PRISM-Games 2.0: A Tool for Multi-objective Strategy Synthesis for Stochastic Games

Marta Kwiatkowska[1], David Parker[2(✉)], and Clemens Wiltsche[1]

[1] Department of Computer Science, University of Oxford, Oxford, UK
[2] School of Computer Science, University of Birmingham, Birmingham, UK
d.a.parker@cs.bham.ac.uk

Abstract. We present a new release of PRISM-games, a tool for verification and strategy synthesis for stochastic games. PRISM-games 2.0 significantly extends its functionality by supporting, for the first time: (i) *long-run average* (mean-payoff) and *ratio reward* objectives, e.g., to express energy consumption per time unit; (ii) strategy synthesis and Pareto set computation for *multi-objective* properties; and (iii) *compositional* strategy synthesis, where strategies for a stochastic game modelled as a composition of subsystems are synthesised from strategies for individual components using assume-guarantee contracts on component interfaces. We demonstrate the usefulness of the new tool on four case studies from autonomous transport and energy management.

1 Introduction

Automatic verification and strategy synthesis are techniques for analysing probabilistic systems. They can be used to produce formal guarantees with respect to quantitative properties such as safety, reliability and efficiency. For example, they can be employed to synthesise controllers in applications such as autonomous vehicles, network protocols and robotic systems. These often operate in uncertain and adverse environments, models of which require both stochasticity, e.g., to represent noise, failures or delays, and game-theoretic aspects, to model non-cooperative agents or uncontrollable events.

PRISM-games is a tool for verification and strategy synthesis for turn-based stochastic multi-player games. The original version focused on model checking for the temporal logic rPATL [7], used to express zero-sum properties in which two opposing sets of players aim to minimise or maximise a single objective: either the probability of an event or the expected reward accumulated before it occurs. It has been successfully applied to, for example, autonomous driving, self-adaptive systems, computer security and user-centric networks [16].

In this paper, we present PRISM-games 2.0, which significantly extends functionality in several directions. First, it supports strategy synthesis for *long-run* properties, such as *average* (mean-payoff) and *ratio* rewards. This provides the ability to express properties of systems that run autonomously for long periods of time, and to specify measures such as energy consumption per time unit.

M. Chechik and J.-F. Raskin (Eds.): TACAS 2016, LNCS 9636, pp. 560–566, 2016.
DOI: 10.1007/978-3-662-49674-9_35

Secondly, a key new area of functionality is support for *multi-objective* properties, which enables the exploration of trade-offs, such as between performance and resource requirements. Specifically, we allow Boolean combinations of objectives expressed as expected total rewards (for *stopping games*), and expected mean-payoffs or ratios of expected mean-payoffs (in so-called *controllable multi-chain* games), as well as conjunctions of almost sure satisfaction for mean-payoffs and ratio rewards (in general games). The tool also performs computation and visualisation of the *Pareto sets* representing the optimal achievable trade-offs.

Thirdly, PRISM-games 2.0 facilitates *compositional* system development. This is done through *assume-guarantee* strategy synthesis, based on contracts over component interfaces that ensure cooperation between the components to achieve a common goal. For example, if one component satisfies the goal B under an assumption A on its environment (i.e. $A \rightarrow B$), while the other component ensures that the assumption A is satisfied, we can compose strategies for the components into a strategy for the full system achieving B. Multi-objective strategy synthesis, e.g., for an implication $A \rightarrow B$, can be conveniently employed to realise such assume-guarantee contracts. Again, Pareto set computation can be performed to visualise the relationship between properties and across interfaces.

In this paper, we summarise the algorithms and implementation [2,3,8,9,15] behind the new functionality in PRISM-games 2.0, describe its usage in the tool, and illustrate the benefits it brings with results from four case studies drawn from autonomous systems and energy management.

Related Tools. For stochastic games, there is support for qualitative verification in GIST [6] and partial support in the general purpose game solver GAVS+ [10], but there are no tools for multi-objective or compositional analysis. Multi-objective verification for the simpler model of Markov decision processes is available in PRISM [13] (for LTL and expected total reward objectives) and MultiGain [4] (for mean-payoff objectives), but not for stochastic games. Analysis of Nash equilibria (which also balance contrasting objectives for different players) can be performed with EAGLE [14] or PRALINE [5], but only for non-stochastic games. Lastly, Uppaal Stratego [11] performs strategy synthesis against quantitative properties, but with a focus on real-time systems.

