Automated Game-theoretic Verification for Probabilistic Systems

Dave Parker
University of Birmingham

Joint work with:
Taolue Chen, Vojtěch Forejt, Marta Kwiatkowska, Aistis Simaitis

Imperial College London, December 2012
Verifying stochastic systems

- **Quantitative verification**
  - probability, time, costs/rewards, ...
  - in particular: systems with stochastic behaviour
  - e.g. due to unreliability, uncertainty, randomisation, ...
  - often: subtle interplay between probability/nondeterminism

- **Automated verification**
  - probabilistic model checking
  - efficiency and scalable algorithms/techniques
  - tool support: PRISM model checker

- **Practical applications**
  - wireless communication protocols, security protocols,
    systems biology, DNA computing, robotic planning, ...
Competitive/collaborative behaviour

• **Open systems**
  – need to account for the behaviour of system components not under our control, possibly with differing/opposing goals
  – giving rise to competitive/collaborative behaviour

• **Many occurrences in practice**
  – e.g. security protocols, algorithms for distributed consensus, energy management or sensor network co-ordination

• **Natural to adopt a game-theoretic view**
  – widely used in computer science, economics, ...

• **This talk**
  – verifying systems with stochastic and game-theoretic aspects
  – stochastic multi-player games
  – temporal logic, model checking, tool support, case studies
Overview

• Probabilistic model checking

• Stochastic multi-player games (SMGs)
  – strategies, probabilities, rewards

• Property specification: rPATL
  – syntax, semantics, subtleties

• rPATL model checking
  – algorithms, tool support

• Case study: Energy management in microgrids
Probabilistic model checking

Probabilistic model
- e.g. Markov chain, Markov decision process

System
- trigger → $P > 0.999$ [ $F \leq 20$ deploy ]

System requirements
- Probabilistic temporal logic specification
  - e.g. PCTL, CSL, LTL, ...

Result
- Quantitative results
- Counter-example

Probabilistic model checker
- e.g. PRISM
• Usually focus on **quantitative** (numerical) properties:
  – \( P_E [ F \leq 20 \text{ deploy} ] \) – “what is the probability of the airbag deploying within 20ms?”

• Then analyse **trends** in quantitative properties as system parameters vary
  – looking for flaws, anomalies, ...

• Unlike (non–probabilistic) model checking
  – often investigate effect of (known) failures, rather than identifying existence of (unknown) bugs

• **Strength:** combines **numerical** and **exhaustive** aspects
  – “worst–case (maximum) probability of the airbag failing to deploy within 20ms, *from any possible* crash scenario”
  – “worst–case (maximum) expected algorithm execution time *for any possible scheduling* of system components”
Stochastic multi-player games

• **Stochastic multi-player game (SMGs)**
  – nondeterminism + multiple players + probability

• **A (turn-based) SMG is a tuple** \((\Pi, S, \langle S_i \rangle_{i \in \Pi}, A, \Delta, L)\):
  – \(\Pi\) is a set of \(n\) players
  – \(S\) is a (finite) set of states
  – \(\langle S_i \rangle_{i \in \Pi}\) is a partition of \(S\)
  – \(A\) is a set of action labels
  – \(\Delta : S \times A \to \text{Dist}(S)\) is a (partial) transition probability function
  – \(L : S \to 2^{\text{AP}}\) is a labelling with atomic propositions from \(\text{AP}\)
Strategies, probabilities & rewards

• **Strategy for player $i$:** resolves choices in $S_i$ states
  – based on execution history, i.e. $\sigma_i : (SA)^*S_i \rightarrow \text{Dist}(A)$
  – can be: deterministic (pure), randomised, memoryless, finite-memory, …
  – $\Sigma_i$ denotes the set of all strategies for player $i$

• **Strategy profile:** strategies for all players: $\sigma=(\sigma_1,\ldots,\sigma_n)$
  – induces a set of (infinite) paths from some start state $s$
  – a probability measure $\Pr_s^\sigma$ over these paths

• **Rewards (or costs)**
  – non-negative integers on states/transitions
  – e.g. elapsed time, energy consumption, number of packets lost, net profit, …
  – this talk: expected cumulated value of rewards
Property specification: rPATL

- **New temporal logic rPATL:**
  - reward probabilistic alternating temporal logic

- **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:**
  - $\langle\langle \{1,3\} \rangle\rangle P_{<0.01} [ F_{\leq 10} \text{ error } ]$
  - “players 1 and 3 have a strategy to ensure that the probability of an error occurring within 10 steps is less than 0.01, regardless of the strategies of other players”
rPATL syntax/semantics

