Automatic Verification of Competitive Stochastic Systems

Dave Parker

University of Oxford

Joint work with:

Taolue Chen, Vojtěch Forejt, Marta Kwiatkowska, Aistis Simaitis

University of Edinburgh, March 2012

Verifying stochastic systems

- Quantitative verification
  - of systems with stochastic behaviour
  - e.g. due to unreliability, uncertainty, randomisation, …
  - probability, costs/rewards, time, …
  - often: subtle interplay between probability/nondeterminism
- Automated verification
  - probabilistic model checking
  - tool support: PRISM model checker
  - techniques for improving efficiency, scalability
- Practical applications
  - wireless communication protocols, security protocols, systems biology, DNA computing, robotic planning, …

Adding competitive behaviour

- Open systems
  - need to account for the behaviour of components not under our control, possibly with differing/opposing goals
  - giving rise to competitive 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, …
  - also used in model checking, e.g. security/comm. protocols
- This talk
  - systems with competitive and stochastic behaviour
  - stochastic multi-player games
  - temporal logic, model checking, tool support, case studies

Overview

- Probabilistic model checking
  - probabilistic models, property specifications
- Stochastic multi-player games (SMGs)
  - the model, probability spaces, rewards
- Property specification: rPATL
  - syntax, semantics, subtleties
- rPATL model checking
  - algorithm, numerical computation, tool support
- Case study: energy management in microgrids

Probabilistic models

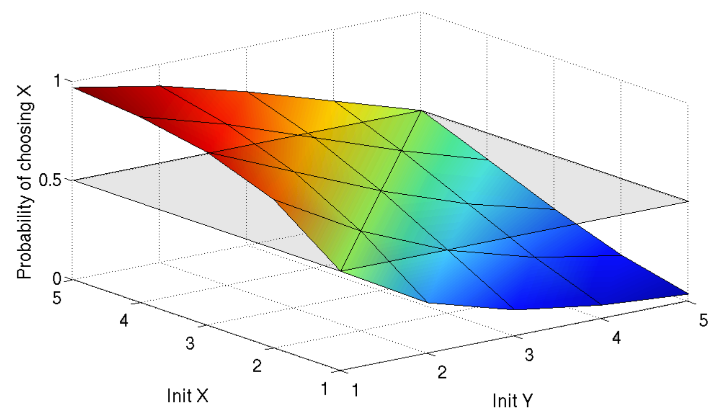
- Discrete-time Markov chains (DTMCs)
  - discrete states + probability
  - for: randomisation, unreliable communication media, …
- Continuous-time Markov chains (CTMCs)
  - discrete states + exponentially distributed delays
  - for: component failures, job arrivals, molecular reactions, …
- Markov decision processes (MDPs)
  - probability + nondeterminism (e.g. for concurrency)
  - for: randomised distributed algorithms, security protocols, …
- Probabilistic timed automata (PTAs)
  - probability, nondeterminism + real-time
  - for wireless comm. protocols, embedded control systems, …

Probabilistic model checking

- Property specifications based on temporal logic
  - PCTL, CSL, probabilistic LTL, PCTL*, …

- Simple examples:
  - $P_{\leq 0.01}$ [F "crash"] – "the probability of a crash is at most 0.01"
  - $S_{>0.999}$ ["up"] – "long-run probability of availability is >0.999"

- Usually focus on quantitative (numerical) properties:
  - $P_{=?}$ [F "crash"]
    "what is the probability
    of a crash occurring?"
  - then analyse trends in
    quantitative properties
    as system parameters vary