2 Modelling and Property Specification Languages

Compositional Modelling. PRISM-games supports action-labelled turn-based stochastic games (henceforth often simply called *games*), which are specified in an extension of the native PRISM modelling language [13]. Version 2.0 adds a compositional modelling approach to facilitate assume-guarantee strategy synthesis for 2-player stochastic games. A *top-level system* consists of several *subsystems* (component games), which are combined using the game composition operator introduced in [3]. This composition synchronises on shared actions, and actions controlled by Player 1 in subsystems are controlled by Player 1 in the composition, thus enabling composition of the synthesised Player 1 strategies.

```
smg  system "S1" || "S2" endsystem

system "S1" G1 endsystem
module G1
    s : [0..2] init 1;
    [d!] s=0 → (s'=1);
    [q1!] s=0 → (s'=1);
    [a?] s=1 → (s'=2);
    [b?] s=1 → 0.5 : (s'=1)
            + 0.5 : (s'=2);
    [] s=2 → (s'=0);
    [a?] s=2 → (s'=2);
endmodule

system "S2" G2 endsystem
module G2
    t : [0..2] init 1;
    [a!] t=0 → 0.5 : (t'=1)
            + 0.5 : (t'=2);
    [q1!] t=0 → (t'=1);
    [b?] t=1 → (t'=2);
    [b?] t=1 → 0.5 : (t'=1)
            + 0.5 : (t'=2);
    [] t=2 → (t'=0);
endmodule

rewards "r1"
    [a] true : 1;
endrewards
rewards "r2"
    [d] true : 1;
    [b] true : 1;
endrewards
rewards "r3"
    [b] true : 1;
endrewards
rewards "c"
    [a] true : 1;
    [b] true : 1;
endrewards
```

Fig. 1. A PRISM-games 2.0 model of a multi-component multi-objective game.

Each subsystem consists of a set of *modules*, which are combined using the original parallel composition of PRISM-games (which ignores player identity). Transitions of modules are specified using guarded commands, optionally labelled with action names (but omitted for non-synchronising transitions). Transitions may be assigned to different players in different subsystems. This is done by tagging an action name `a` with ! or ?, where [a!] assigns the `a`-transition to Player 1 and [a?] to Player 2. No state can have outgoing transitions labelled by both ! and ? since we work with turn-based games. Figure 1 shows a model for a system consisting of two subsystems.

Property Specifications. PRISM-games focuses on *strategy synthesis* for stochastic multi-player games, i.e., finding player strategies that satisfy some winning condition, irrespective of the (finite) strategies of any other players in the game. PRISM-games 2.0 adds *multi-objective queries* (MQs): Boolean combinations of reward-based objectives. Rewards are specified by a *reward structure* that assigns real-valued rewards to transitions of a game (see Fig. 1, right-hand side). We can reason about *total reward* (indefinitely cumulated rewards), *mean payoff* (long-run average reward), or the long-run *ratio* of two rewards. We also support ratios of expected mean-payoffs; expected ratios are synthesised soundly, but not necessarily completely using almost-sure satisfaction of ratio rewards. An *objective* sets a *target* v for a reward value to be exceeded ($\geq$) or upper-bounded ($\leq$). Objectives for the expected mean-payoff of `r`, expected total reward of `r`, and ratio of expected rewards of `r` and `c` are expressed, respectively, as $\mathtt{R}\{\text{“r”}\}_{\geq \mathtt{v}}[\mathtt{S}]$, $\mathtt{R}\{\text{“r”}\}_{\geq \mathtt{v}}[\mathtt{C}]$, and $\mathtt{R}\{\text{“r”}/\text{“c”}\}_{\geq \mathtt{v}}[\mathtt{S}]$, where we use `S` and `C` to denote long-run and cumulative rewards. Almost-sure satisfaction objectives for mean-payoff and ratio rewards are written $\mathtt{P}_{\geq 1}[\mathtt{R}(\mathtt{path})\{\text{“r”}\}_{\geq \mathtt{v}}[\mathtt{S}]]$ and $\mathtt{P}_{\geq 1}[\mathtt{R}(\mathtt{path})\{\text{“r”}/\text{“c”}\}_{\geq \mathtt{v}}[\mathtt{S}]]$. Objectives in an MQ must be of the same type and are combined with the standard Boolean connectives ($\wedge$, $\vee$, $\rightarrow$, $\neg$), but almost-sure satisfaction objectives are only allowed in conjunctions. In the style of rPATL, we use $\langle\langle \mathsf{coalition} \rangle\rangle$ to denote synthesis of strategies for the player(s) in coalition. The following are examples of MQs synthesising strategies for player 1 in a game:

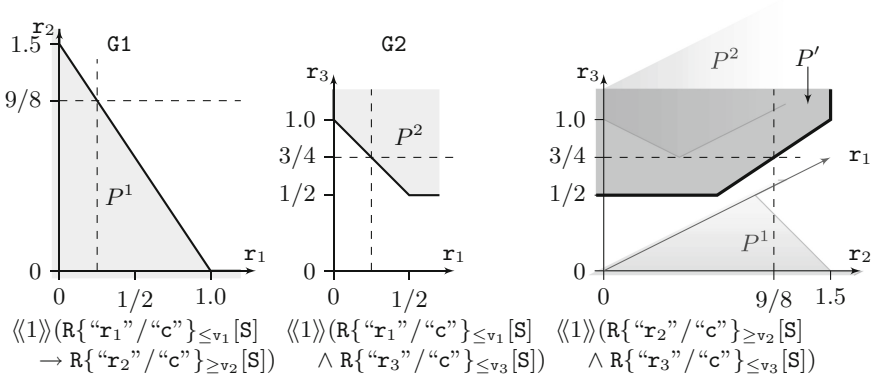

Fig. 2. Pareto sets for the games in Fig. 1, with property specifications beneath the respective sets. On the right is the compositional Pareto set P'. The global target is $(\mathtt{v}_2, \mathtt{v}_3) = (\frac{3}{4}, \frac{9}{8})$, and the local targets can be seen to be consistent with $\mathtt{v}_1 = \frac{1}{4}$.

- $\langle\langle 1 \rangle\rangle(\mathtt{R}\{\text{“}\mathtt{packets\_in}\text{”}/\text{“}\mathtt{time}\text{”}\}_{\leq \mathtt{v_1}}[\mathtt{S}] \rightarrow \mathtt{R}\{\text{“}\mathtt{served}\text{”}/\text{“}\mathtt{time}\text{”}\}_{\geq \mathtt{v_2}}[\mathtt{S}])$ – assuming the expected rate of incoming network packets is at most $\mathtt{v}_1$, the expected rate of serving submitted requests is guaranteed to be at least $\mathtt{v}_2$."
- $\langle\langle 1 \rangle\rangle(\mathtt{R}\{\text{“}\mathtt{passengers}\text{”}\}_{\geq \mathtt{v_1}}[\mathtt{C}] \wedge \mathtt{R}\{\text{“}\mathtt{fuel}\text{”}\}_{\leq \mathtt{v_2}}[\mathtt{C}])$ – "the expected number of passengers transported is at least $\mathtt{v}_1$, while simultaneously ensuring that the expected fuel consumption is at most $\mathtt{v}_2$."

3 Multi-objective Strategy Synthesis

PRISM-games 2.0 implements the multi-objective strategy synthesis methods formulated in [2,8,9], at the heart of which is a fixpoint computation of the sets of achievable targets for multiple reward objectives. For expected total rewards, games must be *stopping*, i.e., terminal states with zero reward must be reached almost surely under all strategies [8]. For expected long-run objectives, games must be *controllable multichain*, i.e., the sets of states that can occur in any maximal end component are almost surely reachable [15].

MQs with objectives of all types are converted into a unified fixpoint computation. In particular, Boolean combinations of expectation objectives are converted to conjunctions by selecting appropriate *weights* for the individual objectives [8]. Then, at each state of the game we iteratively compute polytopic sets of achievable vectors, with each dimension corresponding to one objective. Performance can be improved by computing successive polytopes using in-place (i.e. Gauss-Seidel) updates, as well as rounding the corners of the polytopes at every iteration (which comes at the cost of precision) [9]. We then construct succinct strategies with *stochastic memory updates*, that win by maintaining the target below the expected value of the memory elements, which are the extreme points of the polytopes at the respective states.