- **Syntax:**
  \[ \phi ::= \top | a | \neg \phi | \phi \land \phi | \langle\langle C\rangle\rangle P_{\bowtie q}[\psi] | \langle\langle C\rangle\rangle R_{r \bowtie x} [F^* \phi] \]
  \[ \psi ::= X \phi | \phi U \phi | F \phi | G \phi | \phi U^{\leq k} \phi | F^{\leq k} \phi | G^{\leq k} \phi \]
  - where:
    - \(a \in \text{AP}\) is an atomic proposition, \(C \subseteq \Pi\) is a coalition of players,
    \(\bowtie \in \{\leq,<,>,\geq\}\), \(q \in [0,1] \cap \mathbb{Q}\), \(x \in \mathbb{Q}_{\geq 0}\), \(k \in \mathbb{N}\)
    - \(r\) is a reward structure and \(* \in \{0,\infty,c\}\) is a reward type

- **Semantics:**

- **P operator:** \(s \models \langle\langle C\rangle\rangle P_{\bowtie q}[\psi]\) iff:
  - “there exist strategies for players in coalition \(C\) such that, for all strategies of the other players, the probability of path formula \(\psi\) being true from state \(s\) satisfies \(\bowtie q\)”
Examples

\[ P_{\geq \frac{1}{4}}[ F \checkmark ] \]

true in initial state

\[ P_{\geq \frac{1}{3}}[ F \checkmark ] \]

\[ P_{\geq \frac{1}{3}}[ F \checkmark ] \]
Examples

\[ \langle\langle \text{true} \rangle\rangle P_{\geq \frac{1}{4}}[ F \checkmark ] \]
true in initial state

\[ \langle\langle \text{false} \rangle\rangle P_{\geq \frac{1}{3}}[ F \checkmark ] \]
false in initial state

\[ \langle\langle \text{true, false} \rangle\rangle P_{\geq \frac{1}{2}}[ F \checkmark ] \]
Examples

\[ \langle \langle \circ \rangle \rangle P_{\geq \frac{1}{4}}[ F \checkmark ] \]
true in initial state

\[ \langle \langle \circ \rangle \rangle P_{\geq \frac{1}{3}}[ F \checkmark ] \]
false in initial state

\[ \langle \langle \circ, \square \rangle \rangle P_{\geq \frac{1}{3}}[ F \checkmark ] \]
true in initial state
rPATL semantics (rewards)

- **R operator**: \( s \models \langle\langle C\rangle\rangle R^r_\infty [F^\ast \phi] \) iff:
  - "there exist strategies for players in coalition C such that, for all strategies of the other players, the expected cumulated reward \( r \) to reach a \( \phi \)–state (type \( \ast \)) satisfies \( \bowtie x \)"

- **3 reward types** \( \ast \in \{\infty, c, 0\} \)
  - defining reward if a \( \phi \)–state is never reached
  - reward is: infinite (\( \ast = \infty \)), cumulated sum (\( \ast = c \)), zero (\( \ast = 0 \))
  - \( \infty \): e.g. expected time for algorithm execution
  - \( c \): e.g. expected resource usage (energy, messages sent, ...)
  - \( 0 \): e.g. reward incentive awarded on algorithm completion

- **Note**: \( F^0 \) operator needs finite–memory strategies
  - (for P and other R operators, pure memoryless strat.s suffice)
rPATL extensions

- **Quantitative (numerical) properties:**
  - numerical rather than boolean-valued queries

- **Example:**
  - \( \langle \langle \{1\} \rangle \rangle \ P_{\text{max}=?} [ F \text{ error} ] \)
  - “what is the maximum probability of reaching an error state that player 1 can guarantee?” (against player 2)
  - i.e. \( \sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} \Pr_{s, \sigma_1, \sigma_2} (F \text{ error}) \)

- **Other extensions:**
  - rPATL* (i.e. support for LTL formulae in P operator)
  - reward-bounded operators
  - exact probability/reward bounds
Model checking rPATL

• **Main task: checking individual P and R operators**
  – reduction to solution of zero-sum stochastic 2-player game
  – (probabilistic reachability + expected total reward)
  – e.g. $\langle\langle C \rangle\rangle P_{\geq q}[\psi] \iff \sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} \Pr_{s, \sigma_1, \sigma_2} (\psi) \geq q$
  – complexity: $NP \cap coNP$ (without any $R[F^0]$ operators)
  – complexity for full logic: $NEXP \cap coNEXP$ (due to $R[F^0]$ op.)

• **In practice though:**
  – (usual approach taken in probabilistic model checking tools)
  – evaluation of numerical **fixed points** (“value iteration”)
  – and more: graph-algorithms, sequences of fixed points, …

• **See:** [TACAS’12], [CONCUR’12]
Independence of strategies

- Strategies for each coalition operator are independent
  - for example, in: \langle\langle 1 \rangle\rangle P_{\geq 1} [ G ( \langle\langle 1,2 \rangle\rangle P_{\geq \frac{1}{4}} [ F \checkmark ] ) ]
  - no dependencies in player 1 strategies in quantifiers
  - branching–time temporal logic (like ATL, PCTL, ...)

