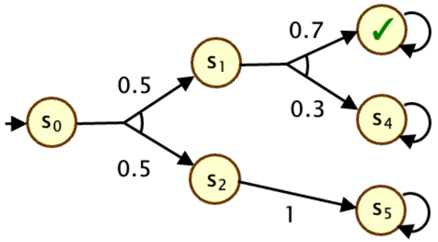


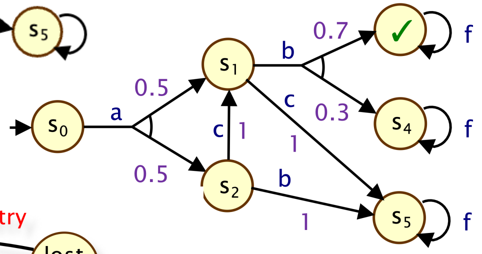
[Budd et al.'22]

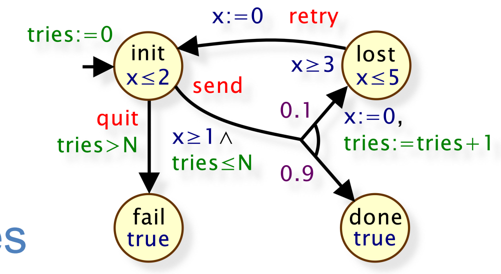
A zoo of probabilistic models

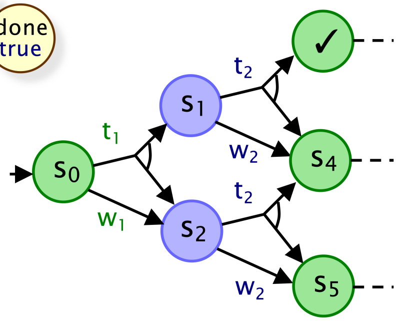
- Increasing variety (and complexity) of probabilistic models supported

• 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





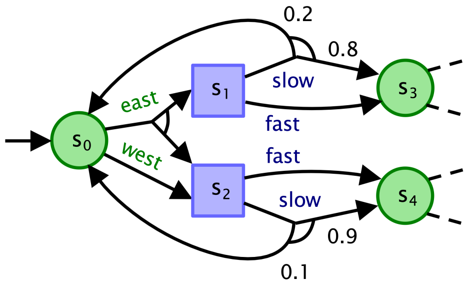




A zoo of probabilistic models

- 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



Overview

- Probabilistic model checking
  - key ideas, applications, trends

- Verification and control
  - Markov decision processes and beyond

- Multi-agent systems
  - probabilistic model checking with stochastic games
  - competitive or collaborative behaviour

- Robustness under model uncertainty
  - probabilistic model checking with epistemic uncertainty
  - robust guarantees for data-driven models

Multi-agent verification

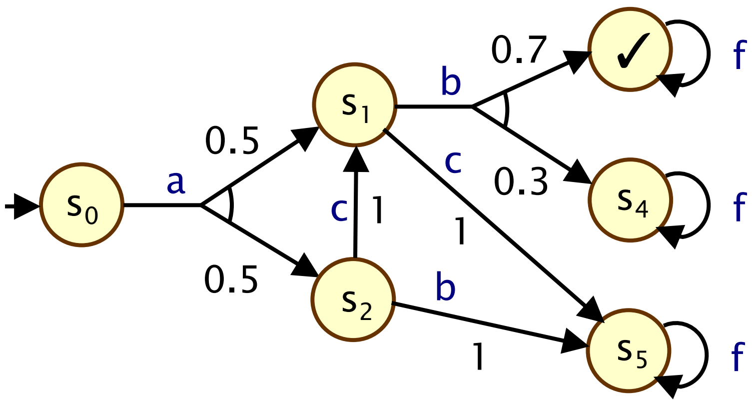
(with stochastic games)

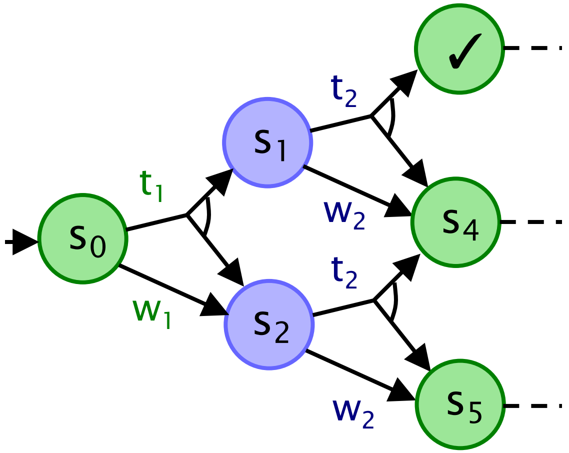
Stochastic multi-player games

- (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)





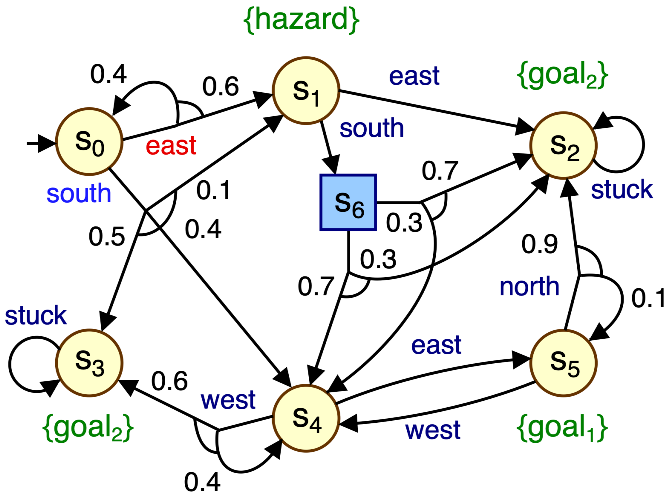
$\delta : S \times A \rightarrow \mathrm{Dist}(S)$

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

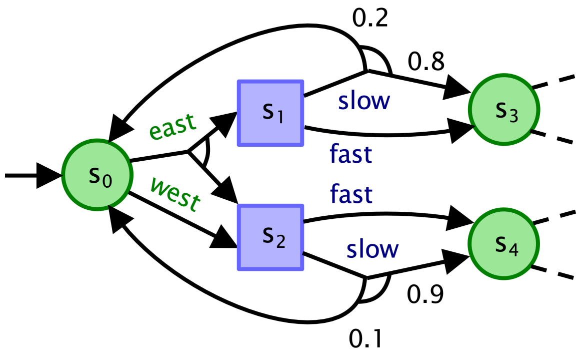
Modelling with turn-based games

- 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 ⟨⟨C⟩⟩ of ATL
    + probabilistic (P) and reward (R) operators

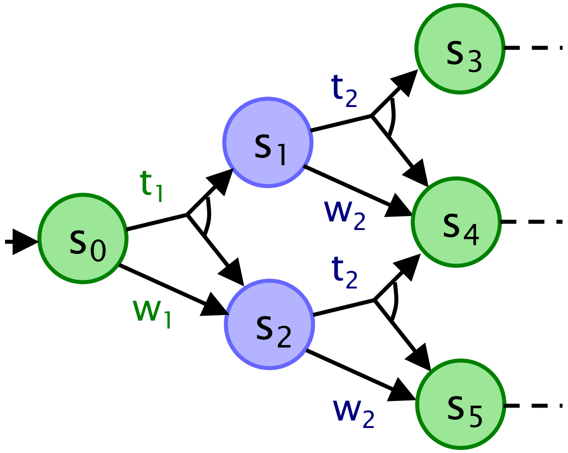
- Example:
  - $\langle\langle\{robot_1,robot_3\}\rangle\rangle\ P_{max=?}\ [\ F\ (goal_1 \vee goal_3)\ ]$
  - "what strategies for robots 1 and 3 <u>maximise</u> the probability of reaching their goal locations, <u>regardless</u> of the strategies of other robots"

Can be seen as a mixture of control <u>and</u> verification

- Other additions:
  - (co-safe) linear temporal logic
    $\neg zone_3\ U\ (room_1 \wedge (F\ room_4 \wedge F\ room_5))$
  - nested specifications
    $\langle\langle\{robot_1,robot_3\}\rangle\rangle\ R_{min=?}\ [$
    $\quad \langle\langle\{robot_1\}\rangle\rangle\ P_{\geq 0.99}\ [\ F^{\leq 10}\ base\ ]$
    $\quad U\ (zone_1 \wedge (F\ zone_4))\ ]$
    "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:

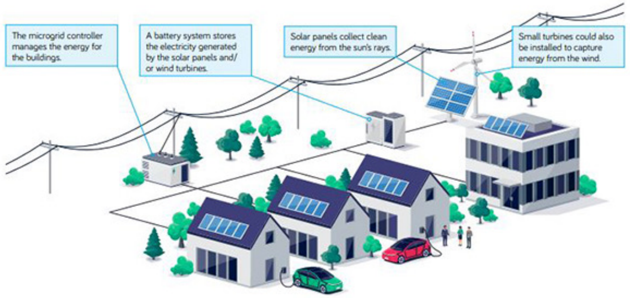
$$p(s) = \begin{cases} 1 & \text{if } s \vDash ✓ \\ \max_a \Sigma_{s'} \delta(s,a)(s') \cdot p(s') & \text{if } s \nvDash ✓ \text{ and } s \in S_1 \\ \min_a \Sigma_{s'} \delta(s,a)(s') \cdot p(s') & \text{if } s \nvDash ✓ \text{ and } s \in S_2 \end{cases}$$

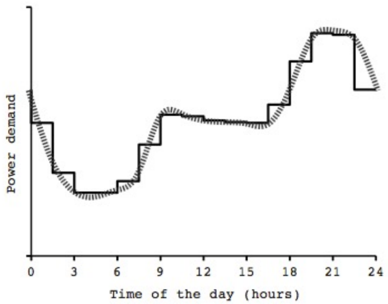
  - and more: graph-algorithms, sequences of fixed points, …

