Verification with Stochastic Games: Advances and Challenges

Dave Parker

University of Birmingham

iFM 2020
Verification with Stochastic Games: Advances and Challenges

Dave Parker
University of Birmingham

Joint work with:
Gabriel Santos, Gethin Norman, Marta Kwiatkowska, ...

ERC Advanced Grant FUN2MODEL
Verification of stochastic systems

- Formal verification needs **stochastic modelling**

  - faulty sensors/actuators
  - unpredictable/unknown environments
  - randomised protocols
Verification with stochastic games

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

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

- **Probabilistic model checking**
  - Markov decision processes (MDPs)
  - example: robot navigation

- **Stochastic multi-player games**
  - example: energy management
  - Concurrent stochastic games
    - example: investor models

- **Equilibria-based properties**
  - example: multi-robot coordination

- **Future challenges**
Probabilistic model checking

- **System**
- **High-level model/design**
- **System requirements**
- **Specification (temporal logic)**
- **Probabilistic model**
- **Probabilistic model checker**

Result: 

- $P \geq 0.999$ [ $F \leq 20$ deploy ]

0.5
0.1
0.4
Probabilistic model checking

Probabilistic model checking

Numerical results/analysis

Result

Strategies/policies/controllers

Probabilistic model

Strategy synthesis

Probabilistic model checker

0.5
0.4
0.1
Probabilistic models

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

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

• **Formally:**
  – we write: \( \sup_{\sigma} \Pr_s^{\sigma}(F ✓) \)
  – where \( \Pr_s^{\sigma} \) denotes the probability from state s under strategy σ
Solving MDPs

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

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

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

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

- basis for iterative numerical computation
Example: Robot navigation

- Robot planning with probabilistic guarantees
  - **MDP** models navigation in (learnt) uncertain environment
  - **temporal logic** for formal robot task specification
    - \(-\text{zone}_3 \cup (\text{room}_1 \land (F \text{ room}_4 \land F \text{ room}_5))\) (co-safe LTL)
  - **strategy synthesis** performed to generate controllers
    - also: costs & rewards, multi-objective,..
  - **PRISM** built into a ROS module
    - 100s of hrs of autonomous robot deployment

[IROS'14, IJCAI'15, ICAPS'17, IJRR'19]
Stochastic games
Stochastic multi-player games

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

Markov decision processes (MDPs)

Turn-based stochastic games (TSGs)
Property specification: rPATL

• **rPATL** (reward probabilistic alternating temporal logic)
  – branching-time temporal logic for stochastic games

• **CTL, extended with:**
  – coalition operator $\langle\langle C\rangle\rangle$ of ATL
  – probabilistic operator $P$ of PCTL
  – generalised (expected) reward operator $R$ from PRISM

• **In short:**
  – zero-sum, probabilistic reachability + expected total reward

• **Example:**
  – $\langle\langle\{\text{robot}_1,\text{robot}_3\}\rangle\rangle \ P_{>0.99} [ \ F_{\leq 10} (\text{goal}_1 \lor \text{goal}_3) ]$
  – “robots 1 and 3 have a strategy to ensure that the probability of reaching the goal location within 10 steps is $>0.99$, regardless of the strategies of other players”
**Model checking rPATL**

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

- **We again use value iteration**
  - values $p(s)$ are the least fixed point of:
    $$p(s) = \begin{cases} 
    1 & \text{if } s \Vdash \checkmark \\
    \max_a \Sigma_s \delta(s,a)(s') \cdot p(s') & \text{if } s \not\Vdash \checkmark \text{ and } s \in S_1 \\
    \min_a \Sigma_s \delta(s,a)(s') \cdot p(s') & \text{if } s \not\Vdash \checkmark \text{ and } s \in S_2 
    \end{cases}$$
  - and more: graph-algorithms, sequences of fixed points, ...
• **PRISM-games:** prismmodelchecker.org/games
  – extension of PRISM modelling language (see later)
  – implementation in explicit engine
  – prototype symbolic (MTBDD) version also available

• **Example application domains**
  – security: attack–defence trees; DNS bandwidth amplification
  – self–adaptive software architectures
  – autonomous urban driving
  – human-in-the-loop UAV mission planning
  – collective decision making and team formation protocols
  – energy management protocols
Example: Energy management

• Demand management protocol for microgrids
  – random back-off to minimise peaks

• Stochastic game model + rPATL
  – exposes protocol weakness
    (incentive for clients to act selfishly)
  – propose/verify simple fix using penalties

Adding penalties reverses trend

Strong incentive to deviate
Concurrent stochastic games
Concurrent stochastic games

• Motivation:
  – more realistic model of components operating concurrently, making action choices **without** knowledge of others
Concurrent stochastic games

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

- We again use the logic rPATL for properties

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

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

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

- where \( Z \) is the matrix game with \( z_{ij} = \Sigma_s \delta(s,(a_i,b_j))(s') \cdot p(s') \)
- so each iteration requires solution of a matrix game for each state (LP problem of size \( |A| \), where \( A = \) action set)
CSGs in PRISM–games

