Infinite Games and Verification
(Extended Abstract of a Tutorial)

Wolfgang Thomas

Lehrstuhl für Informatik VII, RWTH Aachen, 52056 Aachen, Germany
thomas@informatik.rwth-aachen.de

Abstract. The purpose of this tutorial is to survey the essentials of the algorithmic theory of infinite games, its role in automatic program synthesis and verification, and some challenges of current research.

1 Background and motivation

The research on infinite games in theoretical computer science is based on a mixture of several motivations:

1. The beautiful classical theory of infinite games, as developed in descriptive set theory, lacks an algorithmic content. Such an algorithmic orientation is provided by the automata theoretic approach to infinite games, originating in work of Church, Büchi, McNaughton, and Rabin about fourty years ago.
2. Determinacy results for infinite games are closely related to complementation results for logics and automata; thus, infinite games help to analyze logical theories (the most prominent being the monadic theory S2S of two successor functions, see e.g. [Th97]).
3. Games are a natural model of reactive computation, and infinite games are thus a faithful representation of nonterminating reactive systems (for which control problems can be solved in terms of providing winning strategies).
4. The model-checking problem for logics like the μ-calculus can be formulated as the question to determine the winner of an infinite game.

The purpose of this tutorial is to explain the core of the algorithmic (and automata theoretic) theory of infinite games. In the present extended abstract, only some key notions are explained, without technical details and proofs. In the tutorial, more will be said about topics of current research as listed in Section 6 of this abstract.

2 The terminological framework

We consider infinite two-person games with perfect information; the two players are called 0 and 1. A game is specified by a directed *game graph* (also called arena of the game) and a *winning condition*, which singles out those infinite plays which are won by player 0 (the others are won by player 1). The game graph $G = (V, E)$ is equipped with a partition of its vertex set V into sets V_0, V_1. A *play* over G from vertex v_0 is a sequence $\pi = v_0 v_1 \ldots$ of vertices built up for $i = 0, 1, \ldots$ as follows: If $v_i \in V_0$ then player 0 chooses v_{i+1} as next vertex (otherwise player 1 chooses v_{i+1}), in both cases respecting the condition that that $(v_i, v_{i+1}) \in E$. The winning condition for player 0 can be given by a logical formula which expresses a certain property of a play π, or by an acceptance condition from the theory of ω-automata applied to π. If the vertex set V is infinite, one usually assumes a finite coloring of the vertices and expresses the winning condition by referring to the induced sequence of vertex colors rather than to the vertices themselves. In the sequel, up to the last section, we assume that the game graph is finite.

A *strategy* for player p is a function which maps each play prefix $v_0 \ldots v$ ending in a vertex $v \in V_p$ to a suitable "next vertex", i.e. some v' with $(v, v') \in E$. In the automata theoretic framework, two special kinds of strategies are essential: A strategy is *positional* (or: *memoryless*) if its value for a play prefix $v_0 \ldots v$ only depends on the last (or "current") vertex v, and it is called *finite-state* if it can be computed by a finite automaton with output (upon reading $v_0 \ldots v$ as input). A *winning strategy* for player 0 leads to a play won by player 0 whatever choice of vertices is done by player 1. One says that player 0 *wins* a given game *from* v_0 if he has a winning strategy for plays starting in v_0. The set of these initial vertices v_0 forms the *winning region* W_0 of player 0. A game is determined if the winning region of player 1 is the complement set $W_1 = V \setminus W_0$. All games to be considered here are determined.

In the classical theory of Gale-Stewart games, a game involves the infinite binary tree as game arena, and the winning condition is just given by an abstract set of plays. The presentation of games by game graphs and logical formulas or by ω-automata (as winning conditions) raises algorithmic problems which are not relevant in the classical framework:

1. To decide for a given vertex v_0 whether it belongs to the winning region of player 0, and
2. if possible to construct a program which executes a winning strategy for player 0, [1] and
3. to minimize the complexity for determining a winning strategy, as well as to reduce the complexity of the strategy itself. For finite-state strategies the latter can mean to minimize the number of states of the respective strategy automata.

By "solving a game" we mean a solution to questions (1) and (2).

Sometimes, a reactive system can be modelled in this framework of infinite games, by identifying player 0 with a control component and player 1 with the environment. The winning condition corresponds to the specification which the control component has to meet under all possible behaviours of the environment.

[1] It should be noted that in general one cannot infer a computable winning strategy for player 0 from the fact that the winning region of player 0 is decidable; see e.g. [Th95].