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

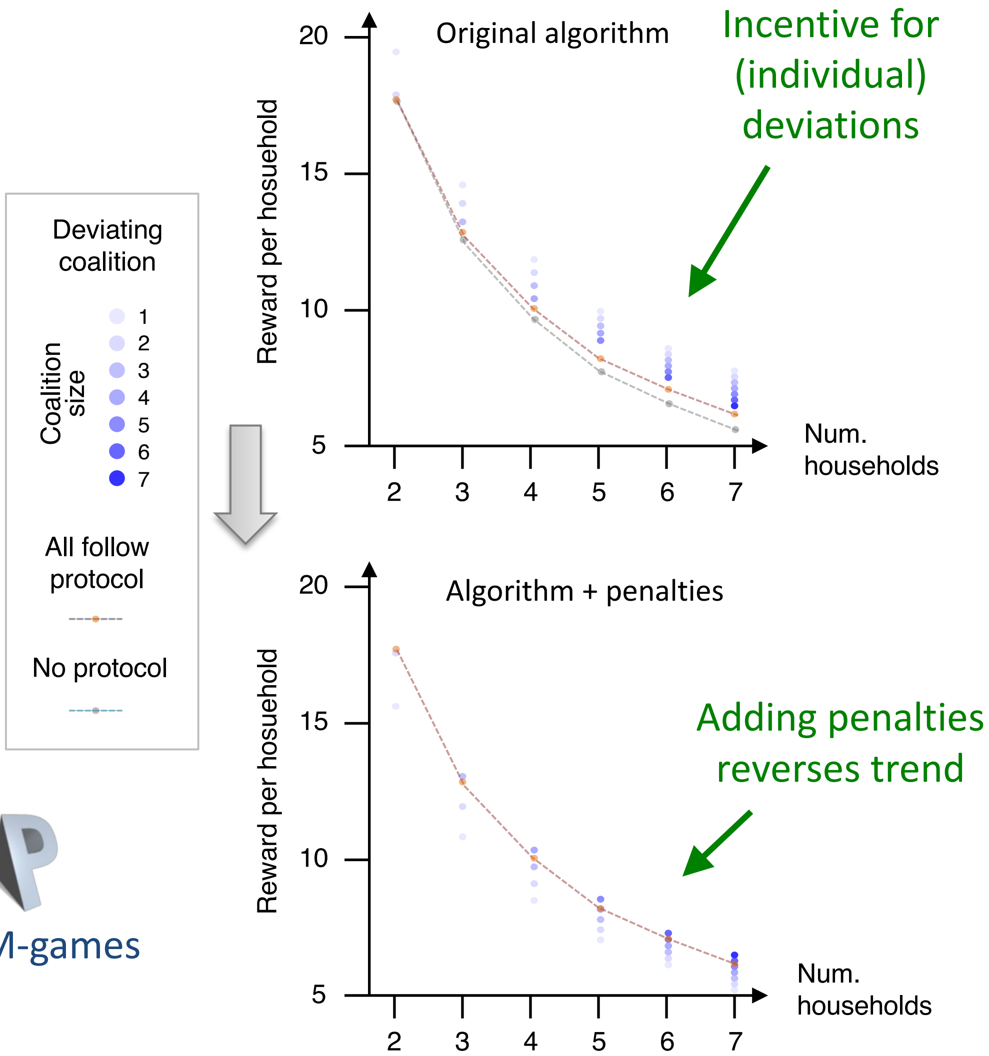
Example: Energy protocols

- 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

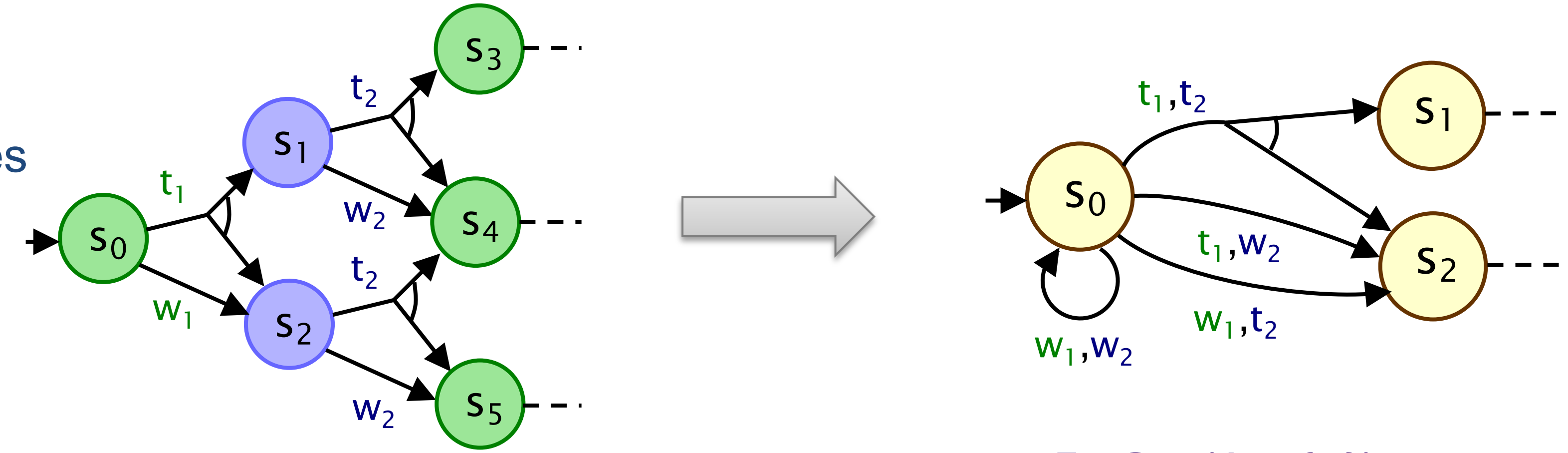


Concurrent stochastic games

- 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

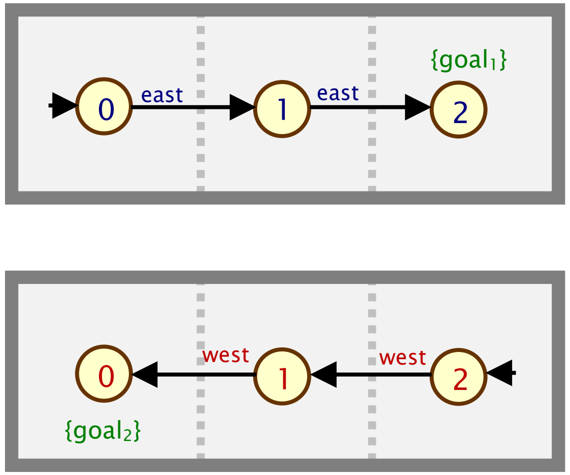
Turn-based stochastic games (TSGs)

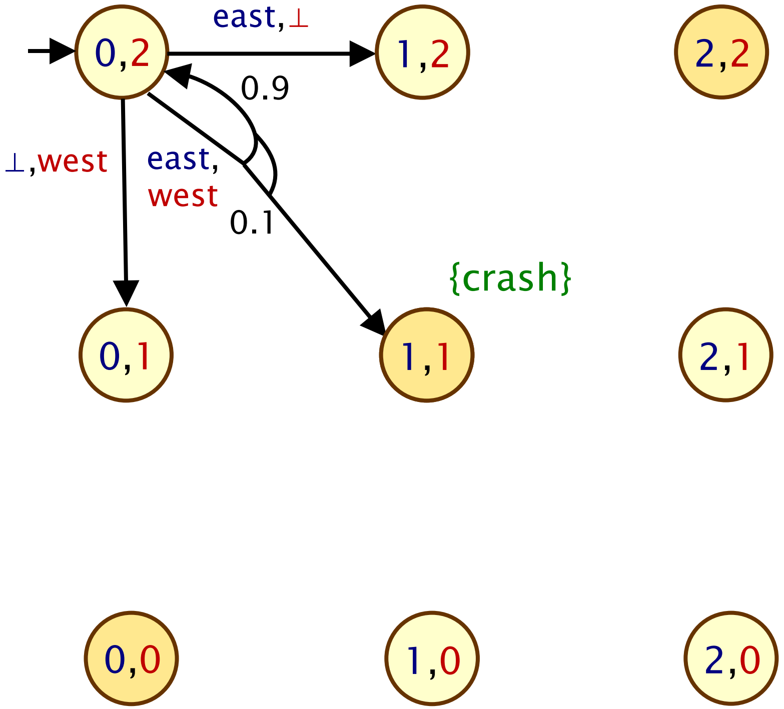


Concurrent stochastic games (CSGs)

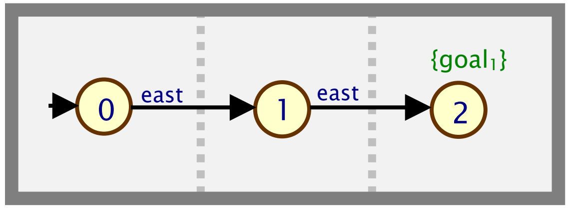
$\delta : S \times (A_1 \cup \{\perp\}) \times \ldots \times (A_n \cup \{\perp\}) \rightarrow Dist(S)$

