Multi-Agent Verification & Control with Probabilistic Model Checking

Dave Parker

University of Oxford

QEST @ CONFEST, Antwerp, Sep 2023
Probabilistic model checking

• Models & logics for automatic verification of stochastic systems

• Builds on an (increasingly) wide range of disciplines
  ▪ logic, automata, Markov models, optimisation, SMT, simulation, control, AI, ...

• Key strengths: exhaustive + numeric analysis
  ▪ often subtle interplay between probability + nondeterminism
  ▪ numerical results & trends can help identify flaws
  ▪ enabled by advances in scalability, e.g., symbolic (BDD-based) methods

• Exploits flexibility of formal modelling languages & logics
  ▪ consistency across wide range of models & properties

\[ P > 0.999 \] \( (\square \text{trigger} \rightarrow \Diamond^{\leq 20} \text{deploy}) \)
Example: Bluetooth

- Device discovery between a pair of Bluetooth devices
  - performance 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
  - 17,179,869,184 initial configurations
  - exhaustive numerical analysis via symbolic model checking
  - highlights flaws in a simpler, analytic analysis
• Increasingly expressive/powerful classes of model
  ▪ real-time, partial observability, epistemic uncertainty, multi-agent, ...
  ▪ leading to ever widening range of application domains

• 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
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

- This talk:
  - probabilistic model checking with stochastic multi-player games
  - models, logics, algorithms, tools, examples
Overview

- Stochastic multi-player games
- Concurrent stochastic games
- Equilibria for stochastic games
- Neuro-symbolic games
- Challenges & directions
Stochastic games
Markov decision processes (MDPs)

- Strategies (or policies) $\sigma$ resolve actions based on history
- e.g.: $P_{\text{max}=?}[F\checkmark] = \sup_\sigma \Pr_\sigma[F\checkmark]$
- What is the maximum probability of reaching $\checkmark$ achievable by any strategy $\sigma$?

Key solution method: value iteration

- Values $p(s)$ are the least fixed point of:
  
  $$p(s) = \begin{cases} 
  1 & \text{if } s \models \checkmark \\
  \max_a \Sigma_{s'} \delta(s,a)(s') \cdot p(s') & \text{otherwise}
  \end{cases}$$

- Also amenable to symbolic (BDD-based) implementation
(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)

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

$S = S_1 \cup \ldots \cup S_n$
• Turn-based stochastic games well suited to some (but not all) scenarios

TSG models

Uncontrollable/unknown navigation interference

Shared autonomy: human-robot control

Player 1

Player 2
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:
  - $\langle\langle \{\text{robot}_1, \text{robot}_3\} \rangle\rangle P_{\text{max}=?} [ F (\text{goal}_1 V \text{goal}_3) ]$
  - “what strategies for robots 1 and 3 maximise the probability of reaching their goal locations, regardless of the strategies of other players”

- Other additions:
  - (co-safe) linear temporal logic
    $\neg \text{zone}_3 U (\text{room}_1 \wedge (F \text{room}_4 \wedge F \text{room}_5))$
  - nested specifications
    $\langle\langle \{\text{robot}_1, \text{robot}_3\} \rangle\rangle R_{\text{min}=?} [\langle\langle \{\text{robot}_1\} \rangle\rangle P_{\geq 0.99} [ F^{\leq 10} \text{base} ] U (\text{zone}_1 \wedge (F \text{zone}_4)) ]$
    “minimise expected time for joint task, while ensuring base reliably reached”

Can be seen as a mixture of control and verification
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: $\sup_{\sigma_1} \inf_{\sigma_2} \Pr_s^{\sigma_1, \sigma_2} (F^\Box)$
  - complexity: $\text{NP} \cap \text{coNP}$ (if we omit some reward operators)

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

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

- 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 + rPATL
  - allow users to collaboratively cheat (ignore protocol)
  - TSGs of up to ~6 million states
  - exposes protocol weakness (incentive for clients to act selfishly)
  - propose/verify simple fix using penalties

