GAVS+: An Open Platform for the Research of Algorithmic Game Solving

Chih-Hong Cheng[1], Alois Knoll[1], Michael Luttenberger[1], and Christian Buckl[2]

[1] Department of Informatics, Technische Universität München
Boltzmann Str. 3, Garching D-85748, Germany
[2] Fortiss GmbH, Guerickestr. 25, D-80805 München, Germany
http://www6.in.tum.de/~chengch/gavs

Abstract. This paper presents a major revision of the tool GAVS. First, the number of supported games has been greatly extended and now encompasses in addition many classes important for the design and analysis of programs, e.g., it now allows to explore concurrent / probabilistic / distributed games, and games played on pushdown graphs. Second, among newly introduced utility functions, GAVS+ includes features such that the user can now process and synthesize planning (game) problems described in the established STRIPS/PDDL language by introducing a slight extension which allows to specify a second player. This allows researchers in verification to profit from the rich collection of examples coming from the AI community.

1 Introduction

We present a major revision of the open-source tool GAVS[1], called GAVS+, which now targets to serve as an open platform for the research community in algorithmic game solving. In addition, GAVS+ is meant to serve as a playground for researchers thinking of mapping interesting problems to game solving.

GAVS+ has three main goals: (i) support of game types currently under active research; many times an implementation of a solver is only hard to come by, or only partial implementations exist; (ii) support of different input and output formats in order to allow for both interoperability with other tools and for easy access of existing collections of models, examples, and test cases in concrete application domains; (iii) ease of use by a unified graphical user interface (GUI) which allows to graphically specify the game and explore the computed solution. The last requirement is partially fulfilled by the previous version of GAVS+: the GUI allows to visualize two-player, turn-based games on finite graph, solve the game, and store intermediate results in order to visualize the algorithms step-by-step. This also makes it a very useful tool for teaching these algorithms. In this release, we have concentrated on goals (i) and (ii).

Regarding goal (i), we have added support for several games of practical relevance: we have introduced support for stochastic [10], concurrent [6], distributed [9] games, as well as games played on pushdown graphs [2]. We opted for these games as they arise

[1] Short for "Game Arena Visualization and Synthesis".

P.A. Abdulla and K.R.M. Leino (Eds.): TACAS 2011, LNCS 6605, pp. 258–261, 2011.

Game type (visualization)	Implemented algorithms
Fundamental game	Symbolic: (Co-)reachability, Büchi, Weak-parity, Staiger-Wagner Explicit state: Parity (discrete strategy improvement) Reduction: Muller, Streett
Concurrent game	**Sure reachability, Almost-sure reachability, Limit-sure reachability**
Pushdown game‡	**Reachability (positional min-rank strategy, PDS strategy), Büchi (positional min-rank strategy), Parity (reduction)**
Distributed game	**Reachability (bounded distributed positional strategy for player-0)**
Markov decision process	**Policy iteration, Value iteration, Linear programming (LP)**
Simple stochastic game	**Shapley (value iteration), Hoffman-Karp (policy iteration)**

Fig. 1. Game types and implemented algorithms in GAVS+ (games marked bold are extensions to the previous version [3]), where "‡" indicates that visualization is currently not available

either naturally in the context of parallel resp. recursive programs, or a widely used for systems with uncertainties.

As to goal (ii), one feature amongst newly introduced utilities in GAVS+ is the ability to be applied to existing planning problems formalized in PDDL (level 1; the STRIPS fragment) [8]. We consider it as an interesting feature, as it allows researchers in the verification community to access the huge collection of planning problems coming from the artificial intelligence (AI) and the robotics community.

Although a large collection of PDDL specific solvers is available, only one tool [7] is based on a similar approach combining planning and games. Unfortunately, this tool is not publicly available. Furthermore, the goal of [7] is to "study the application and extension of planning technology for general game playing" whereas our goal is rather to allow researchers in the verification community to profit from the rich collection of models coming from the AI community.

In the rest of the paper, first we give a very brief overview on the newly supported games, and then illustrate our integration and extension of STRIPS/PDDL into GAVS+. Due to space limit we only outline two tiny examples using GAVS+; for details and complete functionalities we refer interested readers to our website.

2 Supported Games in GAVS+

For a complete overview on all supported games we refer the reader to Figure 1, here, we only give a brief description of the newly added games.

Concurrent games [6] are used to capture the condition when the next location is based on the combined decision simultaneously made by control and environment. When considering randomized strategies in reachability games (CRG), efficient algorithms to compute sure, almost-sure, and limit-sure winning regions are available [6].

Stochastic games model systems with uncertainty. The classical model of *Markov decision process (MDP,* $1\frac{1}{2}$*-player game)* [12] is widely used in economics and machine learning and considers a single player who has to work against an environment exhibiting random behavior. Adding an opponent to MDPs, one obtains *stochastic (*$2\frac{1}{2}$*-player) games* [10]. Currently we focus on the subclass of simple stochastic games (SSG) [5]; many complicated games can be reduced to SSGs or solved by algorithms

similar to algorithms solving SSG. For implemented algorithms for MDP and SSG, we refer readers to two survey papers [12,5] for details.

Games on pushdown graphs (APDS) arise naturally when recursive programs are considered. Symbolic algorithms exist for reachability and Büchi winning conditions [2], and for parity conditions, a reduction[2] to two-player, finite-state parity games based on summarization is possible [2]. An example for the interactive reachability simulation of an APDS using GAVS+ is illustrated with Figure 2: the user can act as the role of player-1 (environment) by selecting the rewriting rule, while GAVS+ updates the cost for the positional min-rank strategy and output the next move for player-0 (control).