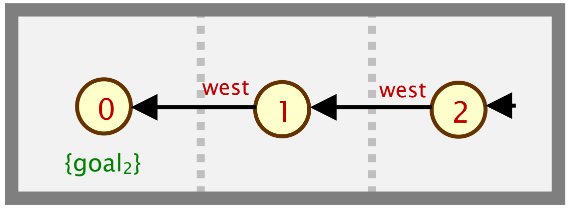
CSG for 2 robots on a 3x1 grid

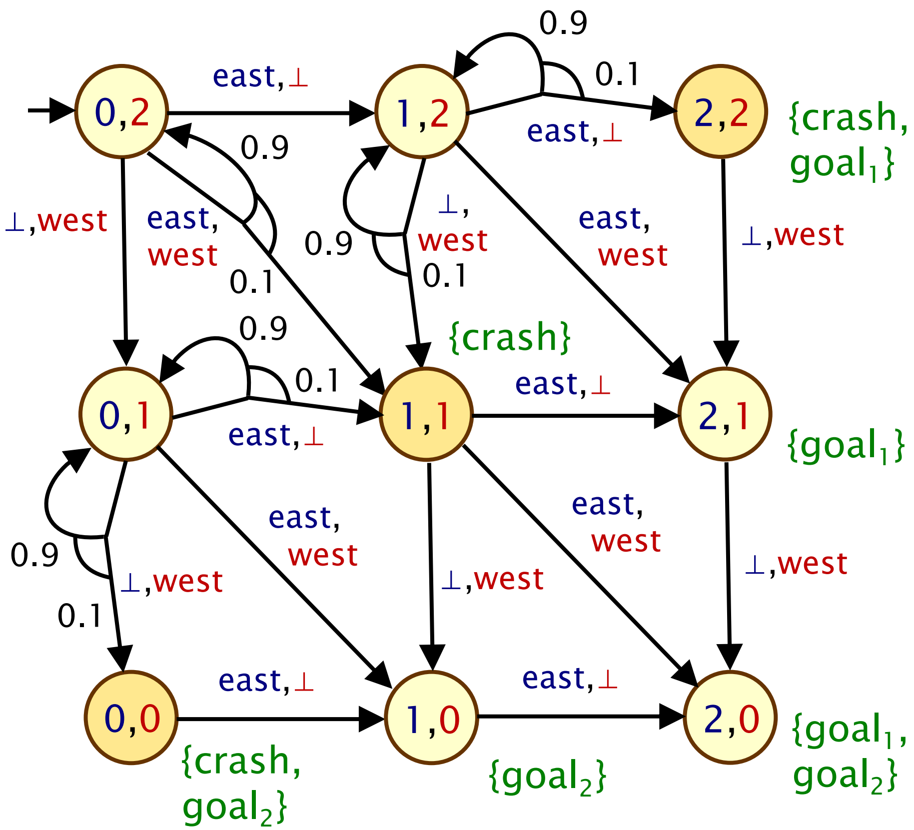




CSG for 2 robots on a 3x1 grid

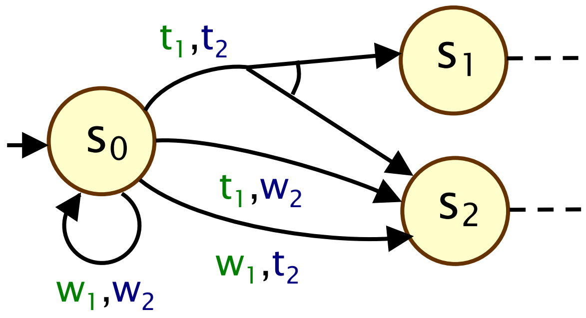






rPATL model checking for CSGs

- 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
  - $\sup_{\sigma_1} \inf_{\sigma_2} Pr_s^{\sigma_1,\sigma_2}(F\ ✓)$ for all states s
  - values $p(s)$ are the least fixed point of:

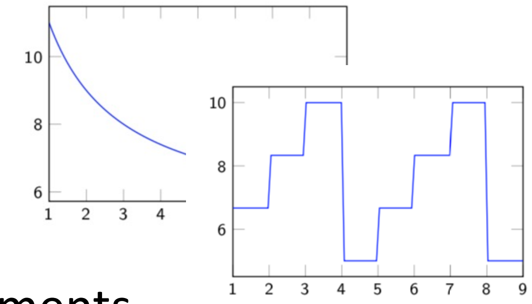
$$p(s) = \begin{cases} 1 & \text{if } s \models ✓ \\ val(Z) & \text{if } s \not\models ✓ \end{cases}$$

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

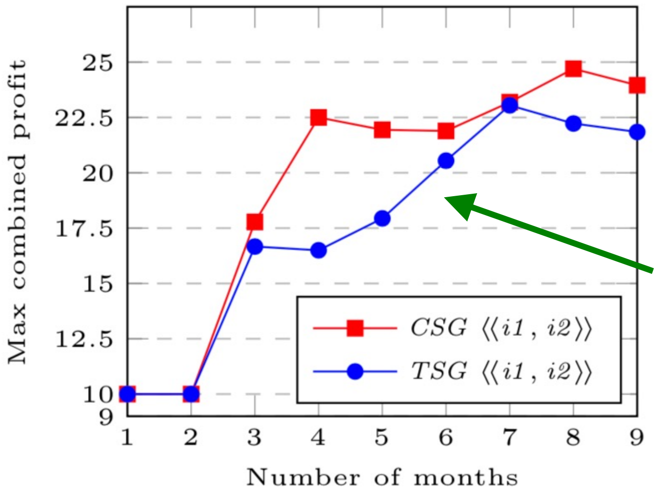
- Implementation
  - matrix games solved as linear programs
    - (LP problem of size |A|)
  - required for every iteration/state
    - which is the main bottleneck
  - but we solve CSGs of ~3 million states

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

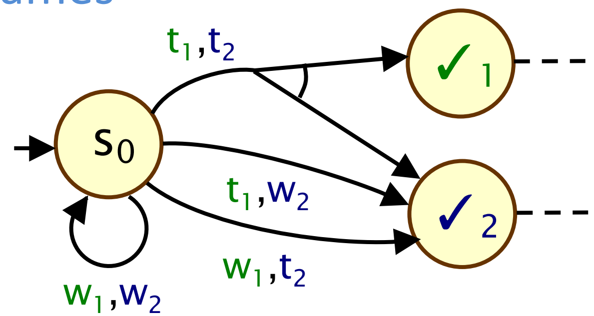
$\langle\langle robot_1 \rangle\rangle_{max=?}\ P\ [\ F^{\leq k}\ goal_1\ ]$

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

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



- We further extend the value iteration approach:

$$p(s) = \begin{cases} (1,1) & \text{if } s \models ✓_1 \wedge ✓_2 \\ (1, p_{max}(s, ✓_2)) & \text{if } s \models ✓_1 \wedge \neg ✓_2 \\ (p_{max}(s, ✓_1), 1) & \text{if } s \models \neg ✓_1 \wedge ✓_2 \\ val(Z_1, Z_2) & \text{if } s \models \neg ✓_1 \wedge \neg ✓_2 \end{cases}$$

standard MDP analysis

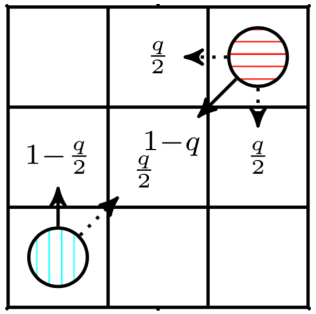
bimatrix game

  - where Z_1 and Z_2 encode matrix games similar to before

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

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

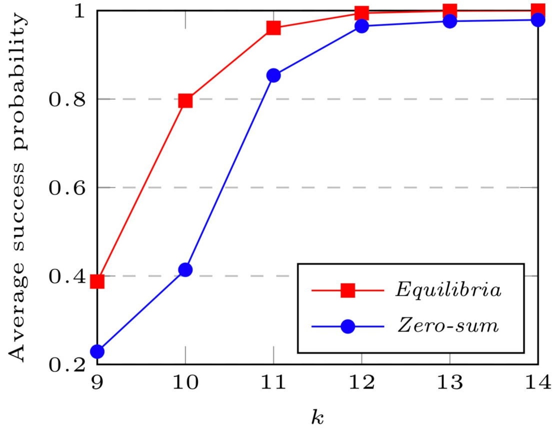


- We synthesise SWNEs to maximise the average probability of robots reaching their goals within time k
  - $\langle\langle robot1:robot2 \rangle\rangle_{max=?}$ (P [$F^{\leq k}$ $goal_1$]+P [$F^{\leq k}$ $goal_2$])
  - and compare to sequential strategy synthesis

Collaboration helps: better performance from equilibria

ε-NE found typically have ε=0

10 x 10 grid



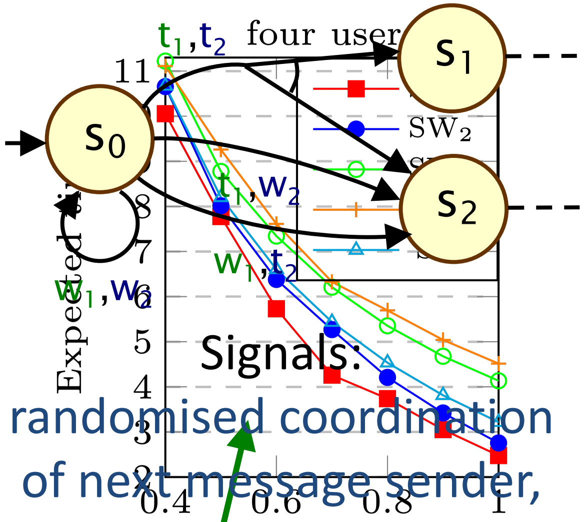
Faster and fairer equilibria

- Limitations of (social welfare) Nash equilibria for CSGs:
  1. can be computationally expensive, especially for >2 players
  2. social welfare optimality is <u>not</u> 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



Signals: randomised coordination of next message sender, adapting over time

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/

Overview

- Probabilistic model checking
  - key ideas, applications, trends

- Verification and control
  - Markov decision processes and beyond

- Multi-agent systems
  - probabilistic model checking with stochastic games
  - competitive or collaborative behaviour

- Robustness under model uncertainty
  - probabilistic model checking with epistemic uncertainty
  - robust guarantees for data-driven models

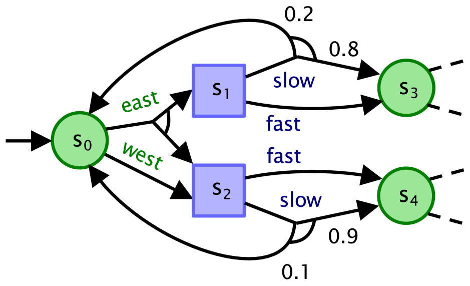
Robust verification

(under model uncertainty)

A zoo of probabilistic models

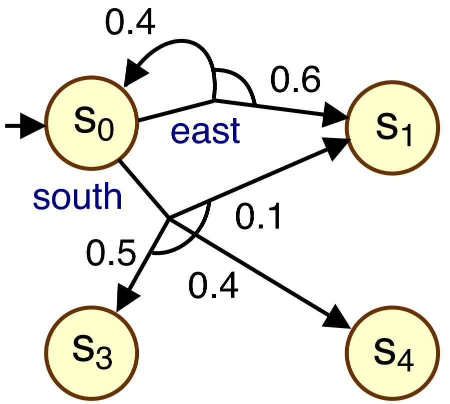
- Increasing variety (and complexity) of probabilistic models supported

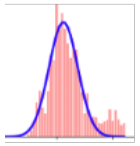
  - discrete-time Markov chains
  - probabilistic automata — + concurrency
  - continuous-time Markov chains — + exponential delays
  - Markov decision processes (MDPs) — + policies / control
  - probabilistic timed automata — + real-time clocks
  - partially observable MDPs — + observability
  - stochastic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

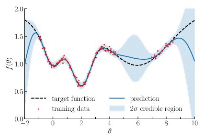


Reasoning about uncertainty

- Probabilistic model checking
  - formal quantitative guarantees on safety, performance, ...
  - based on rigorous modelling of uncertainty
- Caveat: must assume that models are correct/accurate
  - how do we build/learn accurate probabilistic models?
  - how do we factor in model uncertainty?







