Theoretical Computer Science 521 (2014) 73–91

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Alternating traps in Muller and parity games

Andrey Grinshpun [a], Pakawat Phalitnonkiat [b], Sasha Rubin [c,*], Andrei Tarfulea [d]

[a] Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA, United States
[b] Department of Mathematics, Cornell University, Ithaca, NY, United States
[c] IST Austria and TU Vienna, Austria
[d] Department of Mathematics, Princeton University, Princeton, NJ, United States

ARTICLE INFO

Article history:
Received 24 November 2010
Received in revised form 23 October 2013
Accepted 26 November 2013
Communicated by A. Fraenkel

Keywords:
Parity games
Muller games

ABSTRACT

Muller games are played by two players moving a token along a graph; the winner is determined by the set of vertices that occur infinitely often. The central algorithmic problem is to compute the winning regions for the players. Different classes and representations of Muller games lead to problems of varying computational complexity. One such class are parity games; these are of particular significance in computational complexity, as they remain one of the few combinatorial problems known to be in NP ∩ co-NP but not known to be in P. We show that winning regions for a Muller game can be determined from the alternating structure of its traps. To every Muller game we then associate a natural number that we call its *trap depth*; this parameter measures how complicated the trap structure is. We present algorithms for parity games that run in polynomial time for graphs of bounded trap depth, and in general run in time exponential in the trap depth.

1. Introduction

A Muller game [13,7] is played on a finite directed graph in which every vertex is given a label, either red or blue. There is a token on an initial vertex and two players, call them Red and Blue, move the token along edges; it is Red's move if the token is on a red vertex, and otherwise it is Blue's move. To determine the winner, a Muller game also contains a collection $\mathcal{R}$ of sets of vertices. One assumes that there are no dead ends and so the play is an infinite walk. At each turn one records the vertex under the token. The winner is determined by the set S of vertices that occur infinitely often; Red wins if S is in $\mathcal{R}$, and otherwise Blue wins.

Every two-player perfect-information game with Borel winning condition is determined [12]: one of the players has a winning strategy. In particular, every Muller game is determined: either Red or Blue has a winning strategy. To *solve a Muller game* is to determine for every vertex which player has a winning strategy when play starts from the given vertex. This set of vertices is called that player's *winning region*.

One application of these games is to solve Church's synthesis problem: construct a finite-state procedure that transforms any input sequence letter by letter into an output sequence such that the pair of sequences satisfies a given specification. The modern solution to this problem goes through Muller games [17].

Characterization of Muller games. The first part of this paper (Section 3.1) characterizes the winning region of a Muller game G in terms of a two-player reachability game. The length of this reachability game is a measure of the alternating structure of the traps in G; we call it the *trap-depth* of G. We briefly explain these concepts.

* Corresponding author.
E-mail addresses: agrinshp@mit.edu (A. Grinshpun), pp287@cornell.edu (P. Phalitnonkiat), sasha.rubin@gmail.com (S. Rubin), tarfulea@princeton.edu (A. Tarfulea).

0304-3975/$ – see front matter © 2013 Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.tcs.2013.11.032

Muller games admit natural substructures, *Attractors* and *Traps*. The Red-attractor [18] of a subset X of vertices is the set of vertices from which Red can force the token into X; this may be computed in linear time. A Red-trap [18] is a subset Y of vertices in which Blue may keep the token within Y indefinitely (no matter what Red does); i.e. if the token is in Y, Blue may choose to trap Red in the set Y. It should be evident that the complement of a Red-attractor is a Red-trap. Of course, all notions here (and elsewhere) defined for Red may be symmetrically defined for Blue. Thus we talk of Blue-attractors and Blue-traps.