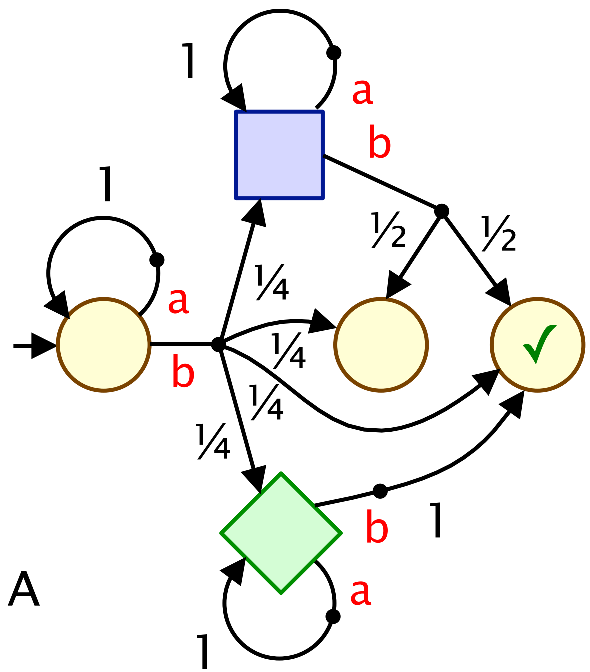
Probabilistic model checking

- Typically combine numerical + exhaustive aspects
  - $P_{max=?}$ [$F^{\leq 10}$ "fail"] – "worst-case probability of a failure occurring within 10 seconds, for any possible scheduling of system components"
  - $P_{=?}$ [$G^{\leq 0.02}$!"deploy" {"crash"}{max}] – "the maximum probability of an airbag failing to deploy within 0.02s, from any possible crash scenario"
  - model checking: graph analysis + numerical solution + …

- Reward-based properties (rewards = costs = prices)
  - $R_{\{\text{"time"}\}=?}$ [F "end"] – "expected algorithm execution time"
  - $R_{\{\text{"energy"}\}max=?}$ [$C^{\leq 7200}$] – "worst-case expected energy consumption during the first 2 hours"

Stochastic multi-player games

- Stochastic multi-player game (SMGs)
  - probability + nondeterminism + multiple players
- A (turn-based) SMG is a tuple $(\Pi, S, \langle S_i \rangle_{i \in \Pi}, A, \Delta, L)$:
  - Π 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
  - $\Delta : S \times A \rightarrow Dist(S)$ is a (partial) transition probability function
  - $L : S \rightarrow 2^{AP}$ is a labelling with atomic propositions from AP
- Notation:
  - $A(s)$ denotes available actions in state A



Paths, strategies + probabilities

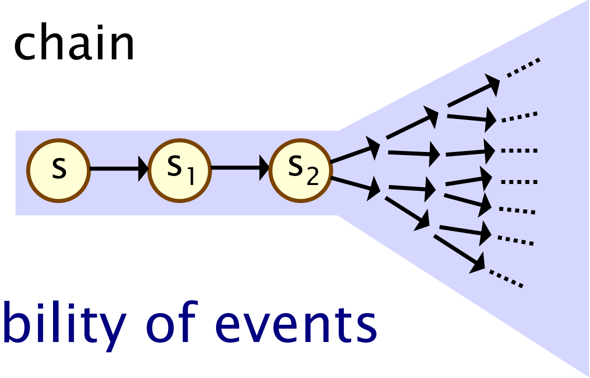
- A path is an (infinite) sequence of connected states in SMG
  - i.e. $s_0 a_0 s_1 a_1 \ldots$ such that $a_i \in A(s_i)$ and $\Delta(s_i,a_i)(s_{i+1})>0$ for all i
  - represents a system execution (i.e. one possible behaviour)
  - to reason formally, need a probability space over paths

- A strategy for player $i \in \Pi$ resolves choices in S_i states
  - based on history of execution so far
  - i.e. a function $\sigma_i : (SA)^* S_i \rightarrow Dist(A)$
  - Σ_i denotes the set of all strategies for player i

- A strategy profile is tuple $\sigma=(\sigma_1,\ldots,\sigma_n)$
  - combining strategies for all n players
  - deterministic if σ always gives a Dirac distribution
  - memoryless if $\sigma(s_0 a_0 \ldots s_k)$ depends only on s_k

Paths, strategies + probabilities…

- For a strategy profile σ:
  - the game's behaviour is fully probabilistic
  - essentially an (infinite-state) Markov chain
  - yields a probability measure Pr_s^σ over set of all paths $Path_s$ from s



- Allows us to reason about the probability of events
  - under a specific strategy profile σ
  - e.g. any (ω-)regular property over states/actions

- Also allows us to define expectation of random variables
  - i.e. measurable functions $X : Path_s \rightarrow \mathbb{R}_{\geq 0}$
  - $E_s^\sigma[X] = \int_{Path_s} X \, dPr_s^\sigma$
  - used to define expected costs/rewards…

Rewards

- Rewards (or costs)
  - real-valued quantities assigned to states (and/or transitions)
- Wide range of possible uses:
  - elapsed time, power consumption, size of message queue, number of messages successfully delivered, net profit, …
- We use:
  - state rewards: $r : S \rightarrow \mathbb{N}$ (but can generalise to $\mathbb{Q}_{\geq 0}$)
  - expected cumulative reward until a target set T is reached
- 3 interpretations of rewards
  - 3 reward types $\star \in \{\infty, c, 0\}$, differing where T is not reached
  - reward is assumed to be infinite, cumulated sum, zero, resp.
  - ∞: 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