- 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

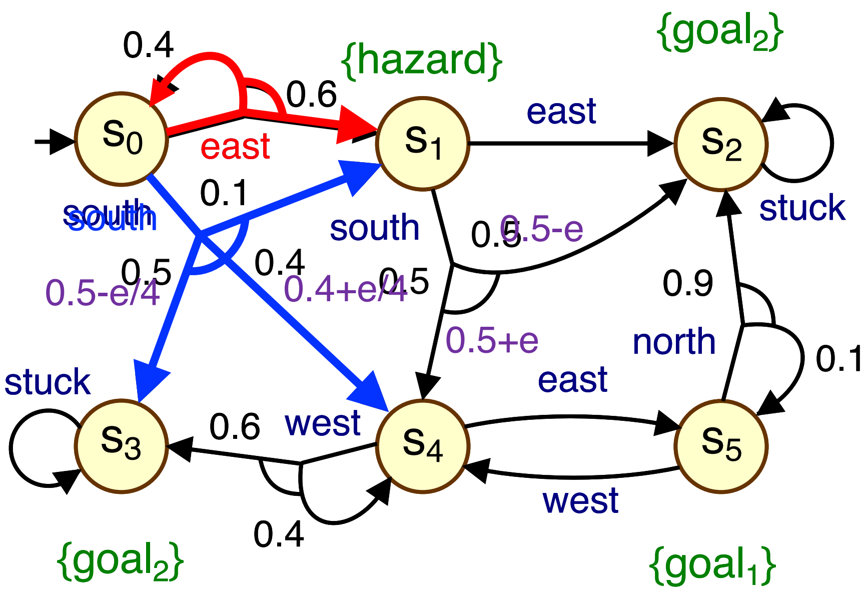
(2)

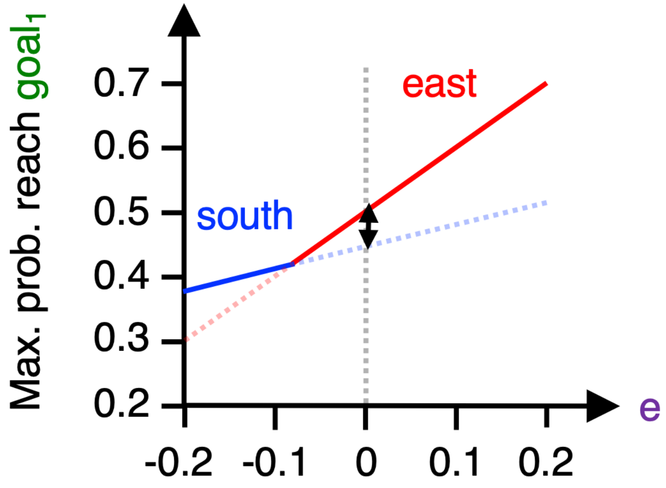
P(H)=1 P(T)=1

Impact of model uncertainty

- We focus on model uncertainty regarding transition probabilities
  - clearly this impacts the results (guarantees) obtained from model checking

- MDP policy optimality can also be sensitive to perturbations in probabilities
  - so “optimal” policies can in fact be sub-optimal



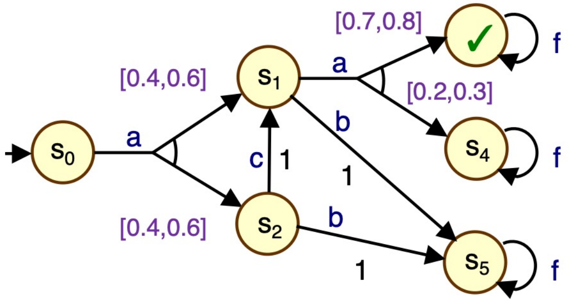


Uncertain MDPs

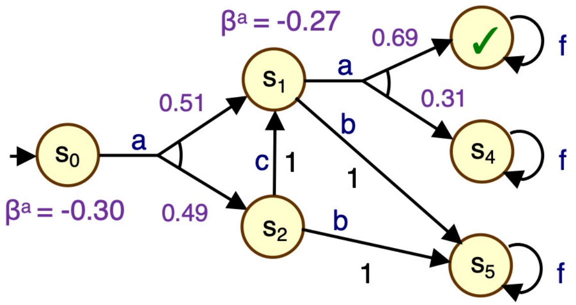
- An uncertain MDP (uMDP), also called a robust MDP
  - models both aleatoric and epistemic uncertainty
  - can be seen as an MDP with a set $\mathcal{P}$ of transition functions
  - i.e., each $\delta \in \mathcal{P}$ is of the form $\delta : S \times A \rightarrow Dist(S)$
  - we often specify separate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

- Some examples of uMDPs

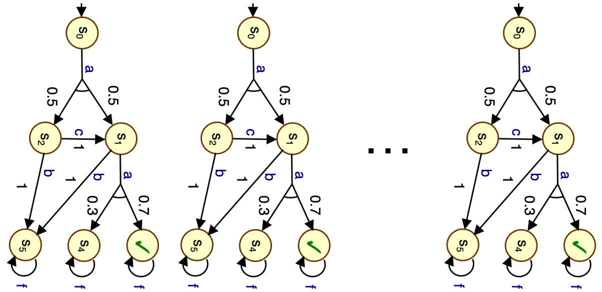
Interval MDPs (IMDPs)



Likelihood MDPs

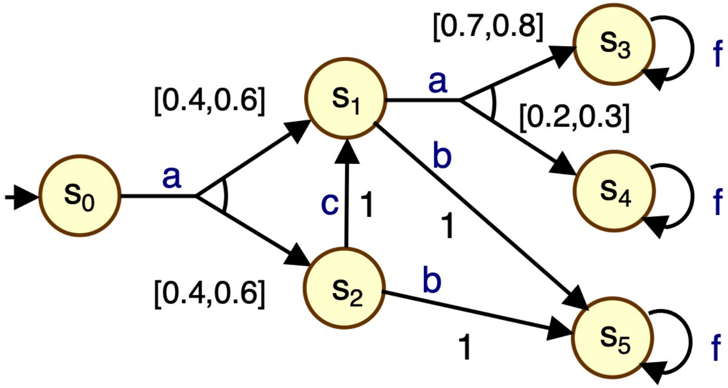


Sampled MDPs

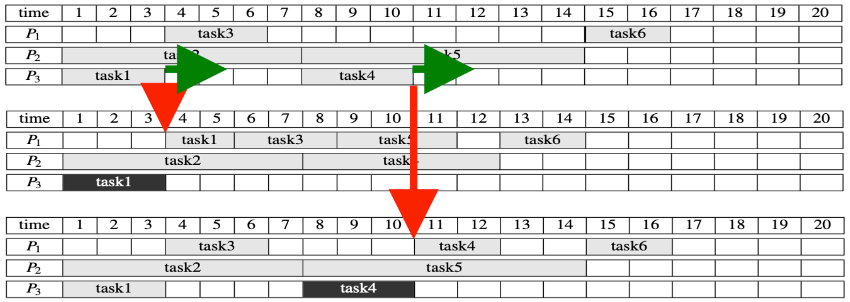


Uncertainty set dependencies

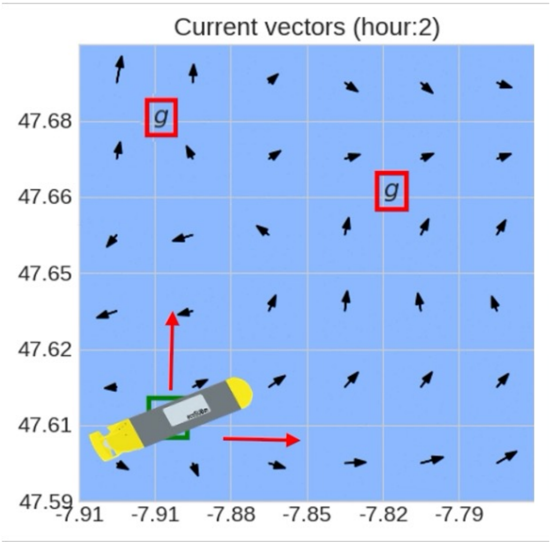
- We often assume (s,a)-rectangularity
  - no dependencies between uncertainty sets: $\mathcal{P} = \times_{(s,a)\in S\times A}\, \mathcal{P}_{s,a}$
  - computational tractability vs. modelling accuracy