A solution to question (1) amounts to the test whether the specification allows a solution, item (2) is concerned with the synthesis of a correct controller, and item (3) with optimizations of this synthesis. In the present tutorial, we stay with "linear-time specifications" and the case of complete information; [KV99] is a reference where branching-time specifications and incomplete information are treated.

3 Topological classification of winning conditions

If the winning condition of a game is expressible in propositional temporal logic or in the monadic logic S1S, one can apply the well-known transformation of such formulas into deterministic ω-automata, say with a Muller acceptance condition. Using such a transformation of a formula φ into a deterministic automaton $\mathcal{A}_\varphi$, one can proceed from the given game graph G to the product $G \times \mathcal{A}_\varphi$, in which a play π over G becomes the pair (π, ρ), where ρ is the run of $\mathcal{A}_\varphi$ on input π. For this induced play, the winning condition φ is captured by the Muller acceptance condition applied to ρ. The *Muller games*, i.e. games with a Muller winning condition, are thus a framework general enough for most applications in synthesis and verification.

The set of all plays over a game graph can naturally be viewed as a topological space (called Cantor space for finitely branching graphs and Baire space for countably branching ones). Properties of plays are classified in the so-called Borel hierarchy, its first level consisting of the closed and open sets. For the question of solving games, it is useful to locate the set of winning plays in this hierarchy. Six basic cases have to be distinguished (for more background see e.g. [MP92]):

1. "reachability games" (or "guarantee games"), where a play is a win for player 0 iff it reaches at some time a vertex of a given "target set",
2. "safety games", where a play is a win for player 0 iff it remains within a given set of vertices,
3. games with boolean combinations of conditions 1 and 2 as winning conditions, the so-called "obligation games" or "weak Muller games", where the set of vertices visited in a play determines whether it is a win for player 0,
4. "recurrence games" (or "Büchi games"), where a play is a win for player 0 iff it meets a given set of "target vertices" infinitely often,
5. "persistence games", in which a play is a win for player 0 if from some point onwards, only vertices of a predefined vertex set occur,
6. games with boolean combinations of conditions 4 and 5 as winning conditions, where the set of vertices visited infinitely often in a play determines whether it is a win for player 0.

The games of item 3 are captured by an automata theoretic winning condition due to Staiger and Wagner, the games of item 6 are the Muller games.

The basis of most strategy constructions is the solution of reachability games: Starting from the set T of target vertices, one computes by an inductive process the sets A_i (for $i = 1, 2, \ldots$) of those vertices from which player 0 can ensure to reach T within i moves. If the game graph is finitely branching, the union of the A_i, called the "0-attractor of T", is the winning region of player 0. On this region, there is a positional winning strategy for player 0 (which just has to ensure that the distance to T decreases in each step).

Variants and extensions of this construction also allow to solve the safety games, the recurrence games, and the persistence games, in each case only by means of positional strategies. Another pleasant feature of these games is that their solution is possible in polynomial time (in fact, linear time in the case of reachability and safety games).

4 Game simulations and parity conditions

The weak Muller games and the Muller games involve some complications, mainly due to the fact that positional winning strategies do no more suffice. In fact, there is a sequence of game graphs G_n with $O(n)$ vertices such that any winning strategy solving a certain associated weak Muller game requires a strategy automaton with at least 2^n states. Similarly, for the Muller games a lower bound of $n!$ can be established ([DJW97]). In ω-automata theory, one can consider the Rabin or the Streett condition instead of the Muller condition; in these cases, player 0 (resp. player 1) can win with a positional strategy, but the other player again needs in general some memory in order to win.

At this point, the so-called parity winning conditions are very convenient: They allow to reach the expressive power of the weak Muller and the Muller condition, but at the same time admit solutions by positional winning strategies. A parity winning condition refers to a coloring of the vertices of a game graph by finitely many integers (formally presented by a function $c : V \to C$ with finite $C \subseteq \mathbb{Z}$). With respect to the parity condition, a play $\pi = v_0 v_1 \ldots$ is a win for player 0 if the maximal color occurring infinitely often in the sequence $c(v_0)c(v_1)\ldots$ is even. The weak parity condition just requires that the maximal color occurring at all (rather than infinitely often) has to be even.

A nested computation of attractor sets suffices to solve a weak parity game, including the (polynomial time) construction of positional winning strategies for the two players on their respective winning regions. The solution of parity games is harder; there are exponential time algorithms for computing the winning regions of the two players and corresponding positional winning strategies, and it is presently open whether a polynomial time solution exists. Below we give a more detailed discussion.