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 version of reward operator R from PRISM
- Example:
  - $\langle\langle \{1,2\} \rangle\rangle \; P_{<0.01} \; [\; F^{\leq 10} \; error \;]$
  - “players 1 and 2 have a strategy to ensure that the probability of an error occurring within 10 steps is less than 0.1, regardless of the strategies of other players”

rPATL syntax

- Syntax:

$$\phi ::= \top \mid a \mid \neg\phi \mid \phi \wedge \phi \mid \langle\langle C\rangle\rangle P_{\bowtie q}[\psi] \mid \langle\langle C\rangle\rangle R^{r}_{\bowtie x}\ [F^{\star}\phi]$$

$$\psi ::= X\ \phi \mid \phi\ U^{\leq k}\ \phi \mid F^{\leq k}\ \phi \mid G^{\leq k}\ \phi$$

- where:
  - $a \in AP$ is an atomic proposition, $C \subseteq \Pi$ is a coalition of players, $\bowtie \in \{\leq,<,>,\geq\}$, $q \in [0,1] \cap \mathbb{Q}$, $x \in \mathbb{Q}_{\geq 0}$, $k \in \mathbb{N} \cup \{\infty\}$
    r is a reward structure and $\star \in \{0,\infty,c\}$ is a reward type
- $\langle\langle C\rangle\rangle P_{\bowtie q}[\psi]$
  - "players in coalition C have a strategy to ensure that the probability of path formula ψ being true satisfies $\bowtie$ q, regardless of the strategies of other players"
- $\langle\langle C\rangle\rangle R^{r}_{\bowtie x}\ [F^{\star}\phi]$
  - "players in coalition C have a strategy to ensure that the expected reward r to reach a ϕ-state (type $\star$) satisfies $\bowtie$ x, regardless of the strategies of other players"

rPATL semantics

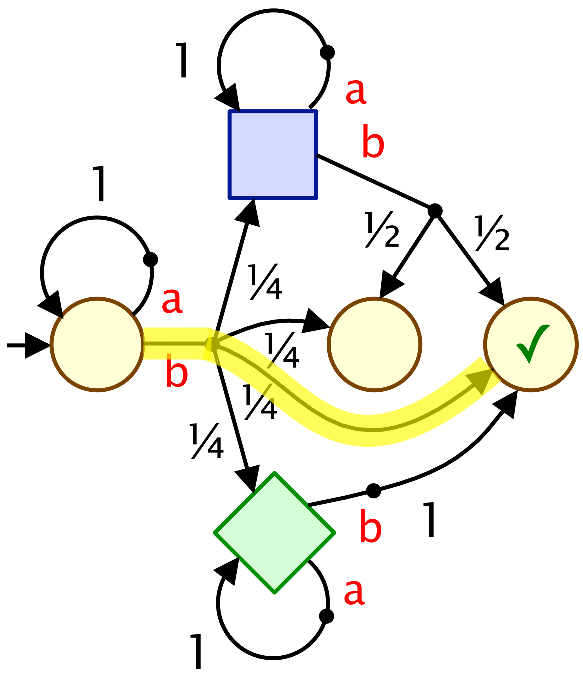
- Semantics for most operators is standard
- Just focus on P and R operators…
  - present using reduction to a stochastic 2-player game
  - (as for later model checking algorithms)

- Coalition game G_C for SMG G and coalition $C \subseteq \Pi$
  - 2-player SMG where C and $\Pi \backslash C$ collapse to players 1 and 2

- $\langle\langle C \rangle\rangle P_{\bowtie q}[\psi]$ is true in state s of G iff:
  - in coalition game G_C:
  - $\exists \sigma_1 \in \Sigma_1$ such that $\forall \sigma_2 \in \Sigma_2$. $Pr_s^{\sigma_1,\sigma_2}(\psi) \bowtie q$