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



Task scheduling in the presence of faulty processors

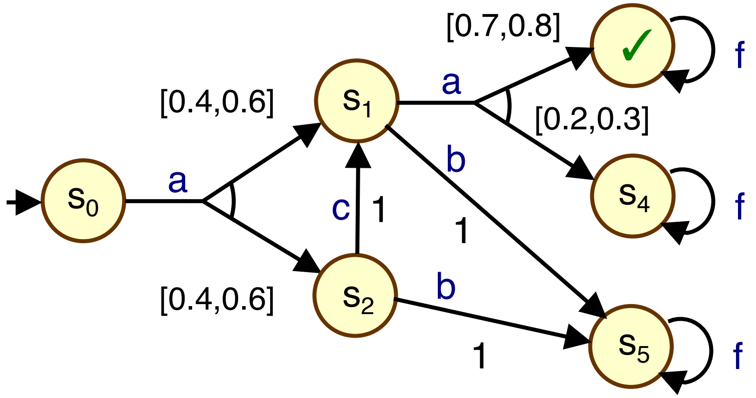


Underwater vehicle control in unknown ocean currents

Robust control

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

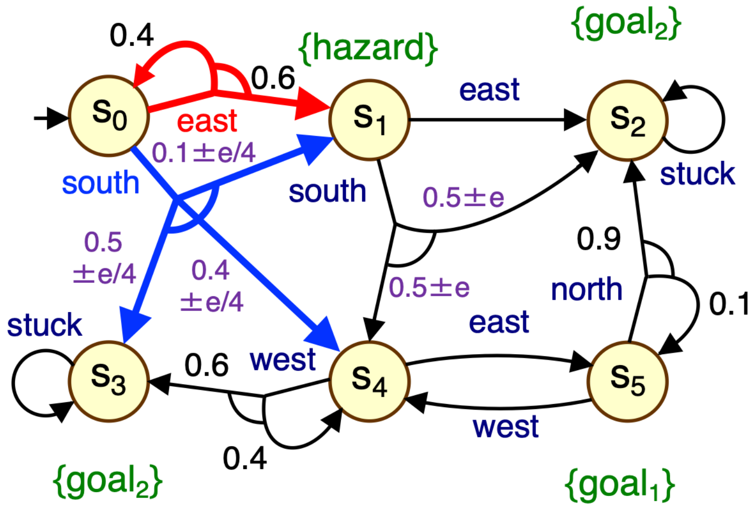
- Robust policy evaluation:
  - policies σ are defined as for MDPs
  - as are objectives e.g. $P_{max=?}$ [F ✓]
  - for a (maximising) policy σ:
  - worst-case value: $\inf_{\delta \in \mathcal{P}} Pr_s^{\delta,\sigma}$ (F ✓)



- 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 $\sigma^* = \mathrm{argsup}_\sigma \inf_{\delta \in \mathcal{P}} Pr_s^{\delta,\sigma}$ (F ✓)
  - p^* represents a robust guarantee, i.e., $P_{\geq p^*}$ [F ✓] always holds

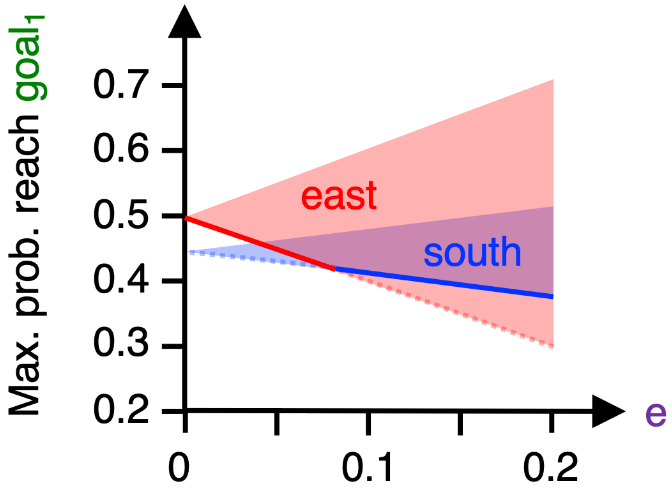
Running example: Robust control

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



  - Note: the degree of uncertainty (e) in states s1 and s2 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

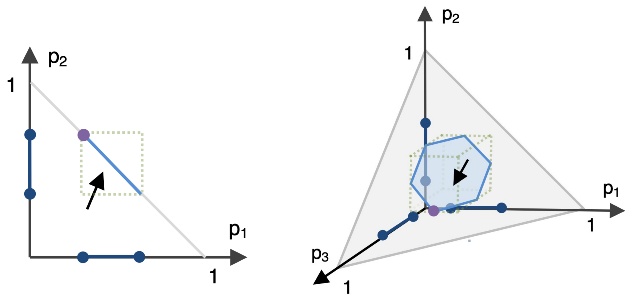


Robust control

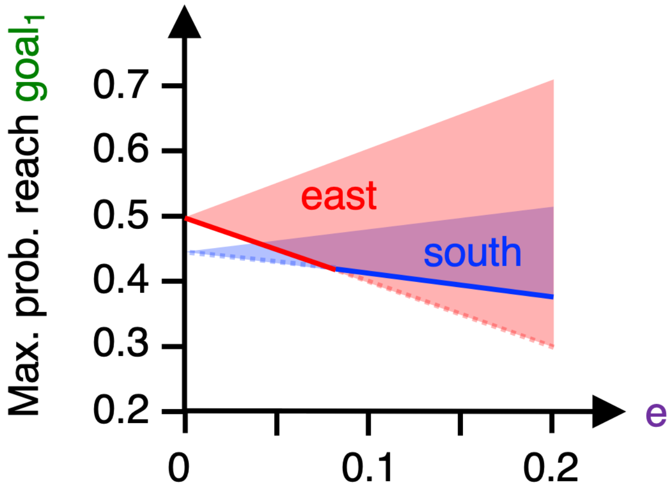
- Can be solved with robust value iteration

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

  - note the similarity to 2-player stochastic games
  - various techniques for solving inner optimisation problems



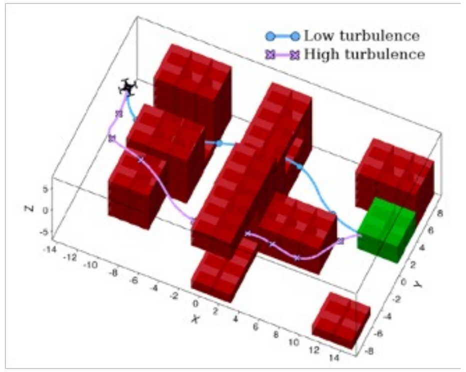
- Robust control
  - for any e, we can pick a "robust" (optimal worst-case) policy
  - and give a safe lower bound on its performance

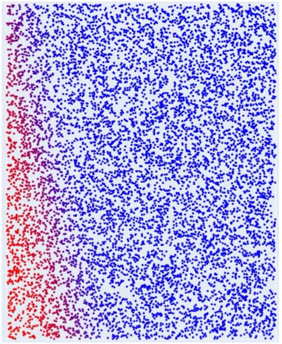


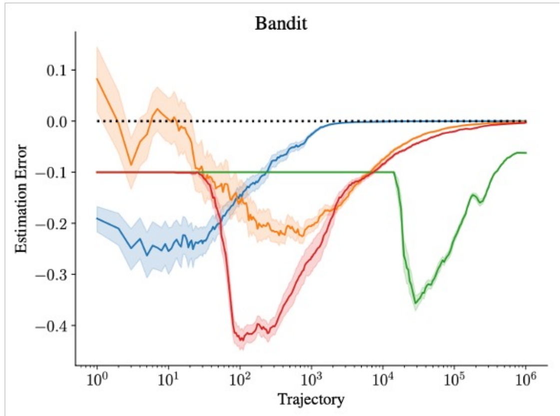
- Implemented/available in PRISM

Generating IMDP intervals

- Some examples of IMDP generation







- Unmanned aerial vehicle
  - robust control in turbulence
  - continuous-space dynamical model with unknown noise
  - discrete abstraction + finite “scenarios” of sampled noise yields IMDP abstraction

[AAAI’22 / JAIR’23]

- Deep reinforcement learning
  - worst-case analysis of abstractions of probabilistic policies for neural networks
  - intervals between IMDP abstract states constructed by sampling the policy

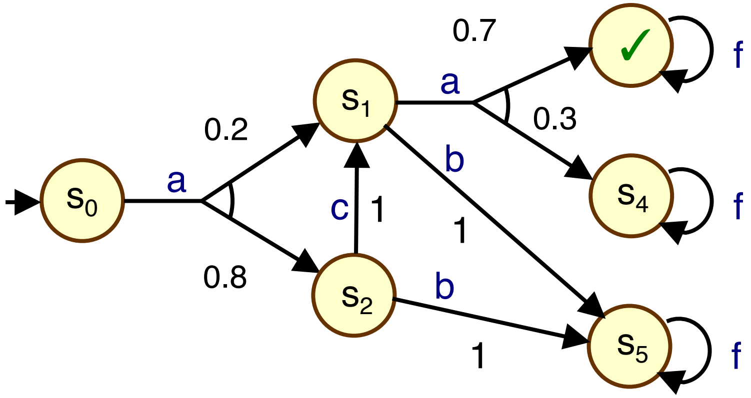
[FORMATS’20]

- Robust anytime MDP learning
  - sampled MDP trajectories
  - IMDPs constructed and solved periodically to yield robust predictions on current model
  - PAC or Bayesian interval learning

[NeurIPS’22]

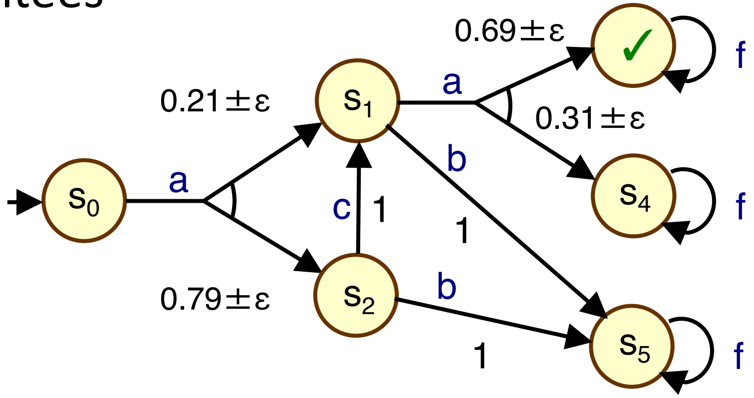
Learning IMDPs

- 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

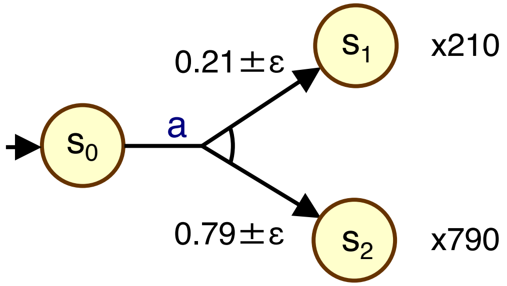
$$Pr(\delta \in \mathcal{P}) \geq 1 - \gamma$$