- Introducing dependencies is problematic
  - e.g. subsumes existential semantics for PCTL on Markov decision processes (MDPs), which is undecidable
  - (does there exist a single adversary satisfying one formula?)
  - \langle\langle 1 \rangle\rangle P_{\geq 1} [ G \langle\langle 1 \rangle\rangle P_{\geq \frac{1}{4}} [ F \checkmark ] ]

- But nested properties still have natural applications
  - e.g. sensor network, with players: sensor, repairer
  - \langle\langle sensor \rangle\rangle P_{<0.01} [ F (\neg \langle\langle repairer \rangle\rangle P_{\geq 0.95} [ F \text{ “operational” } ] ) ]
Why do we need multiple players?

- SMGs have multiple (>2) players
  - but model checking (and semantics) reduce to 2-player case
  - due to (zero sum) nature of queries expressible by rPATL
  - so why do we need multiple players?

1. Modelling convenience
   - and/or multiple rPATL queries on same model

2. May also exploit in nested queries, e.g.:
   - players: sensor1, sensor2, repairer
   - $\langle\langle\text{sensor1}\rangle\rangle P_{<0.01}[ F (\neg\langle\langle\text{repairer}\rangle\rangle P_{\geq0.95}[ F \text{ “operational” } ] ) ]$
Probabilities for P operator

• E.g. $\langle\langle C \rangle\rangle P_{\geq q}[ F \phi ]$ : max/min reachability probabilities
  - compute $\sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} \Pr_{s_1, s_2} (F \phi)$ for all states $s$
  - deterministic memoryless strategies suffice

• Value is:
  - 1 if $s \in \text{Sat}(\phi)$, and otherwise least fixed point of:

$$f(s) = \begin{cases} 
\max_{a \in A(s)} \left( \sum_{s' \in S} \Delta(s, a)(s') \cdot f(s') \right) & \text{if } s \in S_1 \\
\min_{a \in A(s)} \left( \sum_{s' \in S} \Delta(s, a)(s') \cdot f(s') \right) & \text{if } s \in S_2 
\end{cases}$$

• Computation:
  - start from zero, propagate probabilities backwards
  - guaranteed to converge
Example

\[
\text{Compute: } \sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} \Pr_{\sigma_1, \sigma_2}(F \checkmark)
\]

rPATL: \(\langle\langle \bigcirc, \square \rangle\rangle P_{\geq \frac{1}{3}} [ F \checkmark ]\)

Player 1: \(\bigcirc, \square\)  Player 2: \(\diamondsuit\)
Rewards for $R[F^c]$ operator

- E.g. $\langle\langle C \rangle\rangle R^r_{\geq q}[ F^c \phi ]$: max/min expected rewards for P1/P2
  - again: deterministic memoryless strategies suffice

- Value is:
  - $\infty$ if $s \in \text{Sat}(\langle\langle C \rangle\rangle P > 0[ G F \, "pos\_rew" \] )
  - 0 if $s \in \text{Sat}(\phi)$, and otherwise least fixed point of:

\[
f(s) = \begin{cases} 
  r(s) + \max_{a \in A(s)} \left( \sum_{s' \in S} \Delta(s, a)(s') \cdot f(s') \right) & \text{if } s \in S_1 \\
  r(s) + \min_{a \in A(s)} \left( \sum_{s' \in S} \Delta(s, a)(s') \cdot f(s') \right) & \text{if } s \in S_2
\end{cases}
\]
Rewards for $R[F^\infty]$ operator

• E.g. $\langle\langle C \rangle\rangle R^\infty_{\geq q}[ F^\infty \phi ]$: max/min expected rewards for $P_1 / P_2$
  – again: deterministic memoryless strategies suffice

• Value is:
  – $\infty$ if $s \in \text{Sat}(\langle\langle C \rangle\rangle P > 0[ G F \text{“pos rew”} ] )$,
  – 0 if $s \in \text{Sat}(\phi)$, and otherwise greatest fixed point over $\mathbb{R}$ of:

\[
f(s) = \begin{cases} 
  r(s) + \max_{a \in A(s)} \left( \sum_{s' \in S} \Delta(s, a)(s') \cdot f(s') \right) & \text{if } s \in S_1 \\
  r(s) + \min_{a \in A(s)} \left( \sum_{s' \in S} \Delta(s, a)(s') \cdot f(s') \right) & \text{if } s \in S_2
\end{cases}
\]

• Computation:
  – 1. set zero rewards to $\epsilon$, compute least fixed point
  – 2. evaluate greatest fixed point, downwards from step 1
Example: Finite memory for R[F0]

- E.g. $\langle\langle C \rangle\rangle R^r_{\geq q}[ F^0 \phi ]$ : max/min expected rewards for P1/P2
  - now: deterministic memoryless strategies do not suffice

What if incoming reward is 2?