Table 1. Performance. The forward slash (/) indicates values for separate components.

Case study	Model		Objectives		Synthesis	
	Components	States	#	Type	Accuracy	Time[s]
UAV	1	6251	2	exp. total, Pareto	0.1	652
UAV	1	6251	2	exp. total, Pareto	0.01	871
AD(Charlton)	1	501	3	exp. total	0.001	2603
AD(Islip)	1	1527	3	exp. total	0.001	1968
Power(0)	2	3456/3456	2/2	a.s. ratio	0.001	175/172
Power(1)	2	11400/11400	2/2	a.s. ratio	0.001	2261/2298
Power$^+$(0)	2	7296/7296	3/3	a.s. ratio	0.01	586/484
Power$^+$(1)	2	24744/24744	3/3	a.s. ratio	0.01	3325/2377
Temp(w)	3	1478/1740/1478	3/2/3	exp. ratio	0.05	829/69/734
Temp(w)	3	1478/1740/1478	3/2/3	exp. ratio	0.01	860/92/2480
Temp(v)	3	1478/1740/1478	3/2/3	exp. ratio	0.05	678/27/621
Temp(v)	3	1478/1740/1478	3/2/3	exp. ratio	0.01	3370/34/8605

The implementation uses the Parma Polyhedra Library [1] for symbolic manipulation of convex sets. Stochastic games are stored in an explicit-state fashion and analysed using an extension of PRISM's Java-based "explicit" engine.

Pareto Sets. An MQ is achievable for all targets in the *achievable set*; its frontier is the *Pareto set*, containing the targets that cannot be improved in any direction without degrading another. The achievable set for Boolean combinations is the union of the convex achievable sets obtained for the respective weights. The Pareto sets can be visualised by the user selecting two-dimensional *slices*.

4 Compositional Strategy Synthesis

We leverage assume-guarantee verification rules for probabilistic automata (i.e., games with only a single player) for assume-guarantee strategy synthesis in two-player games [3]. Given a system $\mathcal{G}$ composed of subsystems $\mathcal{G}_1$, $\mathcal{G}_2$, ..., a designer supplies respective *local property specifications* φ_1, φ_2, ... via the construct `comp(`φ_1`, `φ_2`, ...)`. By synthesising *local strategies* π_i for $\mathcal{G}_i$ satisfying φ_i, a *global strategy* π can be constructed for $\mathcal{G}$. Using *assume-guarantee rules*, one can then derive a *global property* φ for $\mathcal{G}$ that is satisfied by π. The rules require *fairness* conditions, and we write $\mathcal{G}^\pi \models^{\mathsf{u}} \varphi$ if the Player 1 strategy π satisfies φ against all unconditionally fair Player 2 strategies. For example, the rule:

$$\frac{\mathcal{G}_1^{\pi_1} \models^{\mathsf{u}} \varphi^A \qquad \mathcal{G}_2^{\pi_2} \models^{\mathsf{u}} \varphi^A \rightarrow \varphi^G}{(\mathcal{G}_1 \parallel \mathcal{G}_2)^{\pi_1 \| \pi_2} \models^{\mathsf{u}} \varphi^G} \qquad \text{(Asym)}$$

states that Player 1 wins with strategy $\pi_1 \parallel \pi_2$ for φ^G in the top-level system if π_2 in $\mathcal{G}_2$ achieves φ^G under the contract $\varphi^A \rightarrow \varphi^G$, and π_1 in $\mathcal{G}_1$ satisfies φ^A. Reward structures in shared objectives may only involve synchronised actions.

Compositional Pareto Sets. We compositionally compute a Pareto set for the property φ of the top-level system, which is an under-approximation of the Pareto set computed directly on the monolithic system. For a target in the compositional Pareto set, the targets for the local property specifications φ_i can be derived, so that the local strategies can be synthesised (see Fig. 2).