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: $\boxed{Pr(P_s^a(s_i) \in \tilde{P}_s^a(s_i) \pm \varepsilon) \geq 1-\gamma}$

(via Hoeffding's inequality)

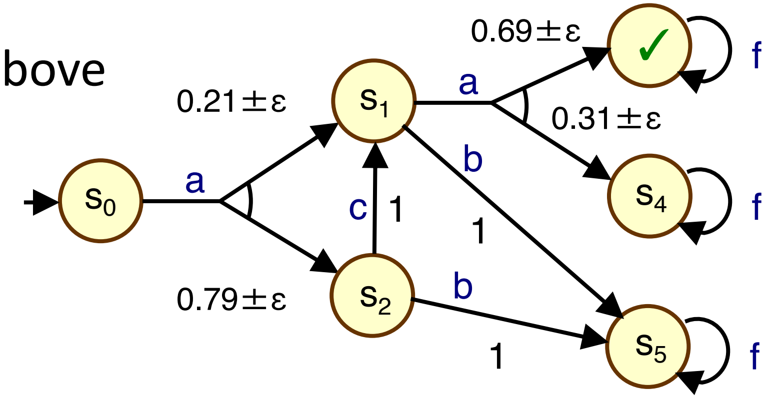


- We can lift this to the whole IMDP
  - building uncertain transition set $\mathcal{P}$ using intervals as above

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

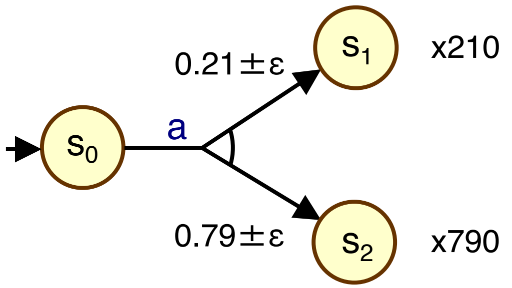
(after distributing error rate γ)

  - and also to our robust guarantees $\mathrm{P}_{\geq p^*}$ [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: $\boxed{Pr(P_s^a(s_i) \in \tilde{P}_s^a(s_i) \pm \varepsilon) \geq 1-\gamma}$



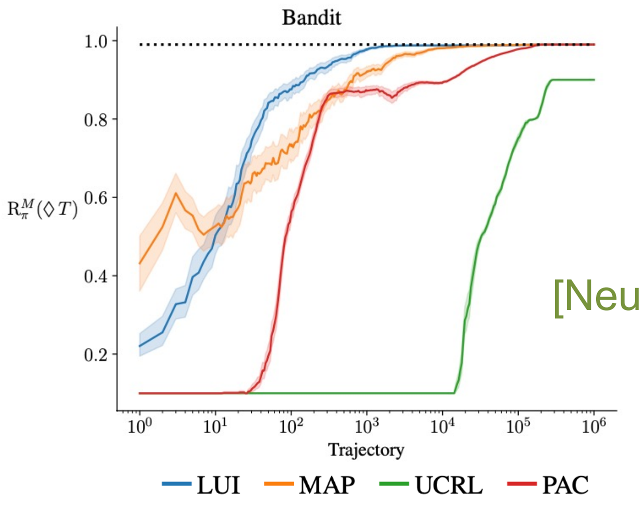
(via Hoeffding's inequality)

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

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

(after distributing error rate γ)

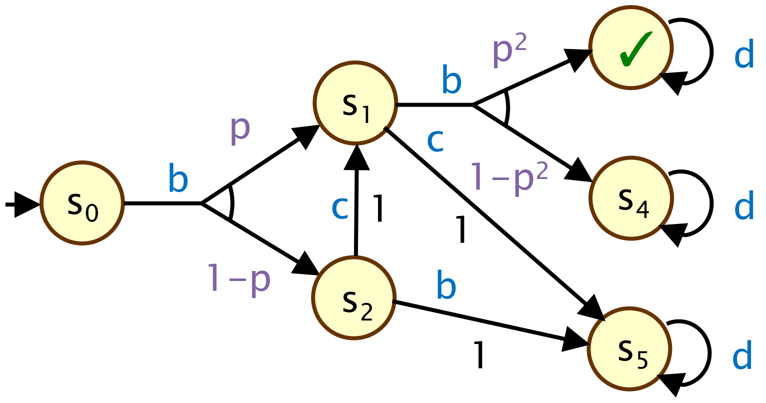
  - and also to our robust guarantees $P_{\geq p^*}$ [F ✓]



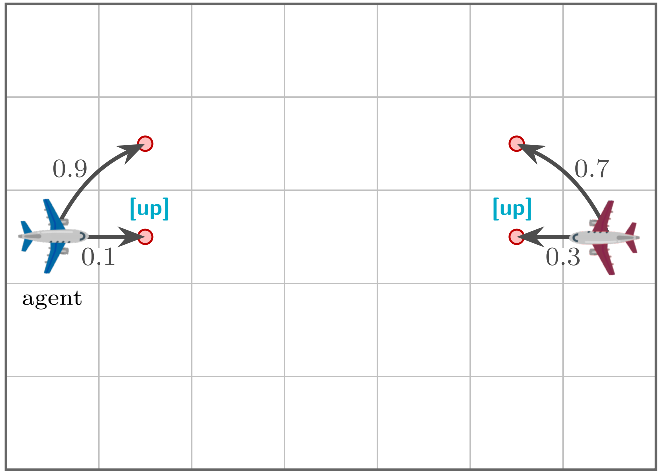
[NeurIPS'22]

Learning parameterised models

- Parametric MDPs (pMDPs)
  - $\delta_\Theta : \Theta \times S \times A \rightarrow \text{Dist}(S)$
  - for parameter space Θ
  - each $\theta \in \Theta$ yields an MDP



- Uncertain parametric MDPs (upMDPs)
  - pMDP + distribution $\mathbb{P}$ over parameter space Θ
  - yields a distribution over MDPs

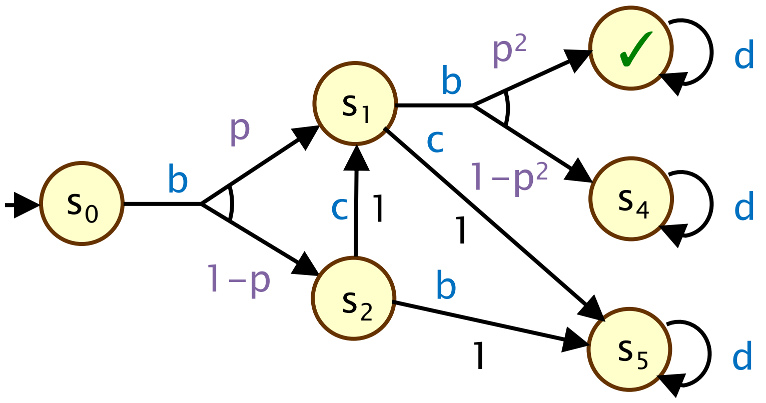


Example: Aircraft collision avoidance [Kochendorfer'15]

Parameters:
- navigation uncertainty
- pilot response factor

Learning parameterised models

- Learning upMDP models from data [TACAS'25]
  - (data = samples of transitions/trajectories)
  - two levels of uncertainty:
    - unknown parameter distribution $\theta \sim \mathbb{P}$
    - unknown transitions δ_θ

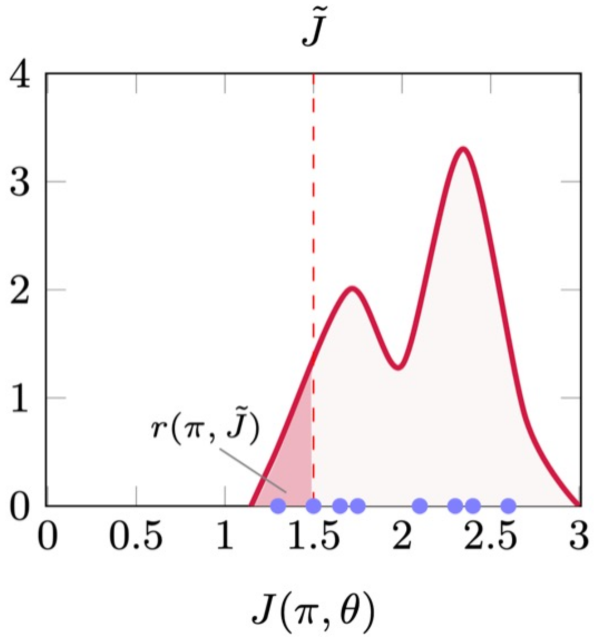


- Setup
  - performance function $J : \Pi \times \Theta \to \mathbb{R}$
    (e.g. probability of reaching ✓)
  - aim for PAC guarantee over risk associated with policy π and performance guarantee $\tilde{J}$

$$r(\pi, \tilde{J}) = \mathbb{P}\left\{\theta \in \Theta \colon J(\pi, \theta) < \tilde{J}\right\}$$

Learning parameterised models

- Learning upMDP models from data [TACAS'25]
  - (data = samples of transitions/trajectories)
  - two levels of uncertainty:
    - unknown parameter distribution $\theta \sim \mathbb{P}$
    - unknown transitions δ_θ