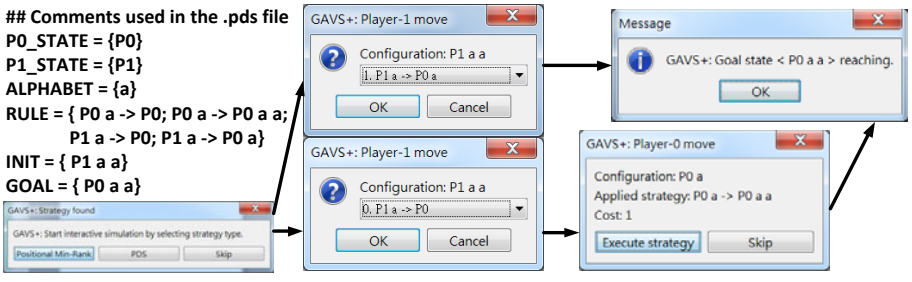

Fig. 2. An APDS in textual form and screenshots of the interactive simulation using GAVS

Distributed games [9] are games formulating multiple processes with no interactions among themselves but only with the environment. Generating strategies for such a game is very useful for distributed systems, as a strategy facilitates orchestration of interacting components. Although the problem is undecidable in general [9], finding a distributed positional strategy for player-0 of such a system ($PositionalDG_0$), if it exists, is a practical problem. As $PositionalDG_0$ is NP-complete for reachability games [4], we modify the SAT-based witness algorithms in [1] and implement a distributed version [4] for bounded reachability games.

3 Extension of STRIPS/PDDL for Game Solving: An Example

To connect the use of games with concrete application domains, GAVS+ offers the translation scheme from PDDL to symbolic representation of games. Here we consider a simplified scenario of a robot with two humanoid arms[3]. For details (game domain and problem as PDDL files, synthesized strategies), we refer readers to the GAVS+ software package in the website.

– **(Fault-free planning)** In the case of fault-free planning, GAVS+ reads the normal PDDL file, performs the forward symbolic search and generates a totally ordered plan (action sequence) similar to other tools.

[2] Currently, as the algorithm introduces an immediate exponential blowup in the graph, it is difficult to solve the game using the built-in algorithm specified in [11].

[3] Motivated by the scenario in our chair: http://www6.in.tum.de/Main/ResearchEccerobot

– **(Fault-tolerant strategy generation)** We outline steps for game creation and solving when faults are introduced in GAVS+ using this example.
  1. *(Fault modeling)* When an arm is out-of-service, the tension over the artificial muscle is freed, and the object under grasp falls down to the ground (fault-effect). It is also assumed that for the robot, at most one arm can be out-of-service during its operation cycle (fault-frequency). The above behavior can be modeled as a fault action in the PDDL domain.
  2. *(State Partitioning)* Currently in GAVS+, we simply introduce a binary predicate `P0TRAN` on each action in the domain to partition player-0 and player-1 states and transitions.
  3. *(Synthesis)* By modeling the domain and problem using PDDL, GAVS+ synthesizes strategies to perform tasks while resisting the potential loss of one arm: the strategy is a FSM outputted to a separate file using Java-like formats.

References

1. Alur, R., Madhusudan, P., Nam, W.: Symbolic computational techniques for solving games. International Journal on Software Tools for Technology Transfer (STTT) 7(2), 118–128 (2005)
2. Cachat, T.: Games on Pushdown Graphs and Extensions. PhD thesis, RWTH Aachen (2003)
3. Cheng, C.-H., Buckl, C., Luttenberger, M., Knoll, A.: GAVS: Game arena visualization and synthesis. In: Bouajjani, A., Chin, W.-N. (eds.) ATVA 2010. LNCS, vol. 6252, pp. 347–352. Springer, Heidelberg (2010)
4. Cheng, C.-H., Rueß, H., Knoll, A., Buckl, C.: Synthesis of fault-tolerant embedded systems using games: From theory to practice. In: Jhala, R., Schmidt, D. (eds.) VMCAI 2011. LNCS, vol. 6538, pp. 118–133. Springer, Heidelberg (2011)
5. Condon, A.: On algorithms for simple stochastic games. Advances in Computational Complexity theory 13, 51–73 (1993)
6. De Alfaro, L., Henzinger, T., Kupferman, O.: Concurrent reachability games. Theoretical Computer Science 386(3), 188–217 (2007)
7. Edelkamp, S., Kissmann, P.: Symbolic exploration for general game playing in pddl. In: ICAPS-Workshop on Planning in Games (2007)
8. Fox, M., Long, D.: Pddl2.1: An extension to pddl for expressing temporal planning domains. Journal of Artificial Intelligence Research 20(1), 61–124 (2003)
9. Mohalik, S., Walukiewicz, I.: Distributed games. In: Pandya, P.K., Radhakrishnan, J. (eds.) FSTTCS 2003. LNCS, vol. 2914, pp. 338–351. Springer, Heidelberg (2003)
10. Shapley, L.: Stochastic games. Proceedings of the National Academy of Sciences of the United States of America 39, 1095 (1953)
11. Vöge, J., Jurdziński, M.: A discrete strategy improvement algorithm for solving parity games. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 202–215. Springer, Heidelberg (2000)
12. White, C., et al.: Markov decision processes. European Journal of Operational Research 39(1), 1–16 (1989)