Incentive for (individual) deviations

Adding penalties reverses trend
Concurrent stochastic 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

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

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

- 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 \( \sim 3 \) 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
  - $\langle\langle\text{investor}_1,\text{investor}_2\rangle\rangle \ R_{\text{max}}^{\text{profit}_{1,2}} = ?$ [ $F \text{ finished}_{1,2}$ ]
  - non-trivial optimal (randomised) investment strategies
  - concurrent game (CSG) yields more realistic results
    (market has less observational power over investors)

![Graph showing profit over months for CSG and TSG strategies]

Too pessimistic: unrealistic strategy for adversary
Equilibria for stochastic games
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 $\varepsilon$-NE, which always exist for CSGs

$$\sigma=(\sigma_1,...,\sigma_n) \text{ is an } \varepsilon\text{-NE for objectives } X_1,...,X_n \iff$$
$$\text{for all } i : E_s \sigma (X_i) \geq \sup \{ E_s \sigma' (X_i) \mid \sigma' = \sigma_i[\sigma_i'] \text{ and } \sigma_i' \in \Sigma_i \} - \varepsilon$$

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

Zero-sum properties

$$\langle \langle \text{robot}_1 \rangle \rangle_{\max=?} P [ F^{\leq k} \text{goal}_1 ]$$

Equilibria-based properties (SWNE)

$$\langle \langle \text{robot}_1: \text{robot}_2 \rangle \rangle_{\max=?} (P [ F^{\leq k} \text{goal}_1 ] + P [ F^{\leq k} \text{goal}_2 ])$$
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:
  - where $Z_1$ and $Z_2$ encode matrix games similar to before

\[\begin{align*}
p(s) &= \begin{cases} 
(1,1) & \text{if } s \models \checkmark_1 \land \checkmark_2 \\
(1,p_{\max}(s,\checkmark_2)) & \text{if } s \models \checkmark_1 \land \neg \checkmark_2 \\
(p_{\max}(s,\checkmark_1),1) & \text{if } s \models \neg \checkmark_1 \land \checkmark_2 \\
\text{val}(Z_1,Z_2) & \text{if } s \models \neg \checkmark_1 \land \neg \checkmark_2
\end{cases}
\end{align*}\]

- 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 $m \times m$ 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 $k$
  - $\langle \langle \text{robot1:robot2} \rangle \rangle_{\text{max}} = \max (P[F^k_{\text{goal}_1}] + P[F^k_{\text{goal}_2}])$
  - and compare to sequential strategy synthesis

Collaboration helps: better performance from equilibria

$\varepsilon$-NE found typically have $\varepsilon=0$
• 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
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

- 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/
Neuro-symbolic games
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

upper bounds on failure probabilities for initial regions
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 $E$
    - $S = (\text{Loc}_1 \times \text{Per}_1) \times (\text{Loc}_2 \times \text{Per}_2) \times S_E$
  - agents use a (learnt) perception function to observe $E$
    - $\text{obs}_i : (\text{Loc}_1 \times \text{Loc}_2) \times S_E \rightarrow \text{Per}_i$
  - 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
Challenges & directions

• Partial information/observability
  ▪ e.g., leveraging progress on POMDPs

• Managing robustness and uncertainty
  ▪ quantifying model uncertainty, e.g., from learning
  ▪ stability of randomised strategies

• Modelling language design and extensions
  ▪ 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 approaches
Challenges & directions

- Partial information/observability
  - e.g., leveraging progress on POMDPs

- Managing robustness and uncertainty
  - quantifying model uncertainty, e.g., from learning
  - stability of randomised strategies

- Modelling language design and extensions
  - 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 approaches

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

More details here:

Funded by: FUN2MODEL

Joint work with:
- Edoardo Bacci, Taolue Chen, Vojtěch Forejt, Marta Kwiatkowska, Gethin Norman, Gabriel Santos, Aistis Simaitis, Rui Yan

More details here:

PRISM-games

prismmodelchecker.org/games/