- Setup
  - performance function $J : \Pi \times \Theta \to \mathbb{R}$
    (e.g. probability of reaching ✓)
  - aim for PAC guarantee over risk associated with policy π and performance guarantee $\tilde{J}$

$$r(\pi, \tilde{J}) = \mathbb{P}\left\{\theta \in \Theta : J(\pi, \theta) < \tilde{J}\right\}$$

$$\boxed{\Pr\left\{r(\pi, \tilde{J}) \leq \varepsilon\right\} \geq 1 - \eta}$$

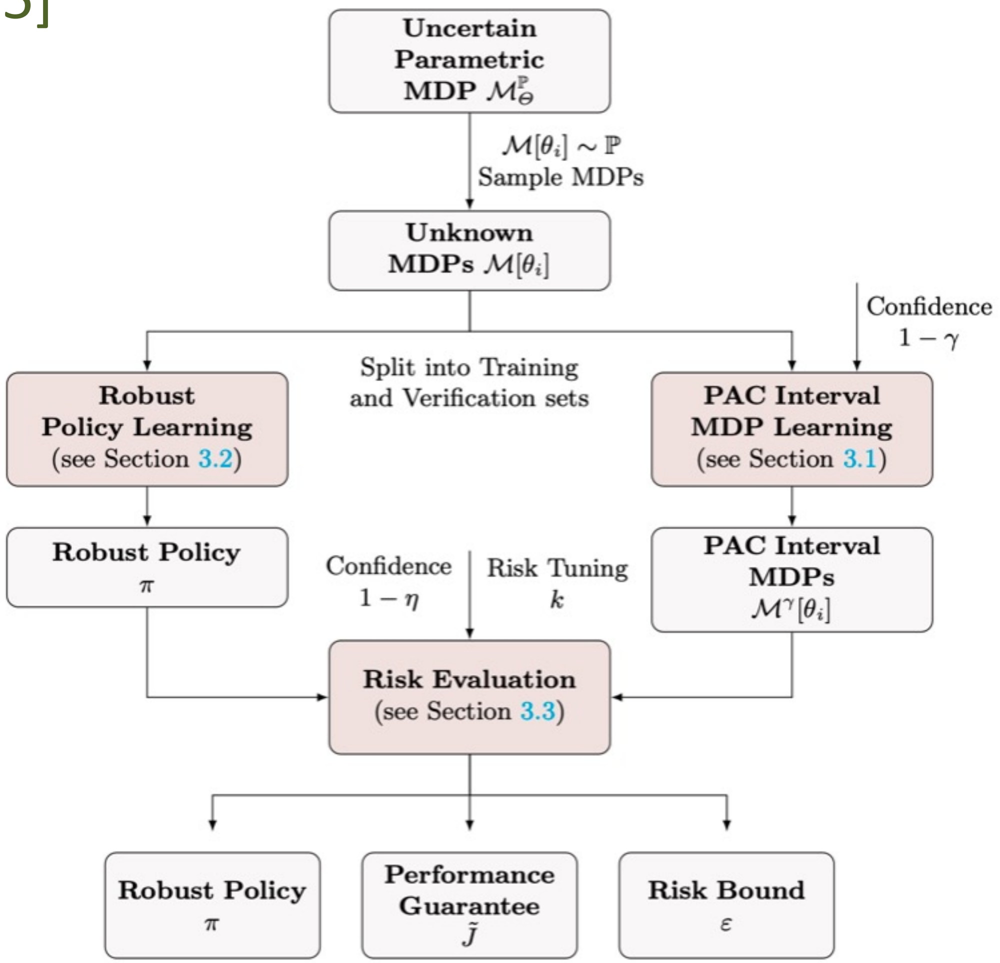
Risk   Performance Bound   Risk Bound   Confidence

Learning parameterised models

- Learning upMDP models from data [TACAS'25]
  - (data = samples of transitions/trajectories)
  - two levels of uncertainty:
    - unknown parameter distribution $\theta \sim \mathbb{P}$
    - unknown transitions δ_θ

- Setup
  - performance function $J : \Pi \times \Theta \to \mathbb{R}$
    (e.g. probability of reaching ✓)
  - aim for PAC guarantee over risk associated with policy π and performance guarantee $\tilde{J}$

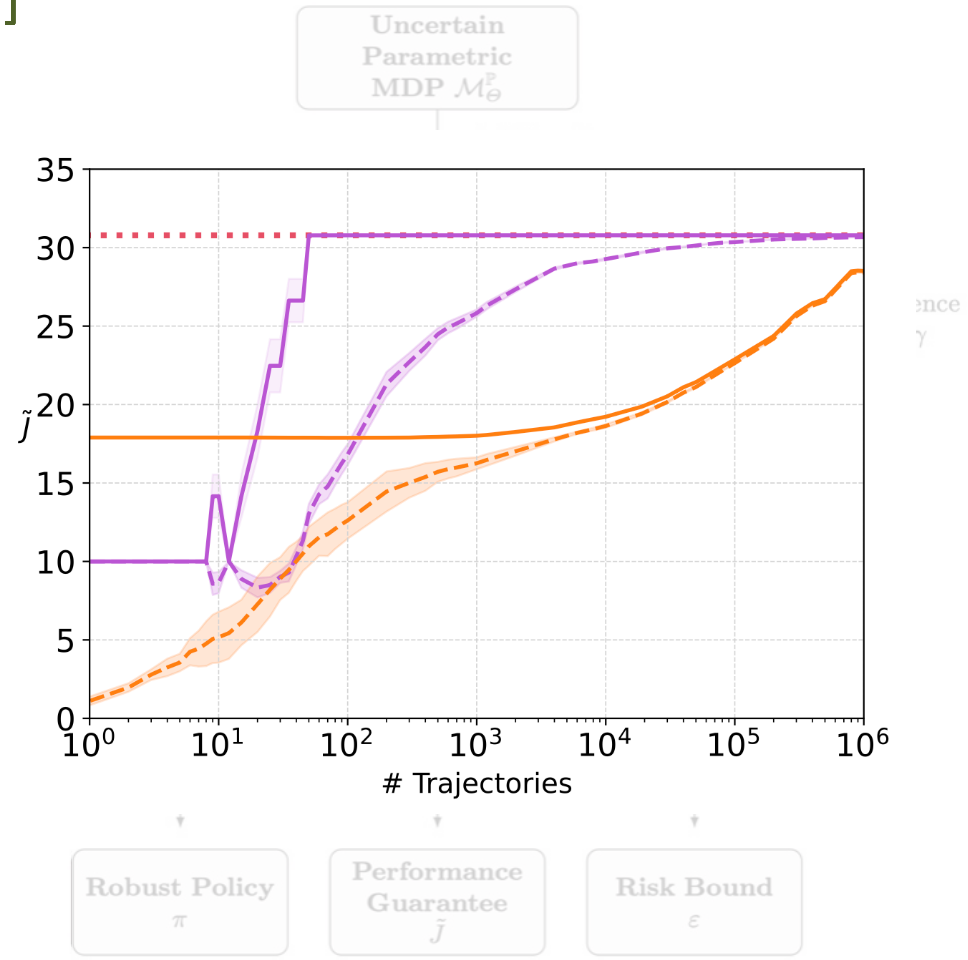
$$r(\pi, \tilde{J}) = \mathbb{P}\left\{\theta \in \Theta : J(\pi, \theta) < \tilde{J}\right\}$$



Learning parameterised models

- Learning upMDP models from data [TACAS'25]
  - (data = samples of transitions/trajectories)
  - two levels of uncertainty:
    - unknown parameter distribution $\theta \sim \mathbb{P}$
    - unknown transitions δ_θ
- Setup
  - performance function $J : \Pi \times \Theta \to \mathbb{R}$
    (e.g. probability of reaching ✓)
  - aim for PAC guarantee over risk associated with policy π and performance guarantee $\tilde{J}$

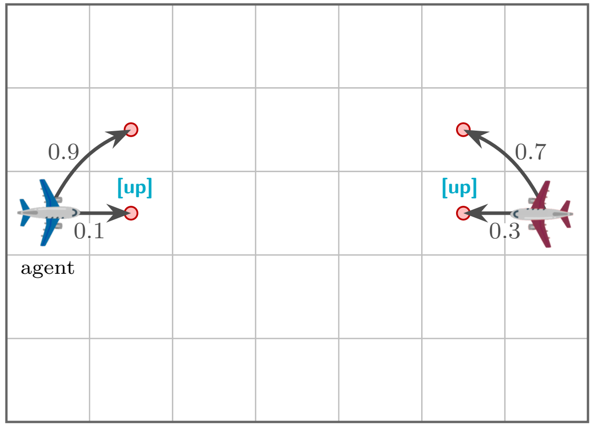
$$r(\pi, \tilde{J}) = \mathbb{P}\left\{\theta \in \Theta : J(\pi, \theta) < \tilde{J}\right\}$$