To verify that the weak parity condition and the parity condition are indeed enough to capture the weak Muller and the Muller condition, respectively, we use a notion of game simulation (see [Th95]): It involves a transformation of a game over a graph $G = (V, E)$ with winning condition φ into a "simulating game" over a (usually larger) game graph $G' = (V', E')$ with a (usually simpler) winning condition φ'. In the cases considered here, the vertex set V' will be a product $V \times S$ for a finite set S (in which a certain element s_0 is designated). A play π

from v through G will determine a well-defined play π' through G' from (v, s_0). The simulation relation $(G, \varphi) \leq (G', \varphi')$ holds if for a play π over G we have that π satisfies φ iff π' satisfies φ'. It turns out that a weak Muller game over $G = (V, E)$ is simulated by a weak parity game over a graph $G' = (V \times 2^V, E')$, and that a Muller game over $G = (V, E)$ can be simulated by a parity game over a graph $G' = (V \times S, E')$ where S is the set of permutations of the V-elements. The first simulation is a variant of the subset construction (namely, in the second component of a V'-vertex the V-vertices visited so far in a play are collected). The second simulation involves sequences of vertices rather than sets, namely the "latest appearance record", i.e. the visited states in the order of their most recent visit.

One can combine the simulation of weak Muller games and Muller games by weak parity games and by parity games, respectively, with the construction of positional winning strategies for the latter. Altogether one obtains finite-state strategies for the weak Muller games and the Muller games. The idea is to use the auxiliary set S introduced in the game simulation as the set of states of a strategy automaton. Also other games where winning strategies involve memory (e.g., Streett games) can be solved by such a reduction to parity games.

5 Parity games and μ-calculus model-checking

To find an efficient solution for parity games is one of the central open problems in the the verification of state-based systems. As Emerson, Jutla, and Sistla [EJS93] have shown, the model-checking problem for the μ-calculus is polynomial-time reducible to the problem of solving parity games. To show this, one transforms (in polynomial time) a given finite Kripke structure $\mathcal{K}$ with designated state s_0 and a μ-calculus formula φ into a game graph $G_{(\mathcal{K}, s_0, \varphi)}$ equipped with a parity condition. The vertices of the game graph are pairs (s, ψ) or (s, X) where s is a state s from $\mathcal{K}$, ψ is a subformula of φ, and X is a fixed point variable occurring in φ. The number of colors reflects the alternation depth of φ. The construction ensures that $(\mathcal{K}, s_0) \models \varphi$ iff over $G_{(\mathcal{K}, s_0, \varphi)}$ player 0 has a winning strategy from vertex (s_0, φ). (For details the reader may consult the recent monograph [Sti01].)

Also fragments and variants of the μ-calculus can be handled in this game theoretical setting. For example, the Computation Tree Logic CTL leads to a weak parity game.

For the solution of parity games, the available upper bounds do so far not allow to infer a polynomial time algorithm. To be specific, one considers the decision problem whether a vertex of a finite game graph of a parity game belongs to the winning region of player 0. It is known that this problem is in NP $\cap$ co-NP, and even in the slightly more restricted complexity class UP $\cap$ co-UP (see [Ju98]). There are several algorithms whose exponential time behavior is due to an exponent $d/2$ where d is the number of colors. Another algorithm (presented in [VJ00]) uses an elegant scheme of "strategy improvement", in which a sequence of positional strategies of player 0 is constructed ending with a uniform winning strategy over the winning region of player 0: Each initial choice of a positional strategy by player 0 is answered by a "best response strategy" of player 1 (again positional), from which player 0 can continue by a local improvement of his strategy, then again obtaining a "best response" of player 1, and so on until player 0 reaches a strategy where no local improvement is possible. (We have to skip the definitions of "best response" and "local improvement" here.) While it is easy to see that each round in this improvement scheme costs only polynomial time, the number of improvement steps can (so far) only be bounded by the number of all possible positional strategies of player 0 (which is exponential in the number of vertices). On the other hand, no family of example graphs seems to be known where the outlined algorithm has to carry out an exponential number of improvement steps.

6 Selected topics of current and future research

We close by listing a number of fields and problems which are subject of present research or seem to be promising future steps in developing the theory of infinite games. A look into the literature will show that this list is far from complete.