Now, consider the following game played on the same arena as a Muller game G. The *trap-depth game on G in which Red goes first* (Definition 3.2) proceeds as follows (the traps discussed in the following are all nonempty): Red picks a Blue-trap $X_1 \subseteq V$ (here V are the vertices of the Muller game G) which is winning for Red (i.e. $X_1 \in \mathcal{R}$). Then Blue picks a Red-trap Y_1 in the smaller game induced by X_1, where Y_1 is winning for Blue (i.e. $Y_1 \notin \mathcal{R}$). Then Red picks a Blue-trap X_2 in the game induced by Y_1 such that X_2 is winning for Red. Red and Blue continue like this, alternately choosing traps. The first player that cannot move (i.e., that cannot find an appropriate nonempty trap) loses. As shown in Theorem 3.4,

> Red has a nonempty winning region in the Muller game if and only if Red has a winning strategy in the trap-depth game in which Red goes first.

And if Red has a winning strategy in this trap-depth game, the first move of any winning strategy, X_1, contains only vertices in Red's winning region of the original Muller game.

Application to parity games. The second part of the paper (Section 4) is algorithmic and applies the characterization of winning regions to a particular class of Muller games, parity games.

A parity game [4] is played on a directed graph with vertices labeled by integers called *priorities*. This game is played between two players, Even and Odd, who move a token along edges. A vertex is called *even* if its priority is even, otherwise it is called *odd*. Even moves when the token is on an even vertex, and Odd moves when the token is on an odd vertex. Play starts from a specific vertex; we assume there are no dead ends in the graph and so a play is an infinite walk. Even wins a play if the largest priority occurring infinitely often is even, otherwise Odd wins the play.

It is evident that parity games may be expressed as Muller games: the set $\mathcal{R}$ consists of all subsets X of vertices in which the largest priority of vertices in X is even.

Parity games are intertwined with a logical problem: the model checking problem for Modal μ-calculus formulas is log-space equivalent to solving parity games [7]. In [15] we see how this work can be applied for software verification. Complexity-wise, the problem is known to be in NP ∩ co-NP [5], and even UP ∩ co-UP [9]: one of the few combinatorial problems in that category that is not known to be in P. A randomized algorithm was presented in [3] that runs subexponentially (in the size of the input game) in the worst case. Several prior results obtained polynomial-time algorithms for parity games with fixed bounds on certain structural qualities, such as DAG-width; see for instance [1], [2], [8], and [14]. Our approach is in the same vein, with a novel structural quality given by the trap-depth game.

The algorithmically-minded reader may observe a potential drawback with reinterpreting games as a game of alternating traps. The number of traps in a game can grow exponentially with the size of the game (just take a graph with only self-loops), and what's worse is that we are looking at chains of alternating traps. Nonetheless, we apply the characterization to parity games: say that a graph has *Even trap-depth at most k* if Even can guarantee that, in the trap-depth game in which Even goes first, the game ends in a win for Even within k rounds. Then, despite the previous observation, we present an algorithm $\mathrm{TDA}(G, \sigma, k)$ (here G is a parity game, σ is a player, and k an integer) that runs in time $|G|^{O(k)}$ and, as shown in Theorem 4.1,

> returns the largest (possibly empty) set starting with which σ can guarantee a win in at most k moves in the trap-depth game on G.

Note that the definition of trap depth may be applied to Muller games as well, though we do not have an algorithmic application; one might hope that there are particularly efficient algorithms for finding winning vertices in Muller games of small trap depth.

Let's put this all together. Say that a parity game has *trap-depth at most k* if either it has Even trap-depth at most k or Odd trap-depth at most k. In Fig. 1 we exhibit, for every integer k, a parity game with $O(k)$ vertices and edges that has trap-depth exactly k. By the end of the paper we will have algorithmically solved the following problems:

(1) decide if a given parity game G has trap-depth at most k.
(2) find a nonempty subset of one of the player's winning region assuming the game has trap depth at most k.

Moreover, these problems can be solved in time $O(mn^{2k-1})$ where n is the number of vertices and m the number of edges of a parity game G.

2. Muller games and parity games

A *Muller game* $G = (V, V_{\mathrm{red}}, E, \mathcal{R})$ satisfies the following conditions: (V, E) is a directed graph in which every vertex has an outgoing edge, V is partitioned into *red vertices* V_{red} and *blue vertices* $V_{\mathrm{blue}} := V \setminus V_{\mathrm{red}}$, and $\mathcal{R} \subset 2^V$ is a collection