- Semantics for R operator defined similarly…

Examples



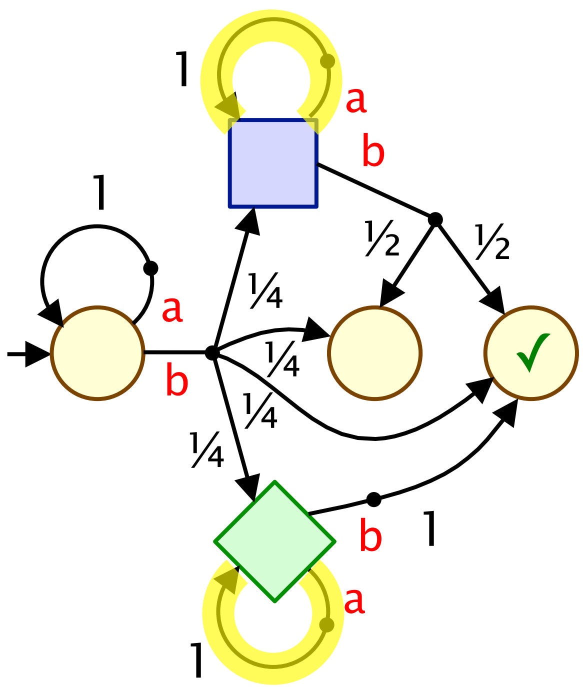
⟨⟨○⟩⟩$P_{\geq 1/4}$[F ✓]

true in initial state

⟨⟨○⟩⟩$P_{\geq 1/3}$ [F ✓]

⟨⟨○,□⟩⟩$P_{\geq 1/3}$ [F ✓]

Examples



$\langle\langle \bigcirc \rangle\rangle P_{\geq 1/4} [\ F\ ✓\ ]$

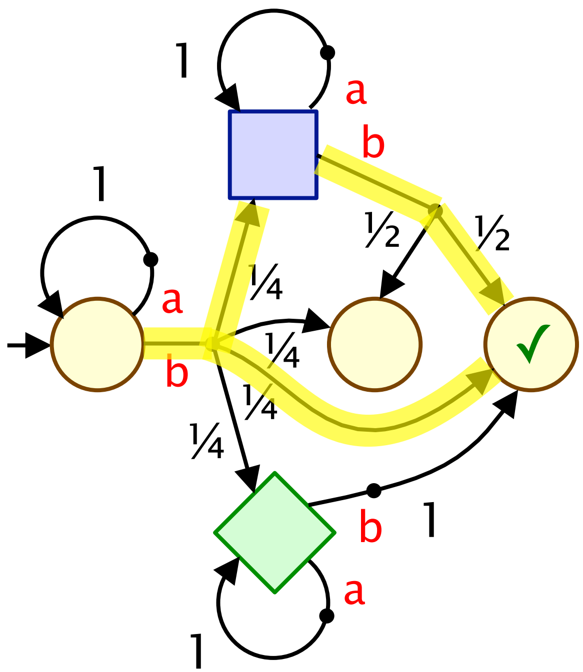
true in initial state

$\langle\langle \bigcirc \rangle\rangle P_{\geq 1/3} [\ F\ ✓\ ]$

false in initial state

$\langle\langle \bigcirc, \square \rangle\rangle P_{\geq 1/3} [\ F\ ✓\ ]$

Examples



⟨⟨○⟩⟩$P_{\geq 1/4}$[F ✓]

true in initial state

⟨⟨○⟩⟩$P_{\geq 1/3}$ [F ✓]

false in initial state

⟨⟨○,□⟩⟩$P_{\geq 1/3}$ [F ✓]

true in initial state

Equivalences + extensions

- Two useful equivalences:

- $\langle\langle C\rangle\rangle P_{\geq q}[\neg\psi] \equiv \langle\langle C\rangle\rangle P_{\leq 1-q}[\psi]$
  - easy to derive path properties e.g. $G\ a \equiv \neg F \neg a$
  - model checking essentially just focuses on reachability

- $\langle\langle C\rangle\rangle P_{\geq q}[\psi] \equiv \neg\langle\langle \Pi \setminus C\rangle\rangle P_{<q}[\psi]$
  - thanks to standard determinacy results
  - model checking focuses on min/max values for P1/P2

- Quantitative (numerical) properties:
  - best/worst-case values
- e.g. $\langle\langle C\rangle\rangle P_{max=?}[\psi] = \sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} Pr_s^{\sigma_1,\sigma_2}(\psi)$

Independence of strategies

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

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

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

Why do we need multiple players?