$b$: reward 2
$a, b$: expected reward 1.5
Rewards for $R[F^0]$ operator

- E.g. $\langle \langle C \rangle \rangle R^r_{\geq q} [ F^0 \phi ]$: max/min expected rewards for P1/P2
  - now: deterministic memoryless strategies do not suffice

- There exists a finite-memory optimal strategy for P1
  - there exists a bound $B$, beyond which strategy is memoryless
  - $B$ is exponential in worst-case, but can be computed...

- Computation:
  - compute bound $B$ (using simpler rPATL queries)
  - perform value iteration for each level 0,...,$B$; combine results
Tool support: PRISM-games

- **Model checker for stochastic multi-player games**
  - PRISM-games: extension of PRISM model checker
  - using new explicit-state model checking engine
  - symbolic (BDD-based) implementation in progress

- **Features:**
  - modelling language for SMGs (guarded command based)
  - rPATL model checking
  - strategy synthesis and analysis
  - GUI: model editor, simulator, graph-plotting, strategies, ...

- **Available now**
Case studies

• Evaluated on several case studies:
  – team formation protocol [CLIMA’11]
  – futures market investor model [McIver & Morgan]
  – collective decision making for sensor networks [TACAS’12]
  – energy management in microgrids [TACAS’12]

• Ongoing applications
  – trust models in user-centric networks
  – (randomised) security protocols
Energy management in microgrids

- Microgrid: proposed model for future energy markets
  - localised energy management

- Neighbourhoods use and store electricity generated from local sources
  - wind, solar, ...

- Needs: demand-side management
  - active management of demand by users
  - to avoid peaks
Microgrid demand-side management

- **Demand-side management algorithm** [Hildmann/Saffre’11]
  - N households, connected to a distribution manager
  - households submit loads for execution
  - execution cost/step = number of currently running loads

- **Simple algorithm:**
  - upon load generation, if cost is below an agreed limit $c_{lim}$, execute it, otherwise only execute with probability $P_{start}$

- **Analysis of [Hildmann/Saffre’11]**
  - load submission probability: daily demand curve
  - load duration: random, between 1 and D steps
  - define household value as $V=\text{loads}_\text{executing}/\text{execution}_\text{cost}$
  - simulation-based analysis shows reduction in peak demand and total energy cost reduced, with good expected value $V$
  - (if all households stick to algorithm)
Microgrid demand-side management

• The model
  – SMG with N players (one per household)
  – analyse 3-day period, using piecewise approximation of daily demand curve
  – fix parameters $D=4$, $c_{\text{lim}}=1.5$
  – add rewards structure for value $V$

• Built/analysed models
  – for $N=2,…,7$ households

• Step 1: assume all households follow algorithm of [HS’11] (MDP)
  – obtain optimal value for $P_{\text{start}}$

• Step 2: introduce competitive behaviour (SMG)
  – allow coalition $C$ of households to deviate from algorithm

<table>
<thead>
<tr>
<th>$N$</th>
<th>States</th>
<th>Transitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>743,904</td>
<td>2,145,120</td>
</tr>
<tr>
<td>6</td>
<td>2,384,369</td>
<td>7,260,756</td>
</tr>
<tr>
<td>7</td>
<td>6,241,312</td>
<td>19,678,246</td>
</tr>
</tbody>
</table>
Results: Competitive behaviour

- Expected total value $V$ per household
  - in rPATL: $\langle \langle C \rangle \rangle R^{r_{C_{\max}}} = ? [F^0 \text{ time}=\text{max time}] / |C|$
  - where $r_C$ is combined rewards for coalition $C$

![Graph showing the relationship between reward per household and number of households.]

- All follow alg.
- No use of alg.
- Deviations of varying size

Strong incentive to deviate
Results: Competitive behaviour

- **Algorithm fix: simple punishment mechanism**
  - distribution manager can cancel some loads exceeding $c_{lim}$

![Graph](chart.png)

- Better to collaborate (with all)
- All follow alg.
- Deviations of varying size
Conclusions

- game-theoretic verification for probabilistic systems
- modelled as stochastic multi-player games
- new temporal logic rPATL for property specification
- rPATL model checking algorithm based on num. fixed points
- model checker PRISM-games
- case studies: e.g. energy management for microgrid

Future work

- more realistic classes of strategy, e.g. partial observation, ...
- further objectives, e.g. multiple objectives, Nash equilibria, ...
- more application areas: security, randomised algorithms, ...

PRISM-games: http://www.prismmodelchecker.org/games/