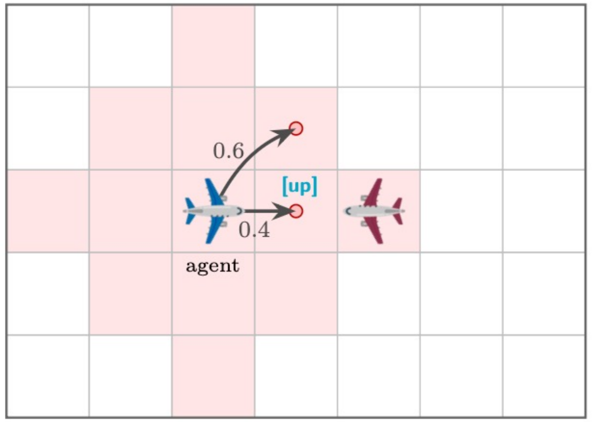


Learning factored models

- Aircraft collision avoidance
  - PRISM model excerpt

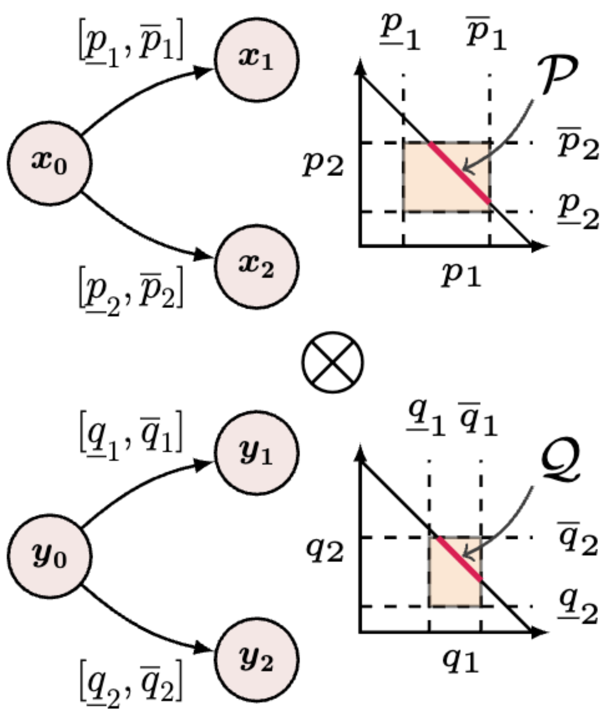
```
module ownship
  x : [0..X] init 0   // horizontal position
  y : [0..Y] init 0   // altitude

  // [up]: dependency on closeness to intruder
  [up] ||(x,y) - (ix,iy)||_1 >= 5 ->  0.9 : (y' = y + 1)
                                    + 0.1 : (y' = y)
  [up] ||(x,y) - (ix,iy)||_1 < 5  ->  0.6 : (y' = y + 1)
                                    + 0.4 : (y' = y)
  ...
endmodule

module intruder
  ix : [0..IX] init IX // intruder horizontal position
  iy : [0..IY] init 0  // intruder altitude

  // [up]: no dependency on ownship
  [up] iy < 10 -> 0.7 : (iy' = iy + 1)
                + 0.3 : (iy' = iy)
  ...
endmodule
```

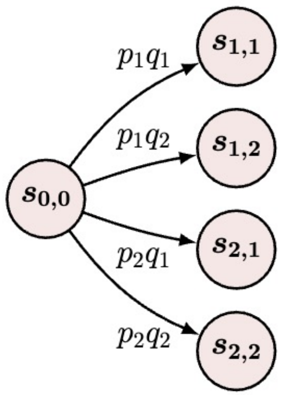


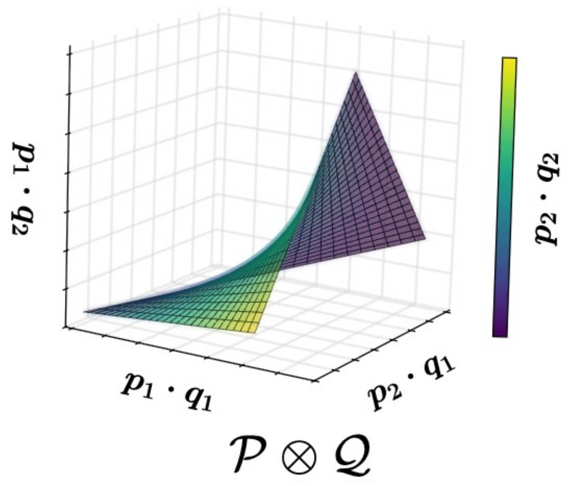


Learning factored models

- Factored models provide compositional modelling
  - but compositional uncertainties are non-convex and computationally complex to solve



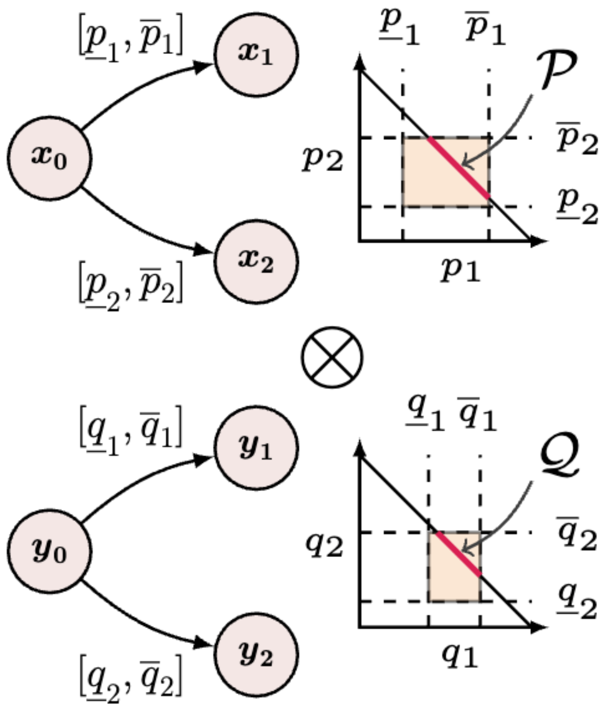


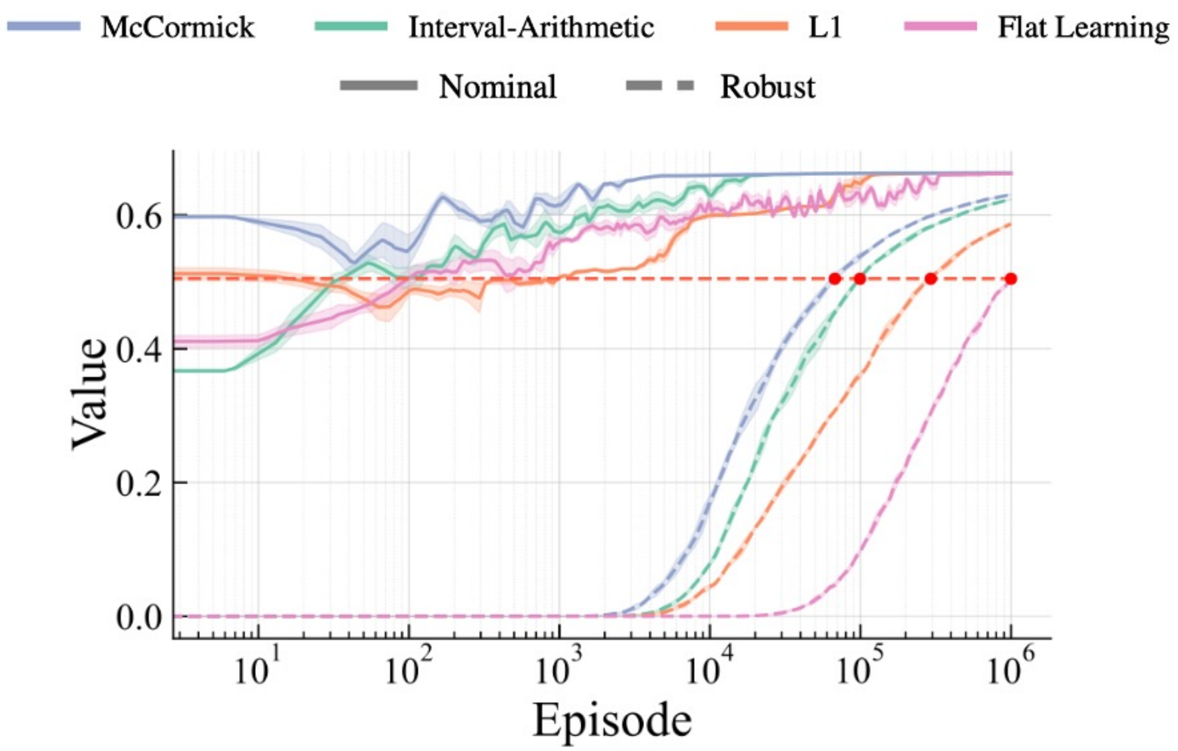


$$p(s) = \begin{cases} 1 & \text{if } s \models \checkmark \\ \max_a \min_{\delta \in \mathcal{P}_{s,a}} \Sigma_{s'}\, \delta(s,a)(s') \cdot p(s') & \text{if } s \not\models \checkmark \end{cases}$$

Learning factored models

- Factored models provide compositional modelling
  - but compositional uncertainties are non-convex and computationally complex to solve
  - we introduce multiple relaxation approaches to optimise learning [AAAI’26]





Wrapping up

Overview

- Probabilistic model checking
  - key ideas, applications, trends

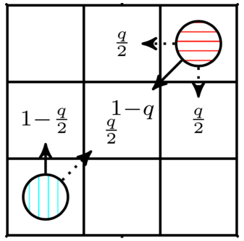
- Verification and control
  - Markov decision processes and beyond

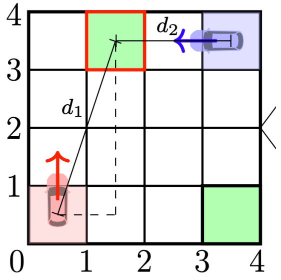
- Multi-agent systems
  - probabilistic model checking with stochastic games
  - competitive or collaborative behaviour

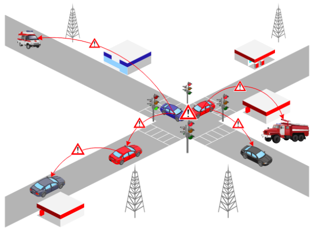
- Robustness under model uncertainty
  - probabilistic model checking with epistemic uncertainty
  - robust guarantees for data-driven models

Challenges & directions

- Partial information/observability
  - e.g., POMDPs + PO stochastic games



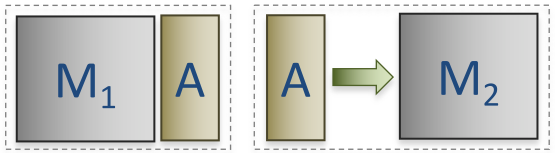




- Further classes of equilibria
  - e.g. Stackelberg equilibria for automotive/security applications
  - robust equilibria for epistemic uncertainty

```
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
```

- Modelling language design and extensions
  - e.g., for specifying epistemic uncertainty
  - e.g., more flexible interchange of components and strategies



- Improving scalability & efficiency
  - e.g. symbolic methods for CSGs, compositional solution

prismmodelchecker.org