- SMGs have multiple (>2) players
  - but semantics (and model checking) 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
  - $\langle\langle sensor1 \rangle\rangle \, P_{<0.01}[\ F\ (\neg \langle\langle repairer \rangle\rangle \, P_{\geq 0.95}[\ F\ \text{“operational”}\ ]\ )\ ]$

Model checking rPATL

- Basic algorithm: as for any branching-time temporal logic
  - recursive descent of formula parse tree
  - compute $Sat(\phi) = \{ s \in S \mid s \models \phi \}$ for each subformula ϕ
- Main task: checking P and R operators
  - reduction to solution of stochastic 2-player game G_C
  - e.g. $\langle\langle C \rangle\rangle P_{\geq q}[\psi] \;\Leftrightarrow\; \sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} Pr_s^{\sigma_1,\sigma_2}(\psi) \geq q$
  - complexity: NP $\cap$ coNP (without any $R[F^0]$ operators)
  - compared to, e.g. P for Markov decision processes
  - complexity for full logic: NEXP $\cap$ coNEXP (due to $R[F^0]$ op.)
- In practice though:
  - evaluation of numerical fixed points ("value iteration")
  - up to a desired level of convergence
  - usual approach taken in probabilistic model checking tools

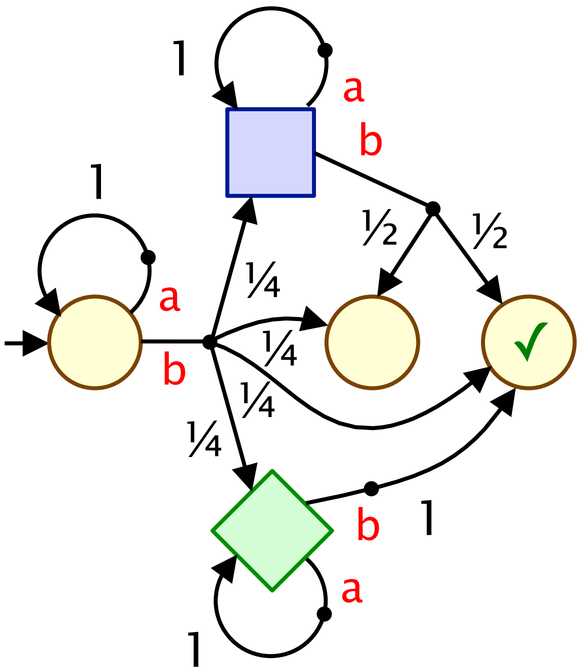
Probabilities for P operator

- E.g. $\langle\langle C \rangle\rangle P_{\geq q}[\ F\ \phi\ ]$: max/min reachability probabilities
  - compute $\sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} Pr_s^{\sigma_1,\sigma_2}(F\ \phi)$ for all states s
  - deterministic memoryless strategies suffice
- Value is:
  - 1 if $s \in Sat(\phi)$, 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



rPATL: $\langle\langle$ ○, □ $\rangle\rangle P_{\geq 1/3}$ [F ✓]

Player 1: ○, □ Player 2: ◇

Compute: $\sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} Pr_s^{\sigma_1, \sigma_2}$ (F ✓)

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:
  - ∞ if $s \in Sat(\ \langle\langle C \rangle\rangle P_{>0}[\ G\ F\ \text{"pos\_rew"}\ ]\ )$,
  - 0 if $s \in Sat(\phi)$, 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. $\langle\langle C \rangle\rangle R^r_{\geq q}[\ F^\infty\ \phi\ ]$: max/min expected rewards for P1/P2
  - again: deterministic memoryless strategies suffice

- Value is:
  - ∞ if $s \in \mathrm{Sat}(\ \langle\langle C \rangle\rangle P_{>0}[\ G\ F\ \text{“pos\_rew”}\ ]\ )$,
  - 0 if $s \in \mathrm{Sat}(\phi)$, and otherwise greatest fixed point over $\mathbb{R}$ of:

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

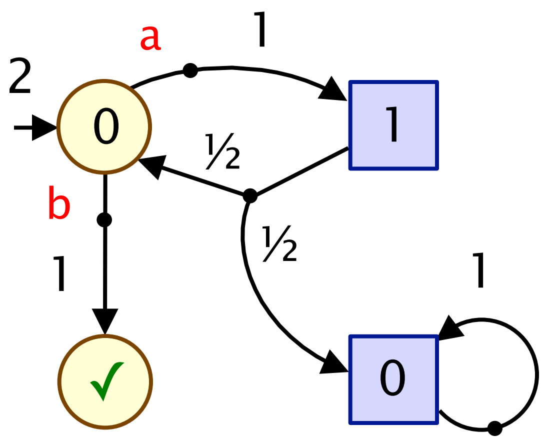
- Computation:
  - 1. set zero rewards to ϵ, compute least fixed point
  - 2. evaluate greatest fixed point, downwards from step 1

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

Example: Finite memory for $R[F^0]$



⟨⟨○,□⟩⟩$R^r_{\geq 1/2}$ [F^0 ✓]

b: reward 0
a, b: expected reward 0.5
a, a, b: expected reward 0.5
a, a, a, b: expected reward 0.375