1. *Games over pushdown transition graphs.* The core results of the theory over finite game graphs have been lifted to pushdown graphs (see [Wal00]). However, computational results and efficiency considerations now have to incorporate a new parameter, the "size of a state" (which is the length of the word representing the state).
2. *Games over other infinite graphs.* It is well-known that over slightly more general graphs than the pushdown graphs, an algorithmic solution even of the simplest games (reachability games) fails. This is not only true for recursive graphs in general, but also for quite restricted classes like the "ground tree rewriting graphs" (where vertices are finite trees and the edge relation is defined in terms of ground rewriting rules).
3. *Games with winning conditions of Borel level greater than 2.* For infinite graphs, even for pushdown graphs, it is reasonable to consider winning conditions which transcend the level of Muller condition, i.e. which are located on higher levels of the Borel hierarchy, but which still admit algorithmic solutions.
4. *Games over structured transition systems.* The flat representation of a system by a game graph does not, in general, provide an appropriate system model. In distributed systems, different winning strategies have to be devised for different components ([PR90], [MT01]). Another research direction is to provide methods for solving games over hierarchical systems.
5. *Timed systems, optimality of strategies.* The subject of strategy synthesis over timed systems has attracted much attention; see e.g. [AMPS98] and [AM99]. Rather than to optimize strategy automata w.r.t. their number of states, other criteria for optimization become now relevant. For example, one tries to ensure that waiting times between different events are minimized.

6. *Nondeterministic strategies.* In hierarchical design, a way of strategy construction is desirable where the idea of refinement can be applied. Nondeterministic strategies are a possible approach; refinement steps would narrow down the nondeterminism.
7. *Compositional strategy construction.* In contrast to model-checking (where a logical specification can be handled by decomposing a formula according to its construction), we do not know of such a compositional approach for solving infinite games; indeed, their solution rests on the presentation of winning conditions by ω-automata. A practical theory would have to offer a more structured method in which logical operators enter.

References

[AMPS98] E. Asarin, O. Maler, A.Pnueli, J. Sifakis, Controller Synthesis for Timed Automata, Proc. IFAC Symposium on System Structure and Control, 469-474, Elsevier, Amsterdam 1998.

[AM99] E. Asarin, O. Maler, As Soon as Possible: Time Optimal Control for Timed Automata, Hybrid Systems (F. Vaandrager et al. Eds.): Computation and Control, Lecture Notes in Computer Science 1569 (1999), 19-30

[EJS93] E.A. Emerson, C.S. Jutla, A.P. Sistla, On model checking for fragments of μ-calculus, in: CAV'93 (C. Coucoubetis, Ed), Lecture Notes in Computer Science **697** (1993), 385-396.

[DJW97] S. Dziembowski, M. Jurdzinski, I. Walukiewicz, How much memory is needed to win infinite games?, Proc. 12th IEEE Symp. on Logic in Computer Science, 1997, 99-110.

[Ju98] M. Jurdzinski, Deciding the winner in parity games is in UP$\cap$co-UP, *Inform. Processing Letters* **68** (1998), 119-124.

[KV99] O. Kupferman, M.Y. Vardi, Church's problem revisited, *Bull. Symb. Logic* **5** (1999), 245-263.

[MP92] Z. Manna, A. Pnueli, *The Temporal Logic of Reactive and Concurrent Programs*, Springer-Verlag, Berlin-Heidelberg-New York 1992.

[MT01] P. Madhusudan, P. S. Thiagarajan: Distributed Controller Synthesis for Local Specifications. Proc. ICALP 2001, Lecture Notes in Computer Science 2076 (2001), 396-407

[PR90] A. Pnueli, R. Rosner, Distributed reactive systems are hard to synthesize, Proc. 31st IEEE Symp. on Foundation of Computer Science, 1990, 746-757.

[Sti01] C. Stirling, *Modal and Temporal Properties of Processes*, Springer-Verlag, New York 2001.

[Th95] W. Thomas, On the synthesis of strategies in infinite games, in: STACS'95 (E.W. Mayr, C. Puech, Eds.), Lecture Notes in Computer Science **900** (1995), 1-13.

[Th97] W. Thomas, Languages, automata, and logic, in: *Handbook of Formal Languages* (G. Rozenberg, A. Salomaa, Eds.), Vol. 3, Springer-Verlag, Berlin Heidelberg 1997.

[VJ00] J. Vöge, M. Jurdzinski, A strategy improvement algorithm for solving parity games, CAV 2000, Lect. Notes in Computer Science **1855** (2000), 202-215.

[Wal00] I. Walukiewicz, Pushdown processes: games and model-checking, *Information and Computation* **157** (2000), 234-263.