Games in System Design and Verification

Thomas A. Henzinger

EPFL, Switzerland, and University of California, Berkeley, USA

Short abstract. We review some classical and recent results about graph games, and their applications in the design and verification of component-based reactive systems.

Extended abstract. We consider *graph games*, where a token is moved along the edges of a directed graph in an infinite sequence of moves. The result of playing a graph game, or *play*, is an infinite path through the graph. Graph games can serve as models for discrete reactive systems: the vertices represent system states; the players represent system components; the moves represent system transitions; the plays represent system behaviors; and the objectives represent system specifications.

Graph games come in different flavors depending, for example, on the number of players, the information that each player has available for choosing a move, the mechanism by which the successor vertex is determined from a given choice of moves, and the type of objective for each player. The variety of possible graph games permits the modeling of different interaction paradigms for component-based systems. Asynchronous interaction is naturally modeled using *turn-based* games: at each vertex, one of the players chooses a move, which determines the successor vertex [18]. Synchronous interaction, on the other hand, requires *concurrent* games: at each vertex, all players choose moves simultaneously and independently, and the combination of choices determines the successor vertex [7]. In *deterministic* games, a move or combination of moves determines a unique successor vertex; in *stochastic* games, a move or combination of moves determines a probability distribution over successor vertices [12]. Note that deterministic games can model nondeterministic choice through a choice of available moves; stochastic games model both nondeterministic and probabilistic choice.

The objectives of the players in a graph game can be qualitative or quantitative. A *qualitative* objective is a set of winning plays. For modeling reactive systems, the interesting class of qualitative objectives are the ω-regular sets, which can express common safety and liveness properties [15]. A *quantitative* objective requires a labeled graph, where each vertex (or edge) is labeled with a rational payoff for each player. Then, the goal of a player is to maximize a reward, which is a function that maps each play to a real number, for instance, the supremum, limit sum, or limit average of all payoffs along the play. Payoffs can be used to represent costs, delays, or other resource values of reactive systems [4].

The solution problem for a graph game asks for the *value* of each player at each vertex. For turn-based deterministic games with qualitative objectives, the value of a player i at a vertex q is binary: the value $V_i(q)$ is 1 if player i has a strategy to ensure that, when the game starts from vertex q, the outcome of the game is a play that is winning for player i; and else $V_i(q) = 0$. For other games, values are real numbers. For turn-based deterministic games with quantitative objectives, the real value $V_i(q)$ indicates the maximal reward achievable by player i when the game starts from vertex q. For stochastic games or concurrent games with qualitative objectives, the real value $V_i(q)$ represents[1] the maximal probability for player i of winning from vertex q. While probabilities obviously arise in stochastic games, it is worth noting that they enter into the analysis of concurrent games even in the deterministic case. Consider, for example, a two-player concurrent game that proceeds in an infinite number of rounds: in each round, player 1 chooses a bit b_1 and player 2 chooses a bit b_2, and the successor state is (b_1, b_2). Suppose that the objective of player 1 is to visit one of the two states (0,0) and (1,1). An optimal strategy for player 1 is to choose, in each round, $b_1 = 0$ with probability 1/2, and $b_1 = 1$ with probability 1/2. This strategy ensures that the set $\{(0,0),(1,1)\}$ is visited with probability 1. However, no *pure* (i.e., nonrandomized) strategy can achieve the value 1 for player 1 is this game.

Besides the need for randomization, another important property of strategies concerns the amount of memory needed by a strategy: a strategy for player i is a recipe for choosing the moves of player i during a play, and this recipe may, in general, depend on the unbounded history of the play. In some cases of interest, however, *memoryless* optimal strategies exist. Such strategies can be implemented by reactive systems without state. In other cases, the required memory is, at least, finite [13].

A setting with two players, one representing a reactive system and the other representing the environment, gives rise to *zero-sum* graph games, where the objectives of the two players are complementary. For example, one may ask if the system has a strategy to satisfy its specification no matter what the environment does; this is called the strategy or controller synthesis question for reactive systems, which dates back to a problem formulated by Church [3, 22]. The central result for zero-sum graph games are the determinacy theorems of Martin, which state that $V_1(q) + V_2(q) = 1$ for all vertices q and all qualitative[2] Borel objectives (i.e., the winning sets are Borel sets in the Cantor topology on infinite sequences of vertices) [16, 17]. Special cases of zero-sum games can be solved in NP via proving the existence of memoryless optimal strategies, and in NP $\cap$ coNP if memoryless strategies suffice for both an objective and its complement. This is the case, for example, for turn-based deterministic games with ω-regular objectives in parity form [20, 11]. The question if these games can indeed be solved in P remains, due to its equivalence with μ-calculus model checking, one of the major open questions in verification theory.