What if incoming reward is 2?

b: reward 2
a, b: expected reward 1.5

Tool support: PRISM-games

- Prototype model checker for stochastic games
  - integrated into PRISM model checker
  - using new explicit-state model checking engine

- SMGs added to PRISM modelling language
  - guarded command language, based on Reactive modules
  - finite data types, parallel composition, proc. algebra op.s, …

- rPATL added to PRISM property specification language
  - implemented value iteration based model checking

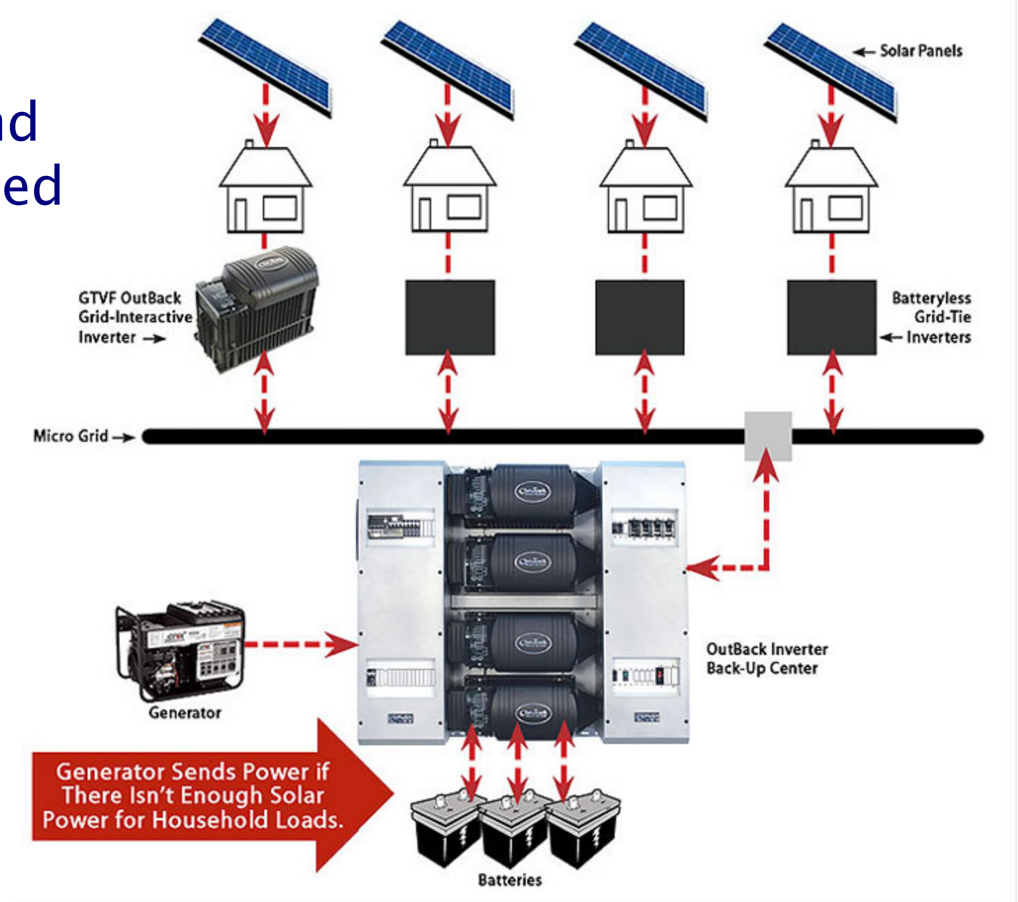
- Available now:
  - http://www.prismmodelchecker.org/games/

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]

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

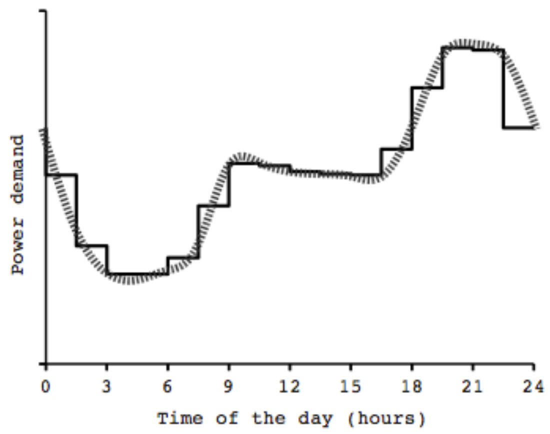


Microgrid demand-side management

- Demand-side management algorithm [Hildmann/Saffre'11]
  - N households, connected to a distribution manager
  - households submit loads for execution
  - load submission probability: daily demand curve
  - load duration: random, between 1 and D steps
  - 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]
  - define household value as V=loads_executing/execution_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_{lim}=1.5
  - add rewards structure for value V