• CSG model checking implemented in PRISM–games 3.0

• Extension of PRISM modelling language
  – (see next slide)

• Explicit engine implementation
  – plus LP solvers for matrix game solution
  – this is the main bottleneck
  – experiments with CSGs up to ~3 million states

• Case studies:
  – future markets investor, trust models for user–centric networks, intrusion detection policies, jamming radio systems
CSGs in PRISM–games 3.0

Example model (medium access control)

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
CSGs in PRISM-games 3.0

Each player comprises one or more modules

Players have distinct actions, executed simultaneously

---

```plaintext
module user1
  s1 : [0..1] init 0; // has player 1 sent?
  e1 : [0..emax] init emax, // energy level of player 1
  [w1] true -> (s1'=0); // wait
  [t1] e1>0 -> (s1'=c' ? 0 : 1) & (e1'=e1-1); // transmit
endmodule

module user2 = user1 [ s1=s2, e1=e2, w1=w2, t1=t2 ] endmodule

module channel
  c : bool init false; // is there a collision?
  [t1,w2] true -> q1 : (c'=false) + (1-q1) : (c'=true); // only user 1 transmits
  [w1,t2] true -> q1 : (c'=false) + (1-q1) : (c'=true); // only user 2 transmits
  [t1,t2] true -> q2 : (c'=false) + (1-q2) : (c'=true); // both users transmit
endmodule
```
Variable updates can refer to other variables updated simultaneously.

Action lists used to specify synchronisation.

---

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
Example: Future markets investor

• Model of interactions between:
  – stock market, evolves stochastically
  – two investors $i_1$, $i_2$ decide when to invest
  – market decides whether to bar investors

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

• Analysed with rPATL model checking & strategy synthesis
  – distinct profit models considered: ‘normal market’, ‘later cash–ins’ and ‘later cash–ins with fluctuation’
  – comparison between TSG and CSG models
Example: Future markets investor

- **Example rPATL query:**
  - $\langle\langle \text{investor}_1, \text{investor}_2 \rangle\rangle \ R^{\text{profit}_{1,2}}_{\text{max}} = \text{?} \ [ F \text{ finished}_{1,2} ]$
  - i.e. maximising joint profit

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

![Graphs showing comparison between CSG and TSG strategies over months.](image-url)
Equilibria–based properties
Equilibria-based properties

- **Motivation:**
  - 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 $\epsilon$-NE, which always exist for CSGs
  - a strategy profile $\sigma=(\sigma_1, \ldots, \sigma_n)$ for a CSG is an $\epsilon$-NE for state $s$ and objectives $X_1, \ldots, X_n$ iff:
    - $\Pr_s \sigma(X_i) \geq \sup \{ \Pr_s \sigma'(X_i) \mid \sigma' = \sigma_{-i}[\sigma_i'] \text{ and } \sigma_i' \in \Sigma_i \} - \epsilon$ for all $i$
Social-welfare Nash equilibria

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

- We extend rPATL accordingly

  Zero-sum properties

  $\langle\langle\text{robot}_1\rangle\rangle_{\max=\text{?}} P[F \leq^k \text{goal}_1]$  
  find a robot 1 strategy which maximises the probability of it reaching its goal, regardless of robot 2

  Equilibria-based properties

  $\langle\langle\text{robot}_1:\text{robot}_2\rangle\rangle_{\max=\text{?}} (P[F \leq^k \text{goal}_1] + P[F \leq^k \text{goal}_2])$
  find (SWNE) strategies for robots 1 and 2 where there is no incentive to change actions and which maximise joint goal probability
Model checking for extended rPATL

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

- We further extend the value iteration approach:

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

- where \(Z_1\) and \(Z_2\) encode matrix games similar to before
PRISM–games support

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

- **Applications & results**
  - robot navigation in a grid, medium access control, Aloha communication protocol, power control
  - SWNE strategies outperform those found with rPATL
  - ε–Nash equilibria found typically have ε=0
Example: multi-robot coordination

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

- We synthesise SWNEs to maximise the average probability of robots reaching their goals within time \( k \)
  - \( \langle \langle \text{robot1:robot2} \rangle \rangle_{\text{max}} =? (P [ F^{\leq k} \text{ goal}_1 ] + P [ F^{\leq k} \text{ goal}_2 ]) \)

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

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

- **Managing model uncertainty**
  - integration with learning
  - robust verification

- **Accuracy of model checking results**
  - value iteration improvements; exact methods

- **Scalability & efficiency**
  - e.g. symbolic methods, abstraction, symmetry reduction
  - sampling–based strategy synthesis methods
• See the PRISM–games website for more info
  – prismmodelchecker.org/games/
  – documentation, examples, case studies, papers
  – downloads: 🍎 ♂️ ☀️ + CAV’20 artefact VM
  – open source (GPLV2): ⚙️