A setting with two or more players that represent the components of a reactive system, and whose objectives represent component specifications, gives rise to *nonzero-sum* graph games. In this case, it is interesting to study notions of rational behavior for the individual players as captured by Nash equilibria, an area that remains largely uninvestigated [5, 6].

In this talk we survey some of the known results and open problems about graph games. We present a convenient notation for conversing about graph games based on ATL (Alternating-time Temporal Logic) [2]. Then, in addition to controller synthesis [23, 21], we discuss some classical and some recent applications of graph games in system design and verification, including simulation relations between reactive systems [19, 14], the realizability of reactive specifications [10, 1], early counterexample detection in model checking [9], and the composition of interface protocols [8].

Acknowledgment. We thank Krishnendu Chatterjee for many stimulating discussions and fascinating insights about graph games.

[1]The value may not be achievable by a single strategy, called an *optimal* strategy, but may only be approximable by a family of strategies.

[2]An analogous result holds for quantitative Borel objectives as well.

References

[1] M. Abadi, L. Lamport, and P. Wolper. Realizable and unrealizable concurrent program specifications. In *International Colloquium on Automata, Languages, and Programming*, pages 1–17. Lecture Notes in Computer Science 372, Springer, 1989.

[2] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49:672–713, 2002.

[3] J.R. Büchi and L.H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the AMS*, 138:295–311, 1969.

[4] A. Chakrabarti, L. de Alfaro, T.A. Henzinger, and M. Stoelinga. Resource interfaces. In *Embedded Software*, pages 117–133. Lecture Notes in Computer Science 2855, Springer, 2003.

[5] K. Chatterjee, T.A. Henzinger, and M. Jurdziński. Games with secure equilibria. In *Logic in Computer Science*, pages 160–169. IEEE Computer Society, 2004.

[6] K. Chatterjee, M. Jurdziński, and R. Majumdar. On Nash equilibria in stochastic games. In *Computer Science Logic*, pages 26–40. Lecture Notes in Computer Science 3210, Springer, 2004.

[7] L. de Alfaro and T.A. Henzinger. Concurrent ω-regular games. In *Logic in Computer Science*, pages 141–154. IEEE Computer Society, 2000.

[8] L. de Alfaro and T.A. Henzinger. Interface automata. In *Foundations of Software Engineering*, pages 109–120. ACM, 2001.

[9] L. de Alfaro, T.A. Henzinger, and F.Y.C. Mang. Detecting errors before reaching them. In *Computer-Aided Verification*, pages 186–201. Lecture Notes in Computer Science 1855, Springer, 2000.

[10] D.L. Dill. *Trace Theory for the Automatic Hierarchical Verification of Speed-Independent Circuits*. MIT Press, 1989.

[11] E.A. Emerson and C. Jutla. Tree automata, μ-calculus, and determinacy. In *Foundations of Computer Science*, pages 368–377. IEEE Computer Society, 1991.

[12] J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer, 1997.

[13] Y. Gurevich and L. Harrington. Trees, automata, and games. In *Symposium on Theory of Computing*, pages 60–65. ACM, 1982.

[14] T.A. Henzinger, O. Kupferman, and S.K. Rajamani. Fair simulation. *Information and Computation*, 173:64–81, 2002.

[15] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer, 1992.

[16] D.A. Martin. Borel determinacy. *Annals of Mathematics*, 102:363–371, 1975.

[17] D.A. Martin. The determinacy of Blackwell games. *Journal of Symbolic Logic*, 63:1565–1581, 1998.

[18] R. McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65:149–184, 1993.

[19] R. Milner. An algebraic definition of simulation between programs. In *International Joint Conference on Artificial Intelligence*, pages 481–489. Kaufmann, 1971.

[20] A.W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In *Computation Theory*, pages 157–168. Lecture Notes in Computer Science 208, Springer, 1984.

[21] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Principles of Programming Languages*, pages 179–190. ACM, 1989.

[22] M.O. Rabin. *Automata on Infinite Objects and Church's Problem.* AMS, 1972.

[23] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete-event processes. *SIAM Journal of Control and Optimization*, 25:206–230, 1987.