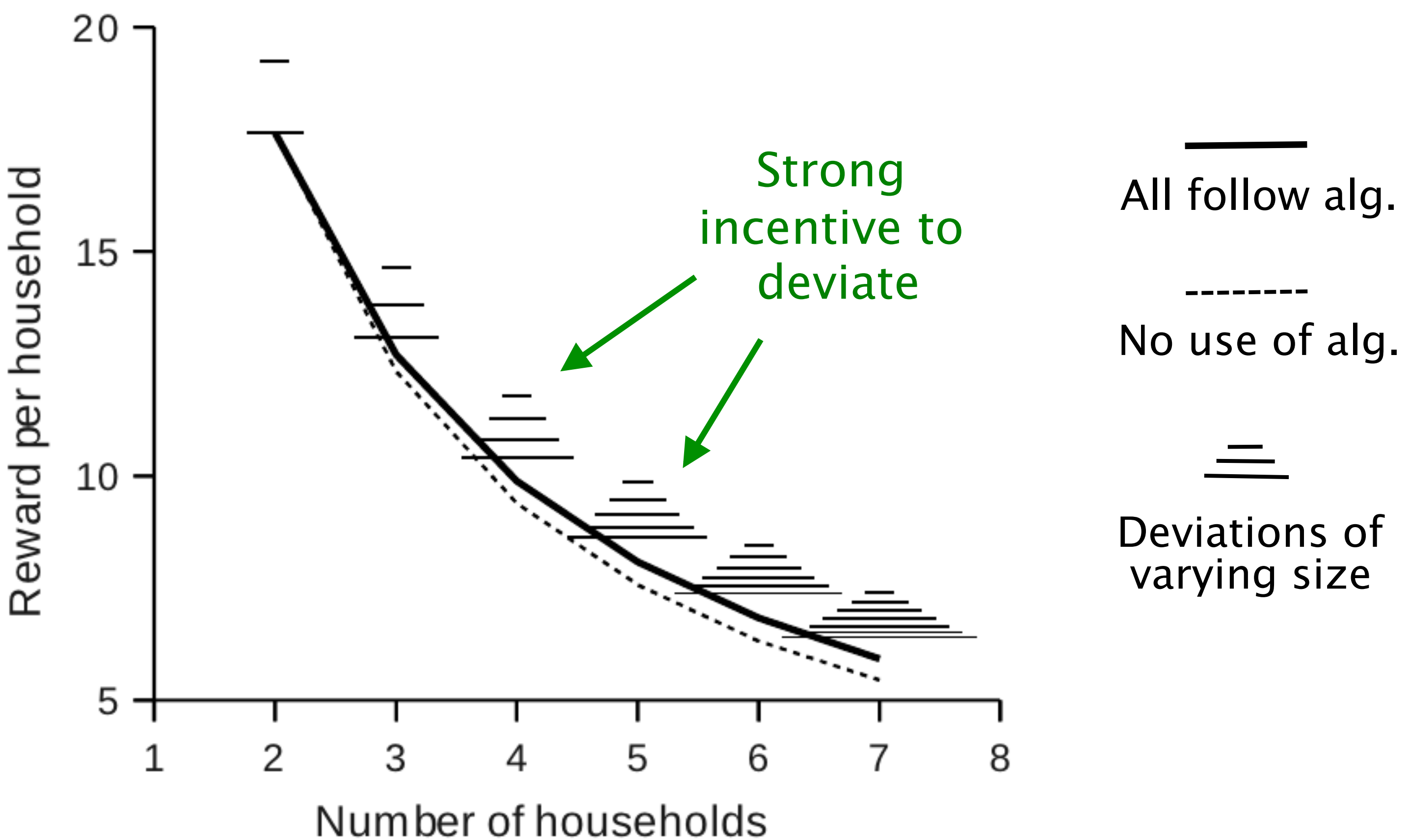
- Built/analysed models
  - for N=2,…,7 households

N	States	Transitions
5	743,904	2,145,120
6	2,384,369	7,260,756
7	6,241,312	19,678,246