5 Case Studies and Tool Availability

We illustrate the new functionality in PRISM-games 2.0 with four case studies, as follows. "UAV": we compute Pareto sets for a UAV performing reconnaissance of roads, reacting to inputs from a human operator, under a conjunction of expected total rewards [12]. "AD(V)": we synthesise a strategy to steer an autonomous car through a village V, reacting to its environment such as pedestrians, or traffic jams, under a conjunction of expected total rewards [9]. "Power": we maximise uptime of two components in an aircraft electrical power network, reacting to generator failures and switch delays d; each component has a conjunction of almost-sure satisfaction of ratio rewards. We use assume-guarantee strategy synthesis for two model variants, with (resp. without) modelling an interface, denoted Power$^{+}(d)$ (resp. Power(d)) [2]. "Temp": we control the temperature in three adjacent rooms, reacting to the outside temperature and whether windows are opened; and use Boolean combinations of expected ratios. We use assume-guarantee strategy synthesis for two model variants, denoted Temp(w) and Temp(v) [15]. Table 1 summarises the tool's performance on these case studies on a 2.8 GHz PC with 32 GB RAM. We observe that scalability mostly depends on the number of objectives, the state space size and accuracy, but our compositional approach greatly increases the viable state space sizes.

PRISM-games 2.0 is open source, released under GPL, available from [16].

Acknowledgement. This work has been supported by the ERC Advanced Grant VERIWARE and EPSRC Mobile Autonomy Programme Grant.

References

1. Bagnara, R., Hill, P., Zaffanella, E.: The parma polyhedra library. Sci. Comput. Program. **72**(1–2), 3–21 (2008)
2. Basset, N., Kwiatkowska, M., Topcu, U., Wiltsche, C.: Strategy synthesis for stochastic games with multiple long-run objectives. In: TACAS 2015, pp. 256–271 (2015)
3. Basset, N., Kwiatkowska, M., Wiltsche, C.: Compositional controller synthesis for stochastic games. In: Baldan, P., Gorla, D. (eds.) CONCUR 2014. LNCS, vol. 8704, pp. 173–187. Springer, Heidelberg (2014)
4. Brázdil, T., Chatterjee, K., Forejt, V., Kučera, A.: MultiGain: a controller synthesis tool for MDPs with multiple mean-payoff objectives. In: TACAS 2015
5. Brenguier, R.: PRALINE: a tool for computing Nash equilibria in concurrent games. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 890–895. Springer, Heidelberg (2013)

6. Chatterjee, K., Henzinger, T.A., Jobstmann, B., Radhakrishna, A.: GIST: a solver for probabilistic games. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 665–669. Springer, Heidelberg (2010)
7. Chen, T., Forejt, V., Kwiatkowska, M., Parker, D., Simaitis, A.: Automatic verification of competitive stochastic systems. FMSD **43**(1), 61–92 (2013)
8. Chen, T., Forejt, V., Kwiatkowska, M., Simaitis, A., Wiltsche, C.: On stochastic games with multiple objectives. In: Chatterjee, K., Sgall, J. (eds.) MFCS 2013. LNCS, vol. 8087, pp. 266–277. Springer, Heidelberg (2013)
9. Chen, T., Kwiatkowska, M., Simaitis, A., Wiltsche, C.: Synthesis for multi-objective stochastic games: an application to autonomous urban driving. In: QEST 2013 (2013)
10. Cheng, C.-H., Knoll, A., Luttenberger, M., Buckl, C.: GAVS+: an open platform for the research of algorithmic game solving. In: Abdulla, P.A., Leino, K.R.M. (eds.) TACAS 2011. LNCS, vol. 6605, pp. 258–261. Springer, Heidelberg (2011)
11. David, A., Jensen, P.G., Larsen, K.G., Mikučionis, M., Taankvist, J.H.: Uppaal Stratego. In: Baier, C., Tinelli, C. (eds.) TACAS 2015. LNCS, vol. 9035, pp. 206–211. Springer, Heidelberg (2015)
12. Feng, L., Wiltsche, C., Humphrey, L., Topcu, U.: Controller synthesis for autonomous systems interacting with human operators. In: ICCPS (2015)
13. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011)
14. Toumi, A., Gutierrez, J., Wooldridge, M.: A tool for the automated verification of Nash equilibria in concurrent games. In: Leucker, M., Rueda, C., Valencia, F.D. (eds.) ICTAC 2015. LNCS, vol. 9399, pp. 583–594. Springer, Heidelberg (2015)
15. Wiltsche, C.: Assume-Guarantee Strategy Synthesis for Stochastic Games. Ph.D. thesis, University of Oxford (2016, forthcoming)
16. PRISM-games website. www.prismmodelchecker.org/games/