- Step 1: assume all households follow algorithm of [HS'11] (MDP)
  - obtain optimal value for P_{start}
- Step 2: introduce competitive behaviour (SMG)
  - allow coalition C of households to deviate from algorithm

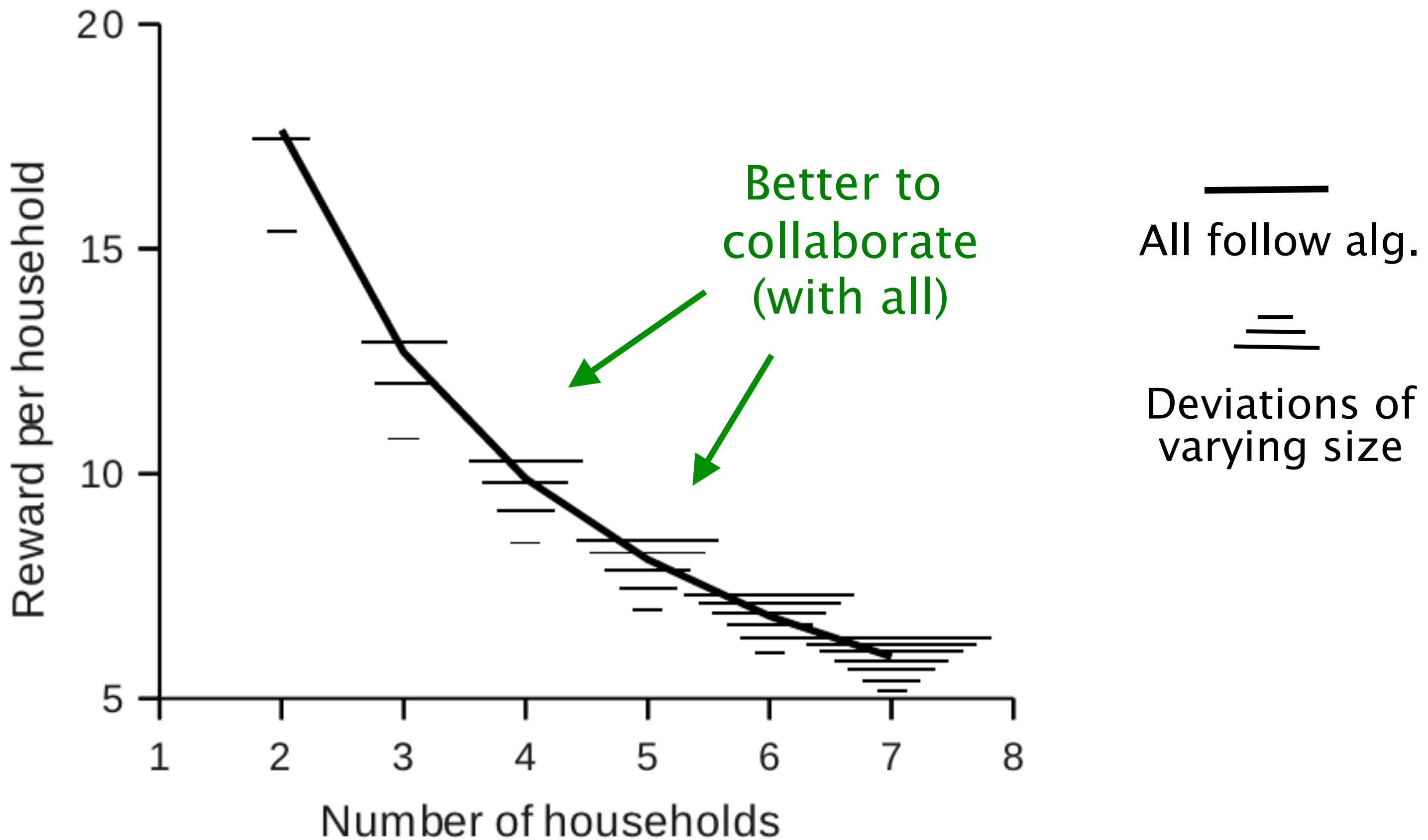
Results: Competitive behaviour

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



Results: Competitive behaviour

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



Conclusions

- Conclusions
  - verification for stochastic systems with competitive behaviour
  - modelled as stochastic multi-player games
  - new temporal logic rPATL for property specification
  - rPATL model checking algorithm based on num. fixed points
  - prototype model checker PRISM-games
  - case studies: energy management for microgrid
- Future work
  - more realistic classes of strategy, e.g. partial information
  - further objectives, e.g. Nash equilibria, multiple objectives, …
  - new application areas, security, randomised algorithms, …